



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

1. Bevezetés	3
2. Felhasználói dokumentáció	4
2.1. A szoftver rövid ismertetése	4
2.2. Célközönség	4
2.3. Első üzembe helyezés és indítás	4
2.4. Általános felhasználói tájékoztató	4
2.5. A rendszer funkcióinak bemutatása	5
2.5.1. Regisztráció és Bejelentkezés	5
2.5.2. Navbar funkciói	6
2.5.3. Dashboard funkciói	7
2.5.4. DayView funkciói	8
2.5.5. Ticketek funkciói	9
2.5.6. Naptárak funkciói	10
2.5.7. Profile funkciói	11
2.6. Futás közbeni rendszerüzenetek	11
2.6.1. Hibaüzenetek és jelentésük	11
2.6.2. Figyelmeztetések	11
3. Fejlesztői dokumentáció - Tervezés és megvalósítás	12
3.1. Rendszer architektúrája	12
3.1.1. Alrendszerek és rétegek szerepei és felelősségei	12
3.1.2. Alkalmazott technológiák és eszközök	13
3.1.3. Fejlesztési módszertan	15
3.1.4. Csomag diagram (UML)	16
3.2. A rendszer működése	16
3.2.1. Kommunikációs diagram (UML)	17
3.2.2. Tevékenységdiagram (UML)	18

3.3.	Adatbázis modell	19
3.3.1.	Az adatmodell áttekintése	20
3.3.2.	Tábla-szintű leírás	20
3.4.	Modul- és osztályszerkezet	23
3.4.1.	Backend modulok és rétegek	23
3.4.2.	Osztálydiagram (UML)	23
3.4.3.	Főbb osztályok leírása és implementációja	25
3.5.	A felhasználói felület	29
3.5.1.	Csomagdiagramm (UML)	30
3.5.2.	Képernyők navigációs logikája	30
3.5.3.	Felhasználói események kezelése	32
3.6.	Telepítési folyamat leírása (lokálisan és szerveren)	33
4.	Fejlesztői dokumentáció - Tesztelés	34
4.1.	Tesztelési stratégia	34
4.1.1.	Tesztfájlok fizikai elhelyezkedése (UML)	35
4.1.2.	Teszteléshez használt eszközök	35
4.2.	Egységtesztek	36
4.3.	Integrációs tesztek	37
4.4.	Rendszertesztek - automatizált	37
4.5.	Rendszertesztek - Manuális	37
4.6.	Tesztelési eredmények	45
4.6.1.	Backend coverage	45
4.6.2.	Frontend e2e teszt megtekintése	48
5.	Összegzés	49

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, tárgyilagossági megfogalmazás

Mondatok szerkezete: Tény megállapítása + következtetés

2. fejezet

Felhasználói dokumentáció

2.1. A szoftver rövid ismertetése

2.2. Célközönség

2.3. Első üzembe helyezés és indítás

2.4. Általános felhasználói tájékoztató

szerintem ez
csak annyi lesz
itt, hogy meg-
nyitja a linket
és szoszi

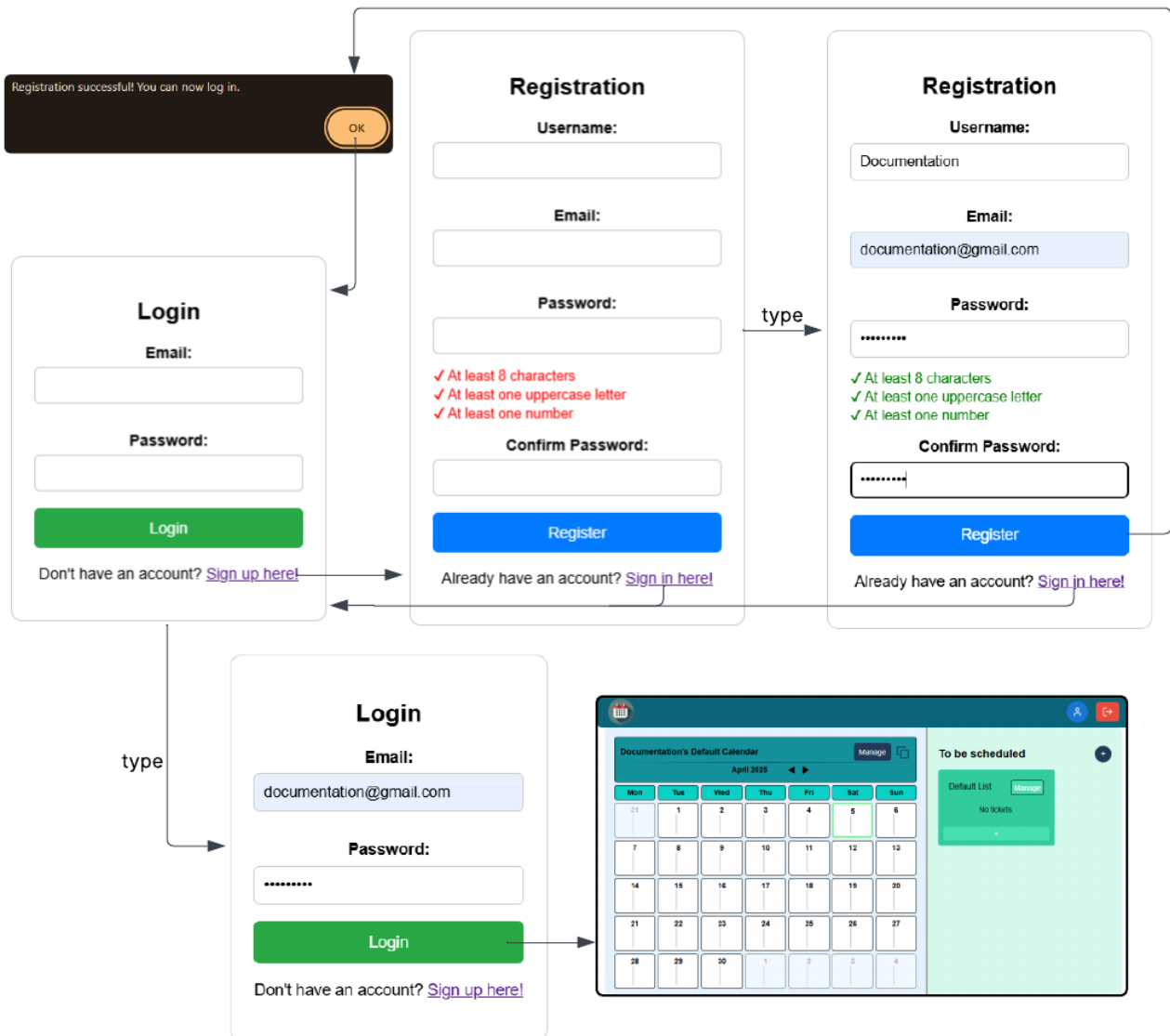
Az oldal tartalmaz tooltipeket. Amennyiben a felhasználó egy elem fölé helyezi a kurzorát és kis ideig nem mozgatja megjelenik egy üzenet amely segít az ott elérhető interakciók értelmezésében.

Az oldalon található ticketekkel kapcsolatban drag and drop mechanizmus él. Azaz a felhasználó az egér lenyomásával meg tudja fogni az adott jegyet és mozgatni azt. (pl naptárba kiosztani, vagy az adott listán belül újrapozícionálni.)

A modalokon megjelennek piros csillagok a kötelezően kitöltendő input mezők mellett, amennyiben ezek mégsem kerülnek kitöltésre hibaüzenetek jelzik azok hiányát.

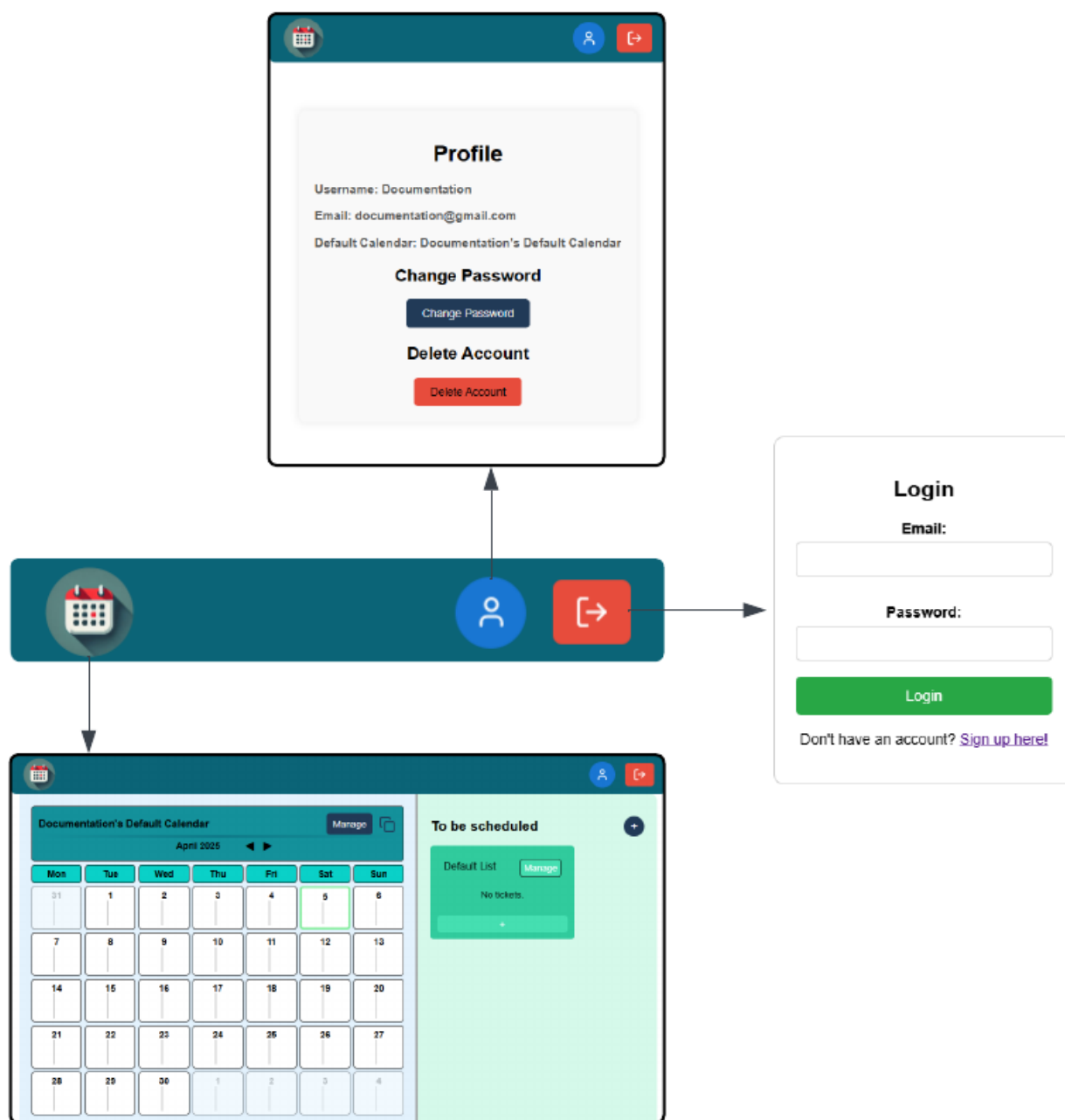
2.5. A rendszer funkcióinak bemutatása

2.5.1. Regisztráció és Bejelentkezés



2.1. ábra. Regisztráció és Bejelentkezés

2.5.2. Navbar funkciói



2.2. ábra. Navbar

2.5.3. Dashboard funkciói



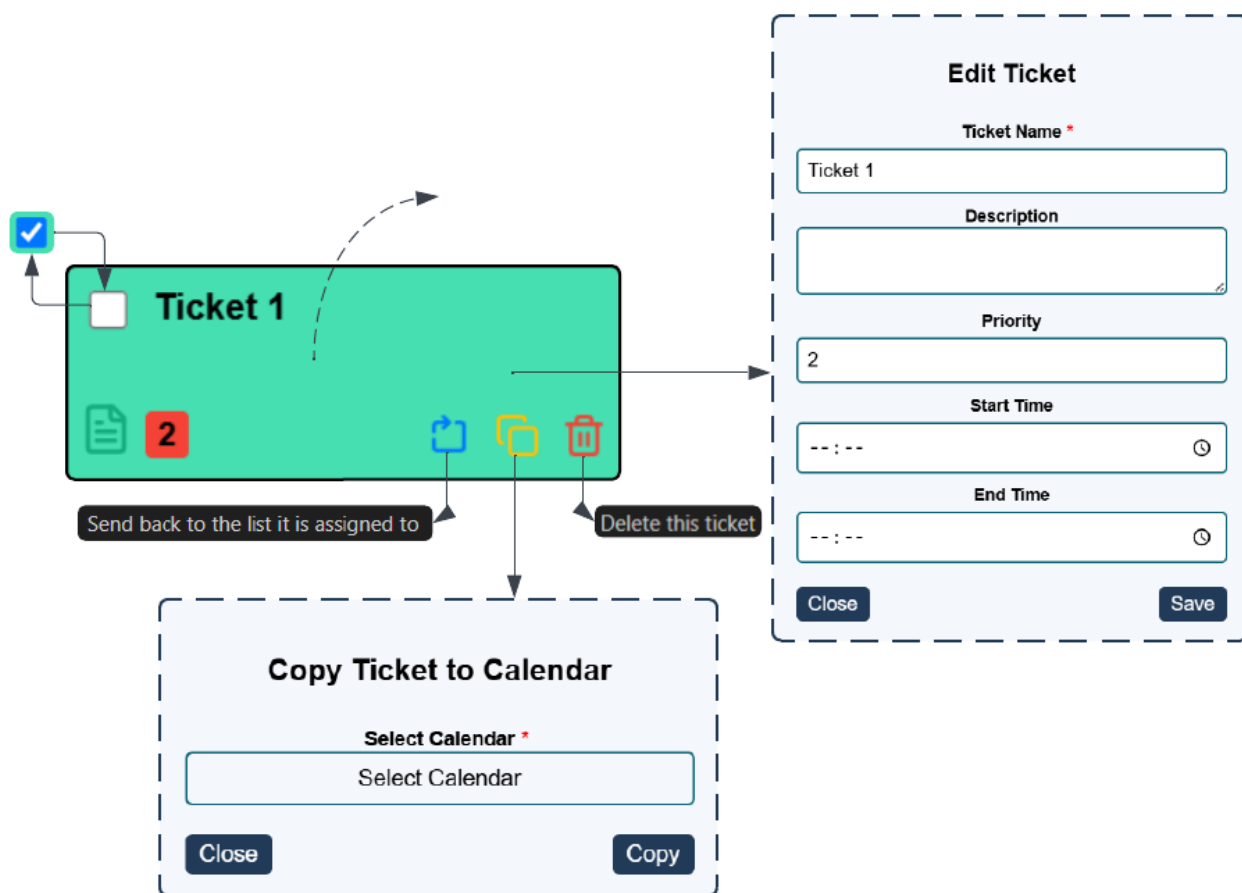
2.3. ábra. Dashboard

2.5.4. DayView funkciói



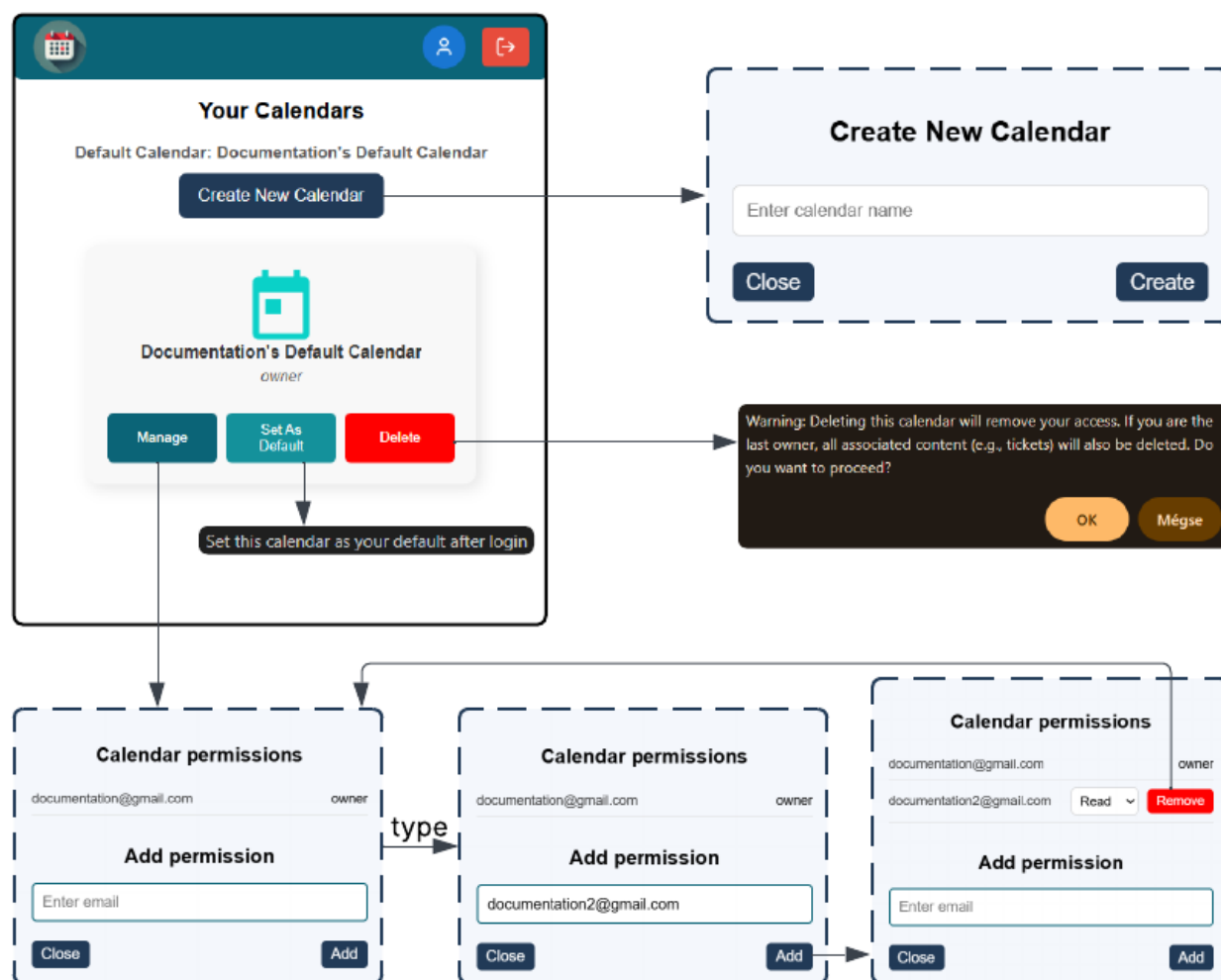
2.4. ábra. DayView

2.5.5. Ticketek funkciói



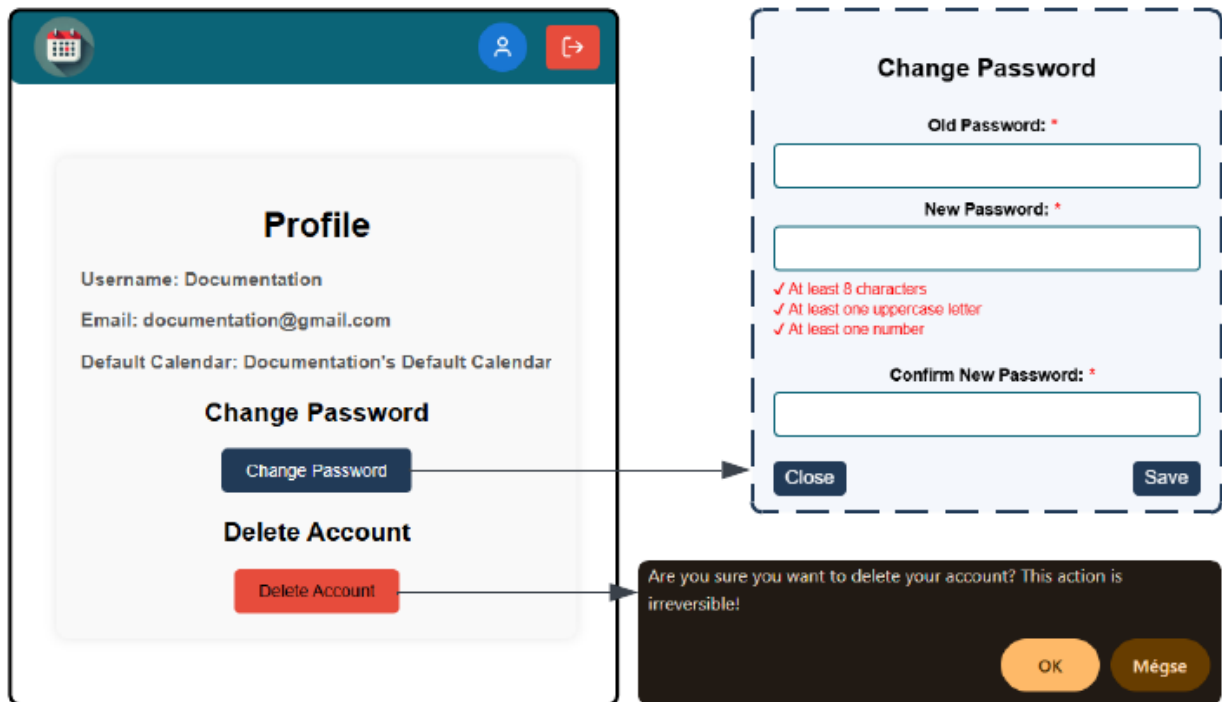
2.5. ábra. Ticketek

2.5.6. Naptárak funkciói



2.6. ábra. Naptárak

2.5.7. Profile funkciói



2.7. ábra. Profil

2.6. Futás közbeni rendszerüzenetek

2.6.1. Hibaüzenetek és jelentésük

Hibaüzenet valamilyen kritikus hibára hívja fel a figyelmet. Leggyakrabban akkor ha valamilyen adat helytelenül kerül kitöltésre, vagy amennyiben egy olyan naptárban szeretne interakciót végrehajtani a felhasználó amelyhez nincs megfelelően magas engedélye.

A hibaüzenetek megjelenésüket követően 5 másodperccel eltűnnek.

2.6.2. Figyelmeztetések

Az oldal fontos, vissza nem csinálható műveletek előtt megerősítést kér arról, hogy a felhasználó biztosan el szeretné-e végezni az adott műveletet. Ilyen például calendar list-ek, naptárak vagy felhasználói profilok törlése.

3. fejezet

Fejlesztői dokumentáció - Tervezés és megvalósítás

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.

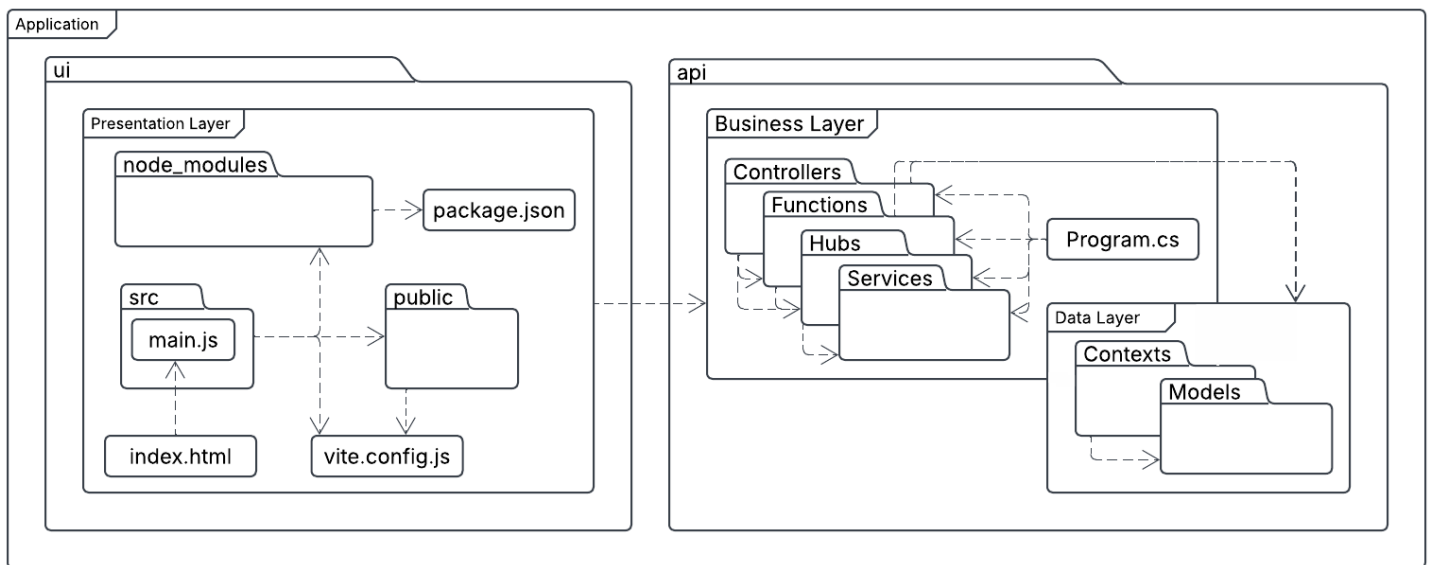
3.1.3. Fejlesztési módszertan

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 leírhatom hogy vettem fel issuekat gitHubon

3.1.4. Csomag diagram (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 a 3.5.1.-es pont alatt a 3.20. ábra mutatja az src mappa csomag diagramját.

3.2. A rendszer működése

(3.2., 3.3, 3.4, 3.5) A működést 2 féle diagrammal mutatom be, az első típus^(3.2., 3.3) hangsúlyt fektet a kommunikáció módjának bemutatására míg a második típus^(3.4., 3.5) 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

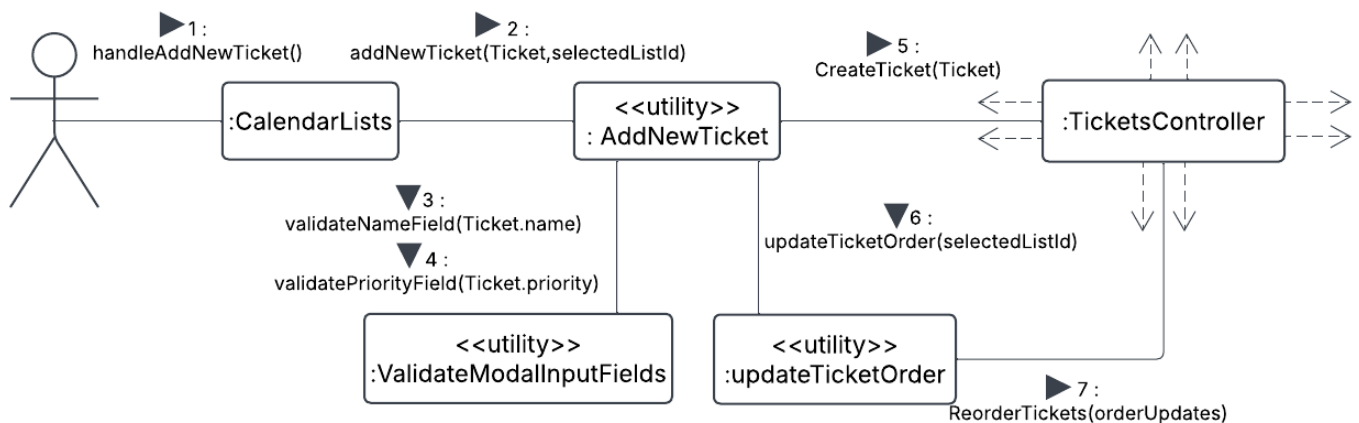
nem kötött segédfüggvények is, melyeket «utility» sztereotípiával és a fent említett névadási konvenciókkal fogunk megjeleníteni a lent látható diagramokban.

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)

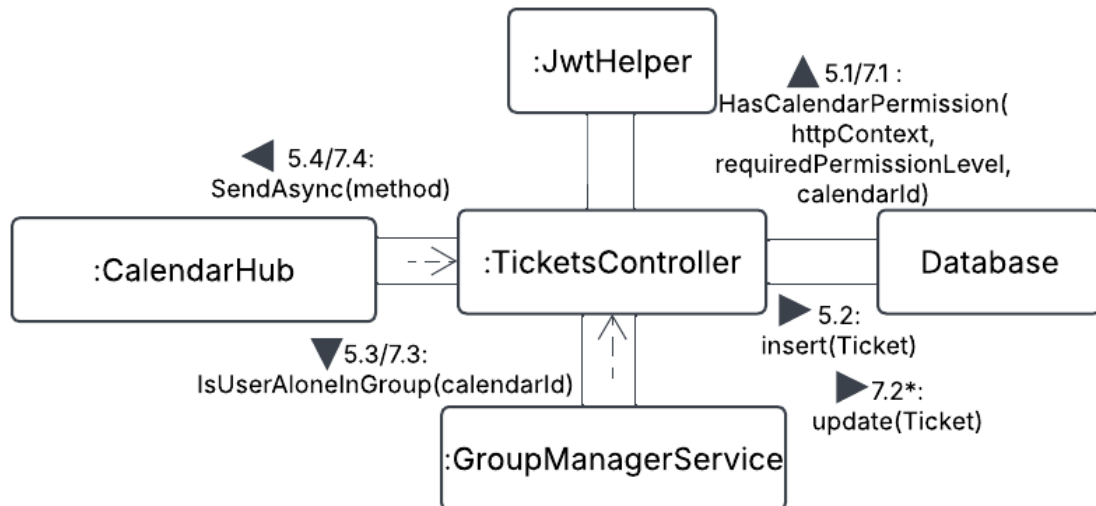
(3.2.) Az alábbi diagrammon látható egy sikeres folyamat működése, a diagramnak hála jól láthatóak lesznek a kommunikációban részt vevő objektumok kapcsolatai és leolvasható lesz a kommunikáció sorrendje is.

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



3.2. ábra. Kommunikációs diagram, kliens

(3.3.) 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. A 7.2-es lépésben a "*" szimbólum az iterációs lehetőséget rejti, azaz esetünkben a hívás a frissíteni kívánt ticketek száma szerint fut le.

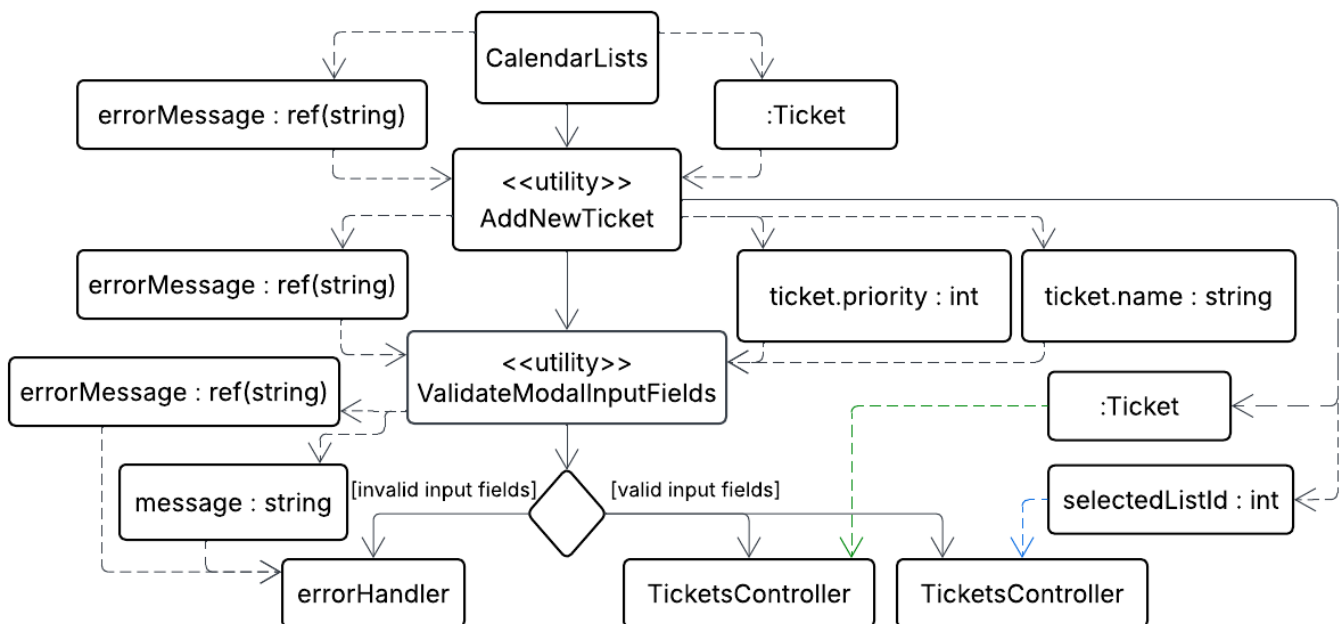


3.3. ábra. Kommunikációs diagram, szerver

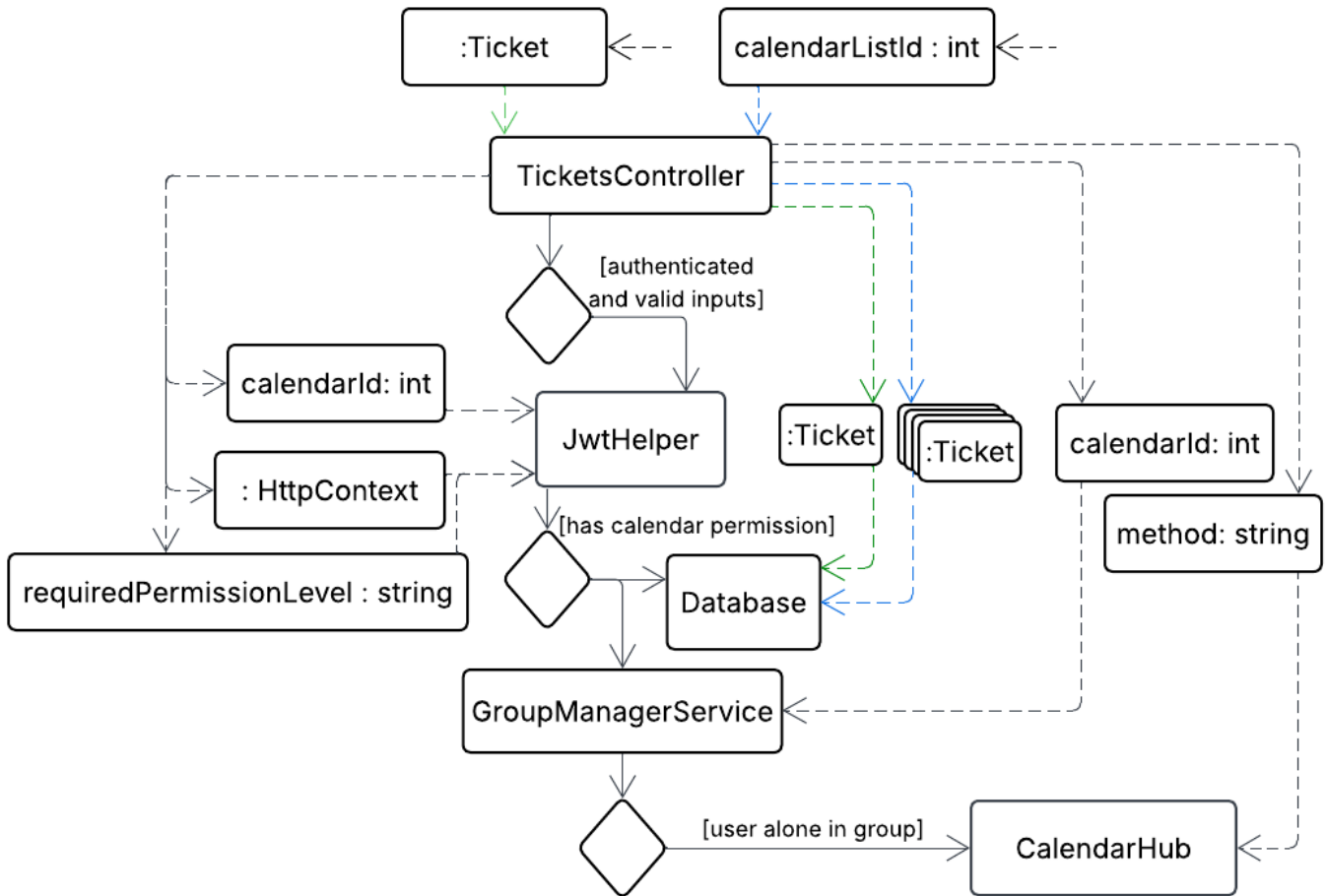
3.2.2. Tevékenységdiagram (UML)

(3.4.) 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^(3.5.) is, hogy pontosan melyik tevékenység melyikhez tartozik.



3.4. ábra. Tevékenység diagram, kliens



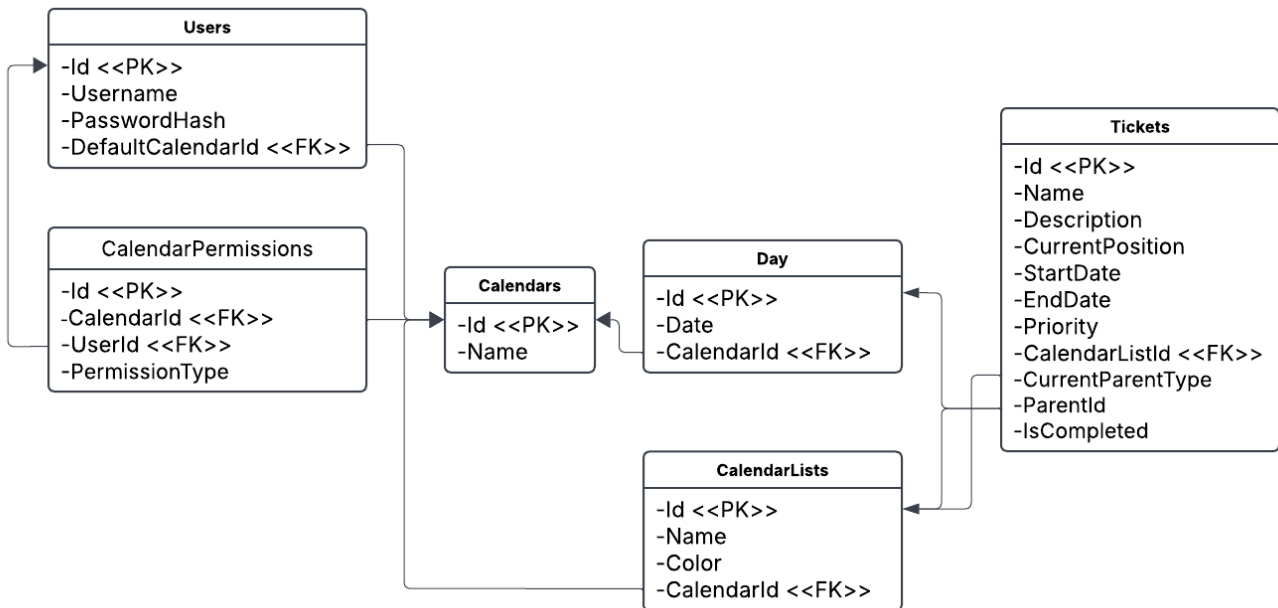
3.5. ábra. Tevékenység diagram, szerver

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.

^(3.6.)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




3.6. ábra. Adatbázis entitások és kapcsolatok

3.3.2. Tábla-szintű leírás



(3.7., 3.8., 3.9., 3.10., 3.11., 3.12.) 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.

(3.7.) 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"/>



3.7. ábra. Calendars

(3.8.) 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"/>

3.8. ábra. Days



(3.9.) 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"/>

3.9. ábra. CalendarLists




(3.10.) 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 nézetet 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"/>




3.10. ábra. Tickets

(3.11.) 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"/>

3.11. ábra. Users

(3.12.) 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"/>

3.12. 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

A backend a Program.cs fájlban építi fel az Api fő komponenseit és azok közötti megfelelő kommunikációt. Az ApplicationBuilder példány létrehozásával kezdődik, amely az alkalmazás konfigurálásának alapját képezi. Majd sorra kerülnek a CORS szabályok, SignalR szolgáltatás, Adatbázis-kapcsolat, JWT alapú hitelesítés konfigurálása valamint további egyedi szolgáltatások regisztrálása Dependency Injection-nel. Amint ezek mind helyesen beállításra kerültek az alkalmazás futtatása következik. A WebApplication amikor objektumpéldányra van szüksége megkéri a DI konténert hogy a Builderben meghatározott szabályok szerint adjon neki egy objektumpéldányt.

3.4.2. Osztálydiagram (UML)

Az UML-en nincs jelölve az olvashatóság kedvéért de amennyiben egy objektum rendelkezik egy adattaggal amely valamelyik másik objektumot igényli az DI-al fog bekerülni az adott objektumba példányosítás során, melynek helyes kezelését a WebApplication DI konténere végzi. Ez az uml-ben 3 helyen szerepel «injected» sztereotípiával jelölve, ezen kapcsolat egyszerűsítve van az olvashatóság kedvéért, a 3.4.3-as pontban pontos kifejtésre kerül pontosan melyik objektum hova injektálódik. ^(3.13.)Nem szokványos sztereotípiák magyarázata:

- «registers scoped», «registers singleton»

A WebApplicationBuilder végzi ezeknek az osztályoknak a regisztrálását a

DI konténerbe. A «registers scoped» jelzi, hogy az osztály példánya HTTP-kérésenként egyszer jön létre. Míg a «registers singleton» összesen 1 példány születik.

- «build»

A `WebApplicationBuilder` a `Build()` metóduson keresztül hozza létre a `WebApplication` példányt.

- «routes»

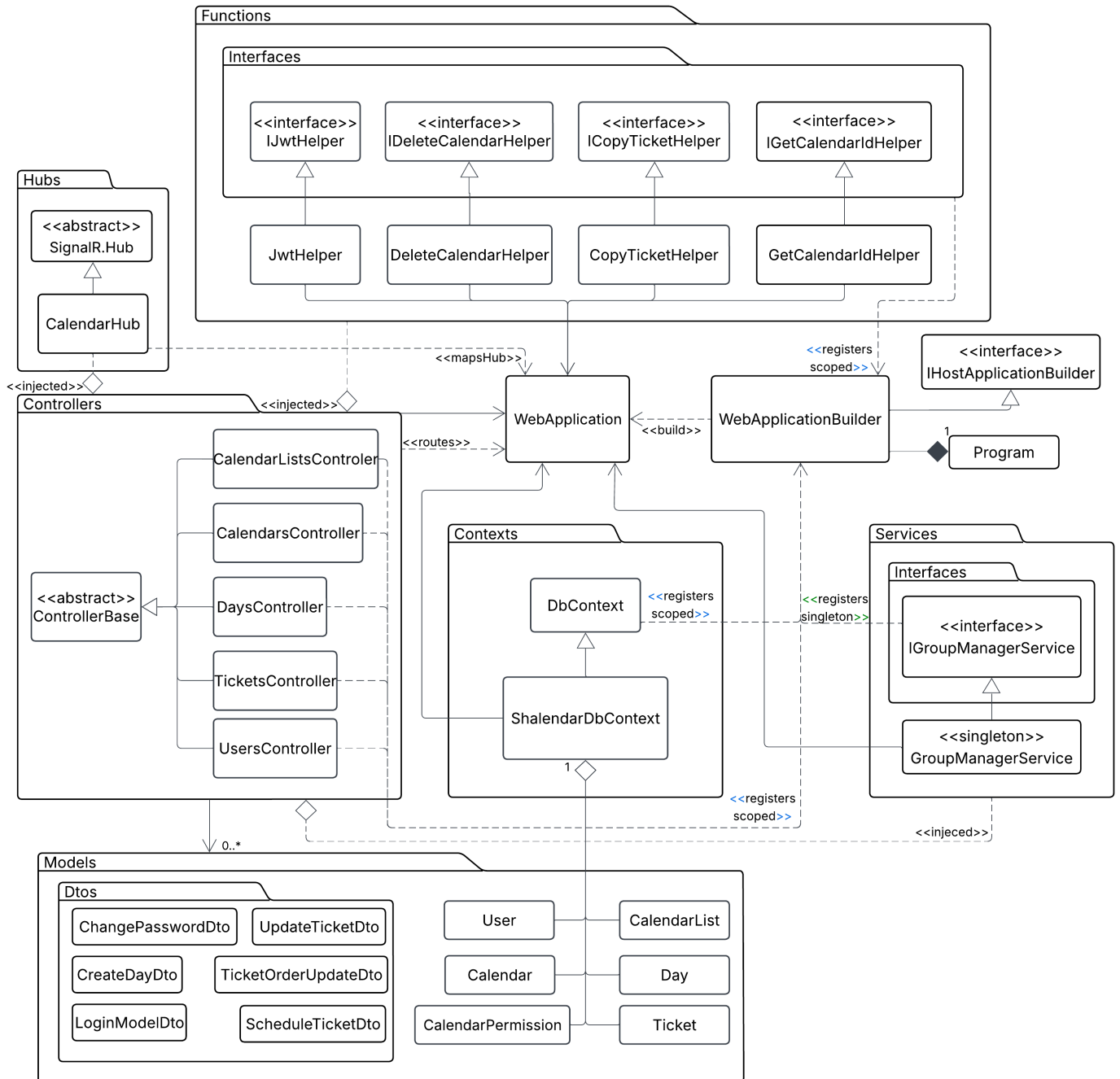
A `WebApplication` automatikusan hozzárendeli a HTTP útvonalakat a `Controller` típusú osztályokhoz a `MapControllers()` segítségével.

- «mapsHub»

Jelzi, hogy egy SignalR Hub végpont (`/calendarHub`) regisztrálásra kerül a `WebApplication` pipeline-jában.

- «injected»

Azt mondja hogy ezek az objektumok bele injektálódtak a mutatott objektumba a `WebApplication` DI konténere által.

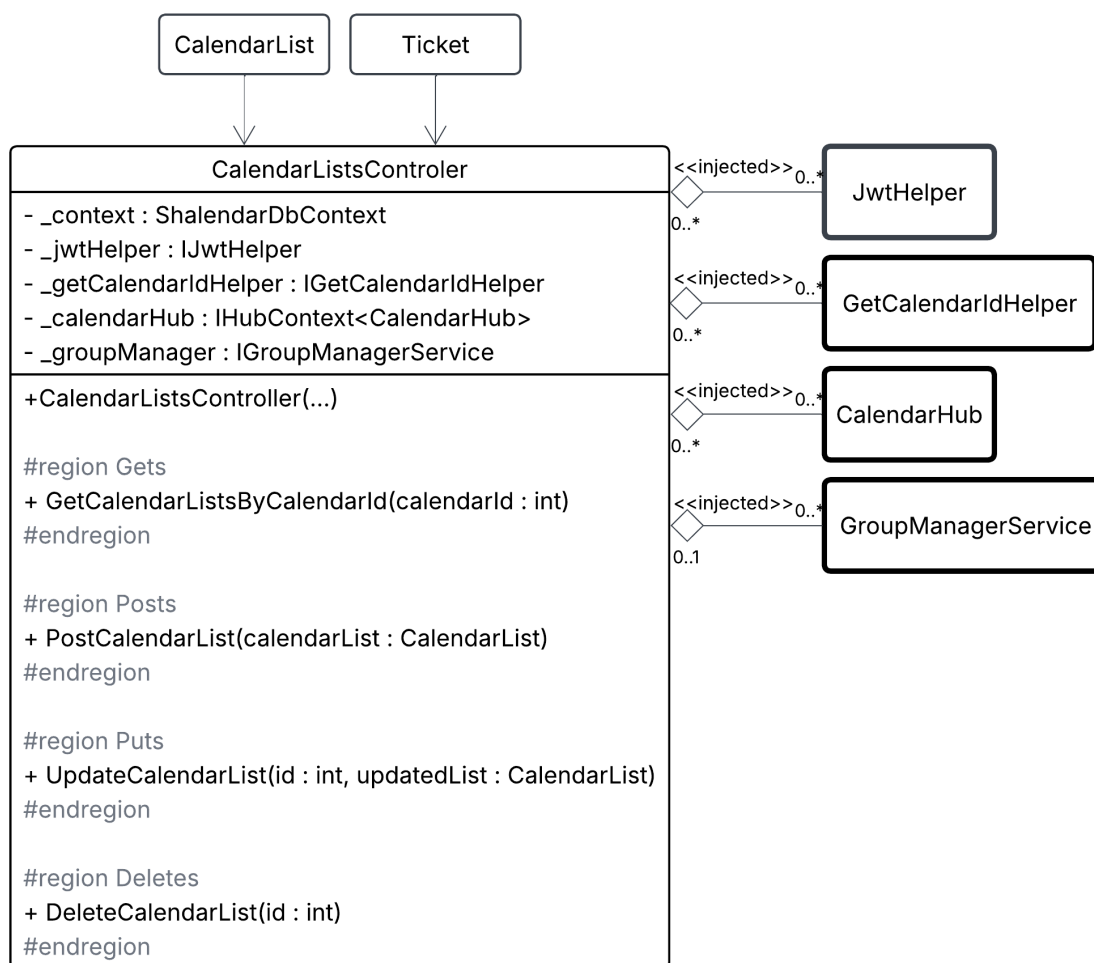


3.13. ábra. Api UML osztálydiagram

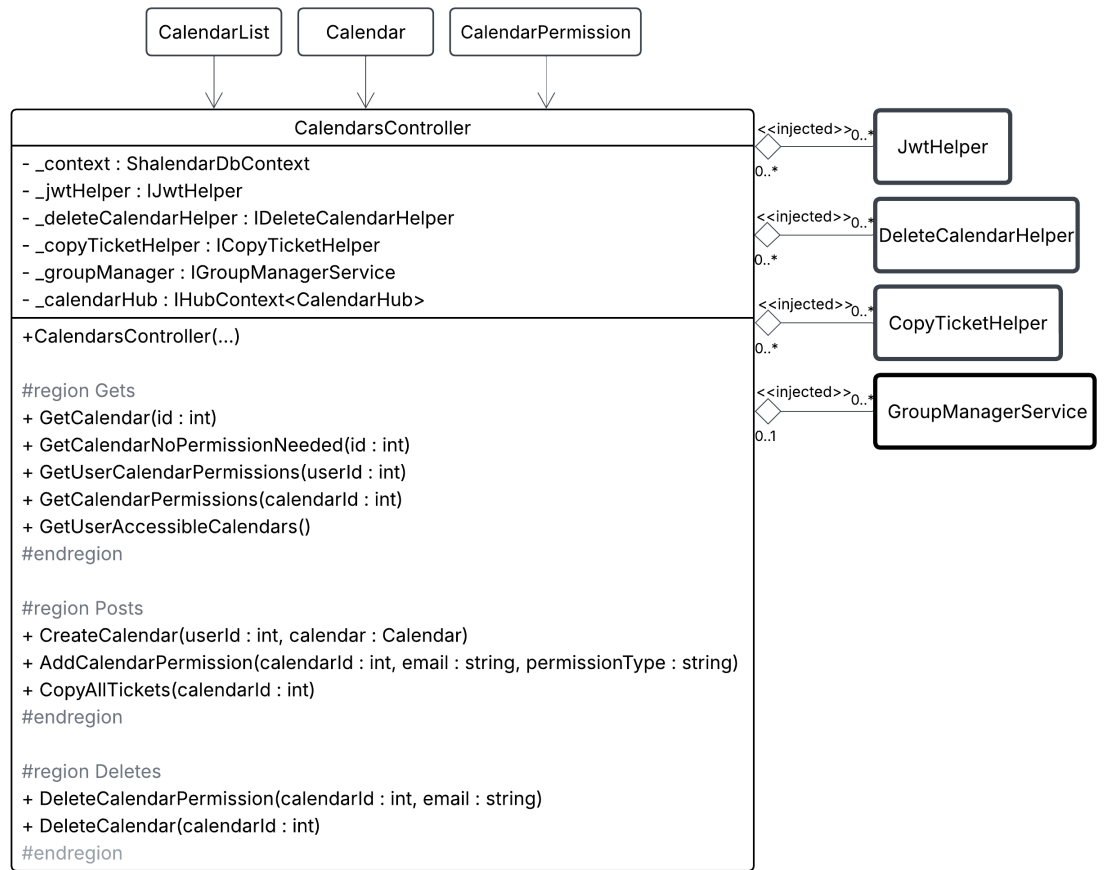
3.4.3. Főbb osztályok leírása és implementációja

(3.14., 3.15., 3.16., 3.17., 3.18., 3.19.) Alábbiakban látható a controllerek és fontosabb osztályok részletes felépítése valamint a modellekkel, Dto-kal és helper függvényekkel való kapcsolata. A dto-k szaggatott vonalas objektumként vannak jelölve a helper objektumok pedig vastag vonallal a könnyebb átláthatóság érdekében. A controlle-

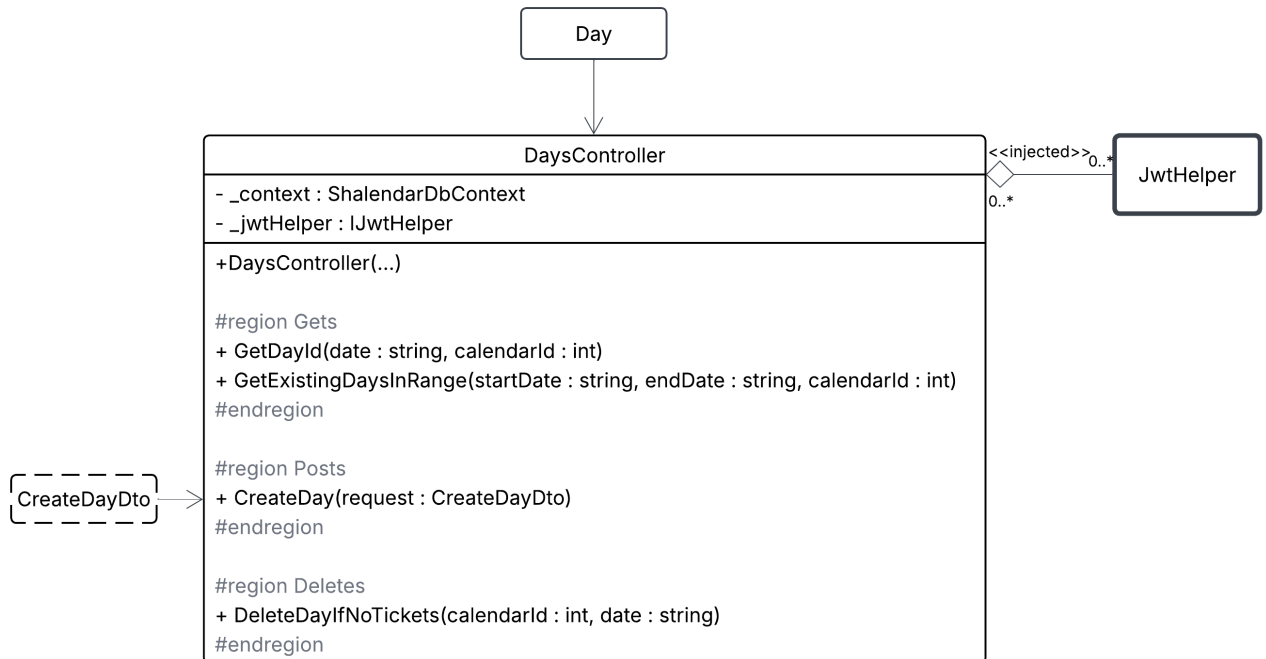
rekben láthatóak regionok, ezek a logikai elkülönítés vizuális szemléltetésére vannak, a forráskódban szintén szerepelnek.



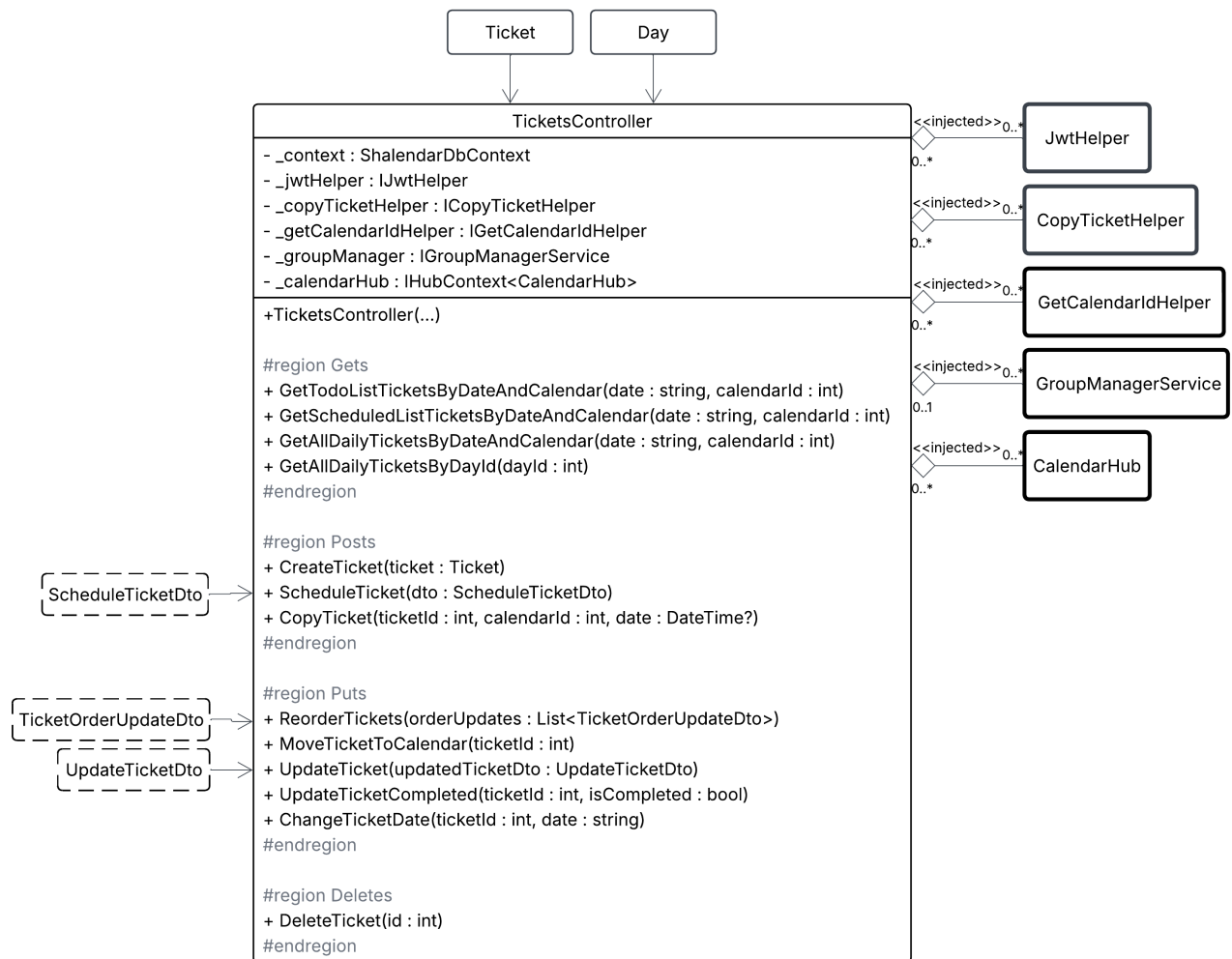
3.14. ábra. CalendarLists



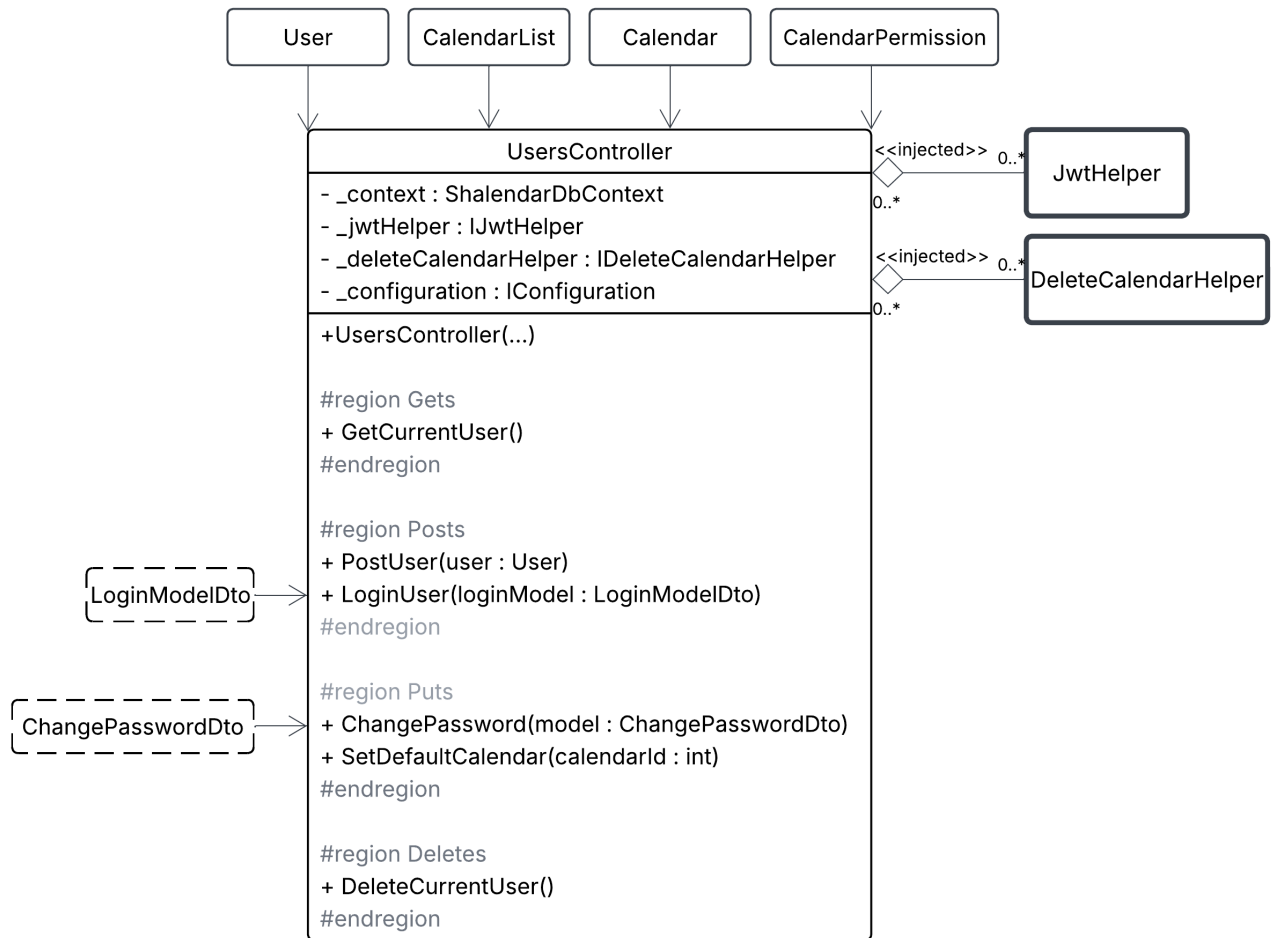
3.15. ábra. Calendars



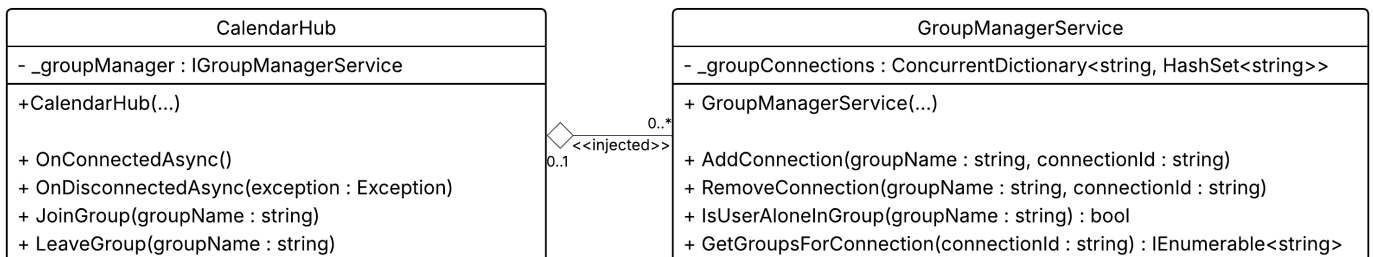
3.16. ábra. Days



3.17. ábra. Tickets



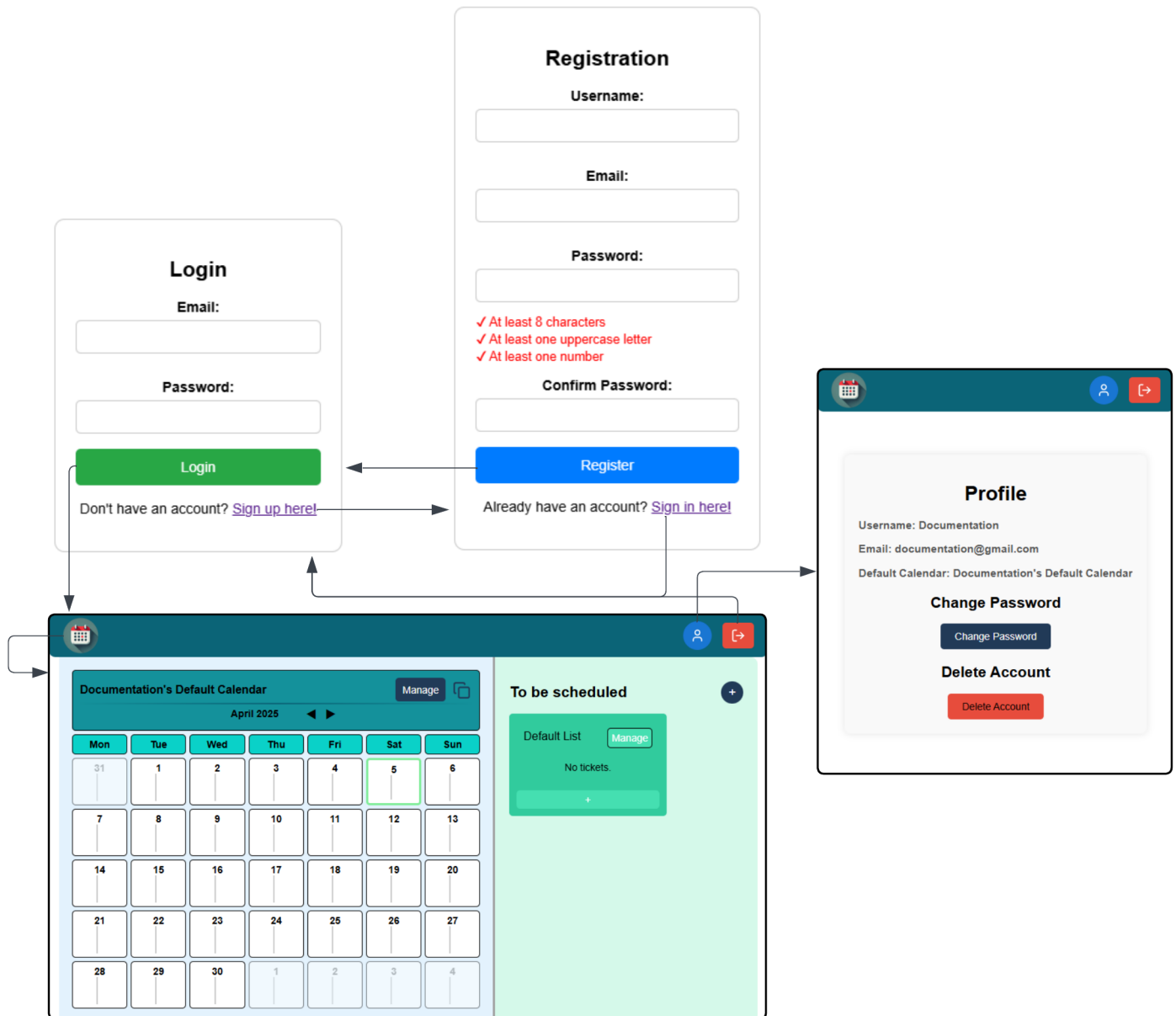
3.18. ábra. Users



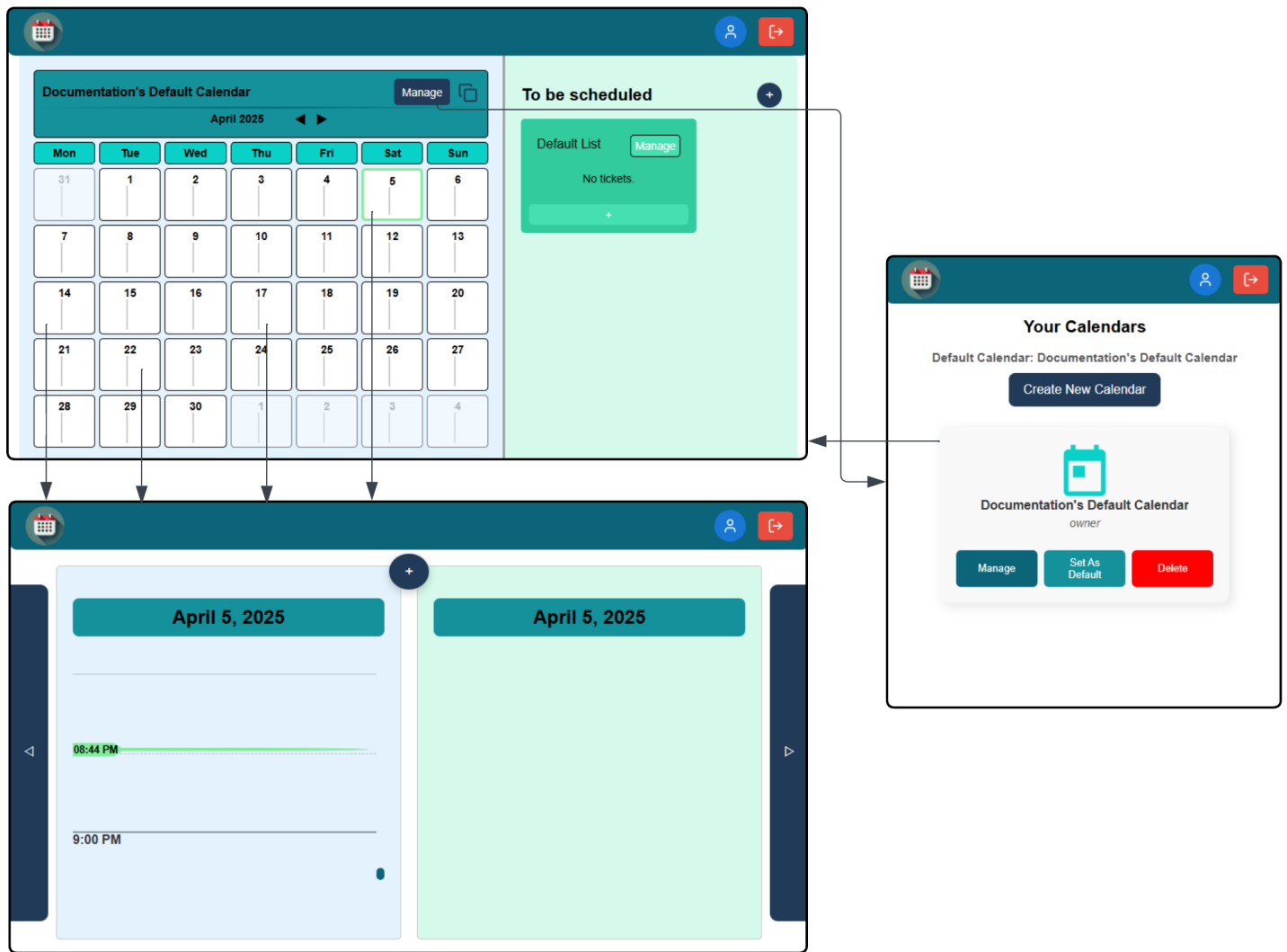
3.19. ábra. CalendarHub és GroupManagerService

3.5. A felhasználói felület

A felhasználó az App.vue-ban meghatározott navbart látja, valamint az alá betöltődő oldalakat. 6 fő view van, ezek a nézetek között tud navigálni a felhasználó. Ezek közül speciális a DayView valamint a Dashboard, mivel ők 2 további fő komponensből állnak. A Views/DayView bal oldali része az organism/ScheduledPanel.vue míg



3.21. ábra. Src mappa csomag diagrammja



3.22. ábra. Src mappa csomag diagrammja

3.5.3. Felhasználói események kezelése

Főbb interakciók leírása

- **Gombok nyomása:** Amennyiben a felhasználó megnyom egy gombot a felületen 2 dolog történhet, vagy betölti neki a kívánt felületet vagy megnyílik egy modal. Modal alatt egy felugró ablakot értünk amelyen fel tud vinni kívánt értékeket majd valamilyen függvényt meghívni azokkal.
- **Drag and drop:** A ticketekkel kapcsolatban különböző "húzási" funkciók vannak. Például ticketek újrendezése egy calendarList-en belül. Vagy a naptár napjára való kiosztás, esetleg a DayView oldalsó nyilaira való ejtés során az előre/hátra küldés egy nappal.

- **Tooltips:** A legtöbb interakció melyet a felhasználó elvégezhet tooltippekkel van ellátva, azaz ha a kurzort egy helyben a elem felett tartja megjelenik milyen interakciók lehetségesek vele.
- **Scroll események:** Hosszú naplista vagy jegylista esetén, vagy ha valamilyen ui elem nem fér el a képernyőben.

Hibakezelés

A felhasználó, vagy a rendszer nem tud a felhasználó számára nem lekezelt hibát megjeleníteni. Minden esetben pirosan megjelenik egy specifikus hibaüzenet, majd 5mp után eltűnik. Ezen hibakezelési logikát a `utils/errorHandler.js` végzi.

3.6. Telepítési folyamat leírása (lokálisan és szerveren)

Eldönteni kell-e majd megírni. (kell szerintem, szerverre is ki kéne tenni és azt is leírni hogyan)

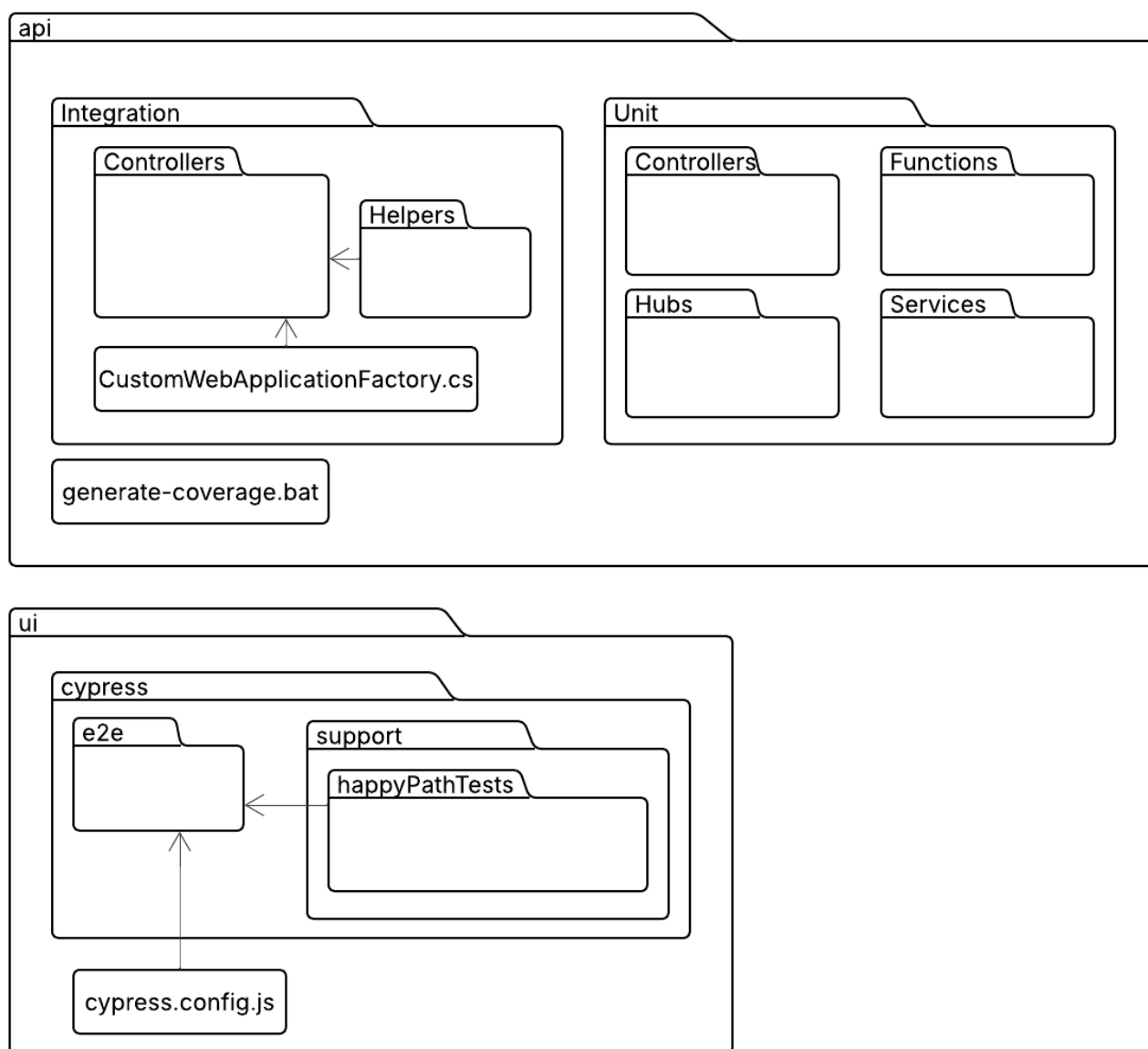
4. fejezet

Fejlesztői dokumentáció - Tesztelés

4.1. Tesztelési stratégia

A projekt megvalósít egység valamint integrációs tesztelést a backend oldalon, valamint egy e2e rendszertesztet a frontend oldalon.

4.1.1. Tesztfájlok fizikai elhelyezkedése (UML)



4.1. ábra. Tesztfájlok és elhelyezkedésük

4.1.2. Teszteléshez használt eszközök

A tesztelést elősegítő keretrendszerek, csomagok listája az alábbi, melyet a fő programban nem használtunk, kizárólag a tesztelésben volt rájuk szükség. Az alábbi listában nem láthatóak azok az eszközök amelyek a fő programban is megjelennek. (3.1.2.-es pont)

- **xUnit** – Az Egység- és integrációs teszteléshez használt tesztelési keretrendszer.

Backend teszteléshez használt eszközök (NuGet csomagok):

- **coverlet.collector** (v6.0.4) – kódfedettség (code coverage) mérésére szolgáló eszköz, amely integrálható a tesztelési pipeline-ba.
- **FluentAssertions** (v8.2.0) – olvashatóbb és kifejezőbb egységteszt assert-ek írását teszi lehetővé.
- **Microsoft.AspNetCore.Mvc.Testing** (v8.0.5) – lehetővé teszi az ASP.NET Core alkalmazások integrációs tesztelését, a WebApplicationFactory osztályon keresztül.
- **Microsoft.EntityFrameworkCore.InMemory** (v9.0.3) – memóriaalapú adatbázis EF Core-hoz, tesztelési célokra.
- **Microsoft.NET.Test.Sdk** (v17.13.0) – az xUnit tesztelési keretrendszer működéséhez szükséges alapsdk, amely a tesztek futtatását támogatja.
- **Moq** (v4.20.72) – mock objektumok létrehozására szolgáló keretrendszer, egységtesztekhez.
- **xunit** (v2.5.3) – xUnit.net tesztelési keretrendszer, a .NET ökoszisztémában elterjedt egységteszt eszköz.
- **xunit.runner.visualstudio** (v2.5.3) – Visual Studio integráció az xUnit tesztek futtatásához és eredményeinek megjelenítéséhez.

Frontend teszteléshez használt eszközök

- **Cypress** – End-to-end (E2E) tesztelési keretrendszer, amely lehetővé teszi a felhasználói interakciók automatizálását a böngészőben. böngészőkben.

4.2. Egységtesztek

A projekt során xUnit keretrendszert használtam az egységtesztek elkészítéséhez. A vezérlők metódusait különböző bemeneti feltételek és edge case-ek mentén teszteltem. A tesztek során InMemory adatbázist használtam az Entity Framework Core támogatásával, valamint a külső függőségek (pl. IJwtHelper, ICopyTicketHelper) Moq segítségével kerültek mockolásra.

4.3. Integrációs tesztek

A rendszer integrációs tesztjeit az xUnit keretrendszer segítségével valósítottam meg, ahol a teljes HTTP-kérések és válaszok tesztelése valós környezethez hasonlóan történik. A CustomWebApplicationFactory osztályon keresztül egy test-specifikus ASP.NET Core alkalmazás példány jön létre, amely InMemory adatbázist és külön konfigurációt használ. Ez lehetővé teszi az autentikáció, a jogosultságkezelés, valamint az adatbázis-kommunikáció teljeskörű vizsgálatát izolált környezetben.

A vezérlők metódusait különböző bemeneti feltételek és edge case-ek mentén teszteltem. A happy path tesztek esetében vizsgáltam az adatbázisra kifejtett hatásukat is.

4.4. Rendszertesztek - automatizált

Az automatizált rendszertesztek a ui mappájában találhatóak. Cypress segítségével szimulálják 2 felhasználó segítségével az összes felhasználói interakciót. A tesztek futásának követelménye az aktív adatbázis megléte valamint a futó backend illetve frontend rendszerek. A tesztelésben beállítható hogy egy, általunk kezelt profillal is megossza a naptárat, ezáltal a signalR működése is ellenőrizhető. A tesztelés során létrehozott profil belépési adatai: email: "CypressTestTester@example.com" jelszó: "Password123".

4.5. Rendszertesztek - Manuális

A tesztelés Given-When-Then struktúrát követve van leírva, melyet a tesztelő manuálisan kell végrehajtson. A táblázatok a happy pathet tartalmazzák, közvetlenül alattuk pedig a kezelt edge casek szerepelnek. A tesztelést ajánlott 2 felhasználóval egy időben végezni, így a releváns funkciók signalR értesítéseinek kezelése is jól látható lesz.

Bejelentkezési oldal funkciói

4.1. táblázat. Regisztráció

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

(4.1.) **Ellenőrizendő hibakezelések:** A felhasználónév üres. Az email cím regisztrált. A jelszavaknak nem egyeznek. A jelszó 8 karakternél rövidebb. A jelszó nem tartalmazza: nagybetű, szám.

Engedélykezelés: -

4.2. táblázat. Bejelentkezés

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

(4.2.) **Ellenőrizendő hibakezelések:** Az emailcímnek nem regisztrált. A jelszó nem megfelelő.

Engedélykezelés: -

Főoldal funkciói

4.3. táblázat. Új feladatlista létrehozása

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

(4.3.) **Ellenőrizendő hibakezelések:** A feladatlista neve üres vagy csak whitespace.

Engedélykezelés: write hozzáférési joggal nem elvégezhető a művelet.

4.4. táblázat. Feladatlista módosítása

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

(4.4.) **Ellenőrizendő hibakezelések:** A feladatlista neve üres vagy csak whitespace.

Engedélykezelés: write hozzáférési joggal nem elvégezhető a művelet.

4.5. táblázat. Feladatlista törlése

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

(4.5.) **Ellenőrizendő hibakezelések:**

Engedélykezelés: write hozzáférési joggal nem elvégezhető a művelet.

4.6. táblázat. Kártya létrehozása az adott feladatlistához

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

(4.6.) **Ellenőrizendő hibakezelések:** A ticket neve üres vagy csak whitespace.

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.7. táblázat. Kártya törlése az adott feladatlistáról

Given	A felhasználó a főoldalon van
When	A kártyát kitörli
Then	A kártya törlődik a feladatlistáról

(4.7.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.8. táblázat. Kártya másolása saját naptárba

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

(4.8.) **Ellenőrizendő hibakezelések:** Nincs kiválasztott naptár.

Engedélykezelés: -

4.9. táblázat. Feladatlista kártyáinak újrendezése

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

(4.9.) **Ellenőrizendő hibakezelések:** A kártya rossz helyre való ejtése nem okoz hibát. (pl.: egy másik listába)

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.10. táblázat. Kártya áthelyezése

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

(4.10.) **Ellenőrizendő hibakezelések:** Időponthoz osztás közben nincs kitöltve mind a 2 input mező. Időponthoz osztás közben az End Time-hoz nem legalább 15 percnél későbbi időpont van írva mint a Start Timehoz.

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.11. táblázat. Naptár napjának megnyitása

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

(4.11.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: -

4.12. táblázat. További naptárak kezelése

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

(4.12.) Ellenőrizendő hibakezelések: -**Engedélykezelés: -**

4.13. táblázat. Naptár másolása saját naptárba

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

(4.13.) Ellenőrizendő hibakezelések: Nincs kiválasztott naptár.**Engedélykezelés: -****Napi nézet funkciói**

4.14. táblázat. Kártya létrehozása

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

(4.14.) Ellenőrizendő hibakezelések: A ticket neve üres vagy csak whitespace. Nincs kiválasztott calendar lista. Csak az egyik time mező van kitöltve. End Time-hoz nem legalább 15 percnél későbbi időpont van írva mint a Start Timehoz. Ha nincs a naptárhoz tartozó calendar lista megjelenik egy hibaüzenet ami arra kér hogy hozz létre egyet.

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.15. táblázat. Kártya törlése

Given	A felhasználó a napi nézet oldalon van
When	A kártyát kitörli
Then	A kártya törlődik

(4.15.) **Ellenőrizendő hibakezelések:**

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.16. táblázat. Todo lista kártyáinak újrendezése

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

(4.16.) **Ellenőrizendő hibakezelések:** A kártya rossz helyre való ejtése nem okoz hibát.

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.17. táblázat. Kártya megjelölése elvégzettként

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

(4.17.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.18. táblázat. Kártya időpontra osztása

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

(4.18.) **Ellenőrizendő hibakezelések:** Időponthoz osztás közben nincs kitöltve mind a 2 input mező. Időponthoz osztás közben az End Time-hoz nem legalább 15 percnél későbbi időpont van írva mint a Start Timehoz.

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.19. táblázat. Kártya visszaküldése a főoldalra

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

(4.19.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: read hozzáférési joggal nem elvégezhető a művelet.

4.20. táblázat. Kártya másolása saját naptárba

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

(4.20.) **Ellenőrizendő hibakezelések:** Nincs kiválasztott naptár.

Engedélykezelés: -

Több naptár kezelése funkciói

4.21. táblázat. Új naptár létrehozása

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

(4.21.) **Ellenőrizendő hibakezelések:** A naptár neve üres vagy csak whitespace.

Engedélykezelés: -

4.22. táblázat. Naptár engedélyeinek kezelése

Given	A felhasználó a több naptár fülön van
When	A naptár manage gombjára kattint
Then	Megnyílik egy modal amely lehetőséget biztosít a naptárhoz kapcsolódó engedélyek kezelésére, bővítésére

(4.22.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: write hozzáférési joggal nem elvégezhető a művelet.

4.23. táblázat. Naptár törlése vagy követésének megszüntetése

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

(4.23.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: write hozzáférési joggal nem elvégezhető a művelet.

4.24. táblázat. Naptár beállítása alapértelmezettnek

Given	A felhasználó a több naptár fülön van
When	A naptár beállítása alapértelmezettnek gombra kattint
Then	A naptár alapértelmezettnek lesz beállítva (azaz loginkor ez töltődik be)

(4.24.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: -

Profil funkciói

4.25. táblázat. Jelszó változtatása

Given	A felhasználó a profil fülön van
When	A jelszó változtatása gombra kattint majd megfelelően paraméterezi a modalt
Then	A jelszava megváltozik

(4.25.) **Ellenőrizendő hibakezelések:** jelszavaknak nem egyeznek. A jelszó 8 karakternél rövidebb . A jelszó nem tartalmazza: nagybetű, szám. A régi jelszó nem megfelelő.

Engedélykezelés: -

4.26. táblázat. Profil törlése

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

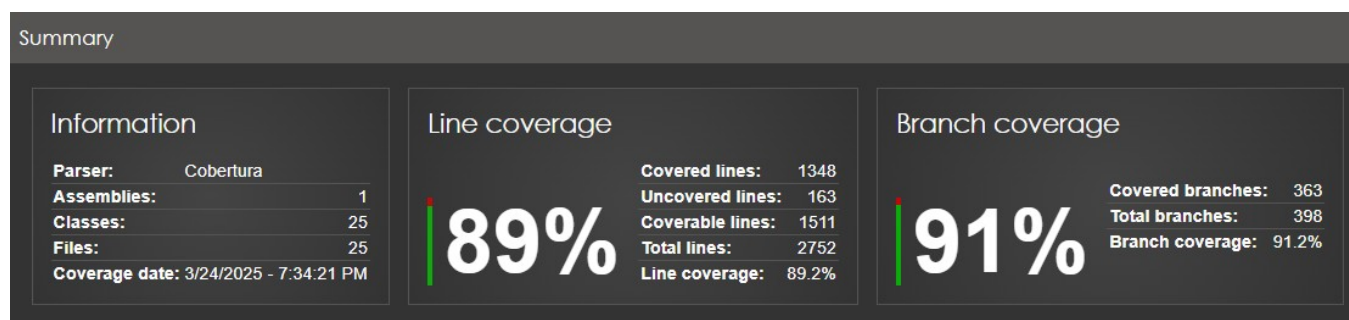
(4.26.) **Ellenőrizendő hibakezelések:** -

Engedélykezelés: -

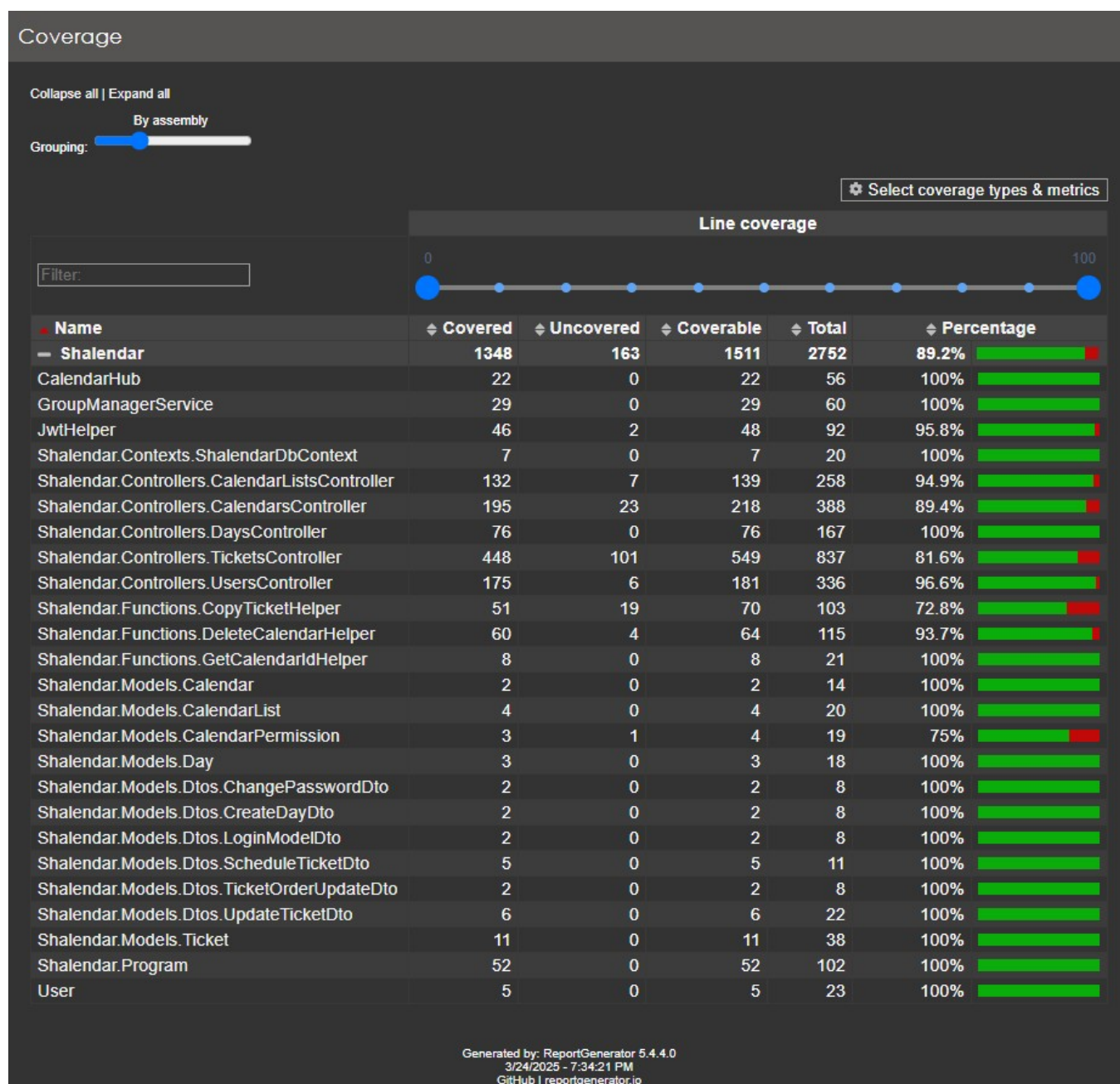
4.6. Tesztelési eredmények

4.6.1. Backend coverage

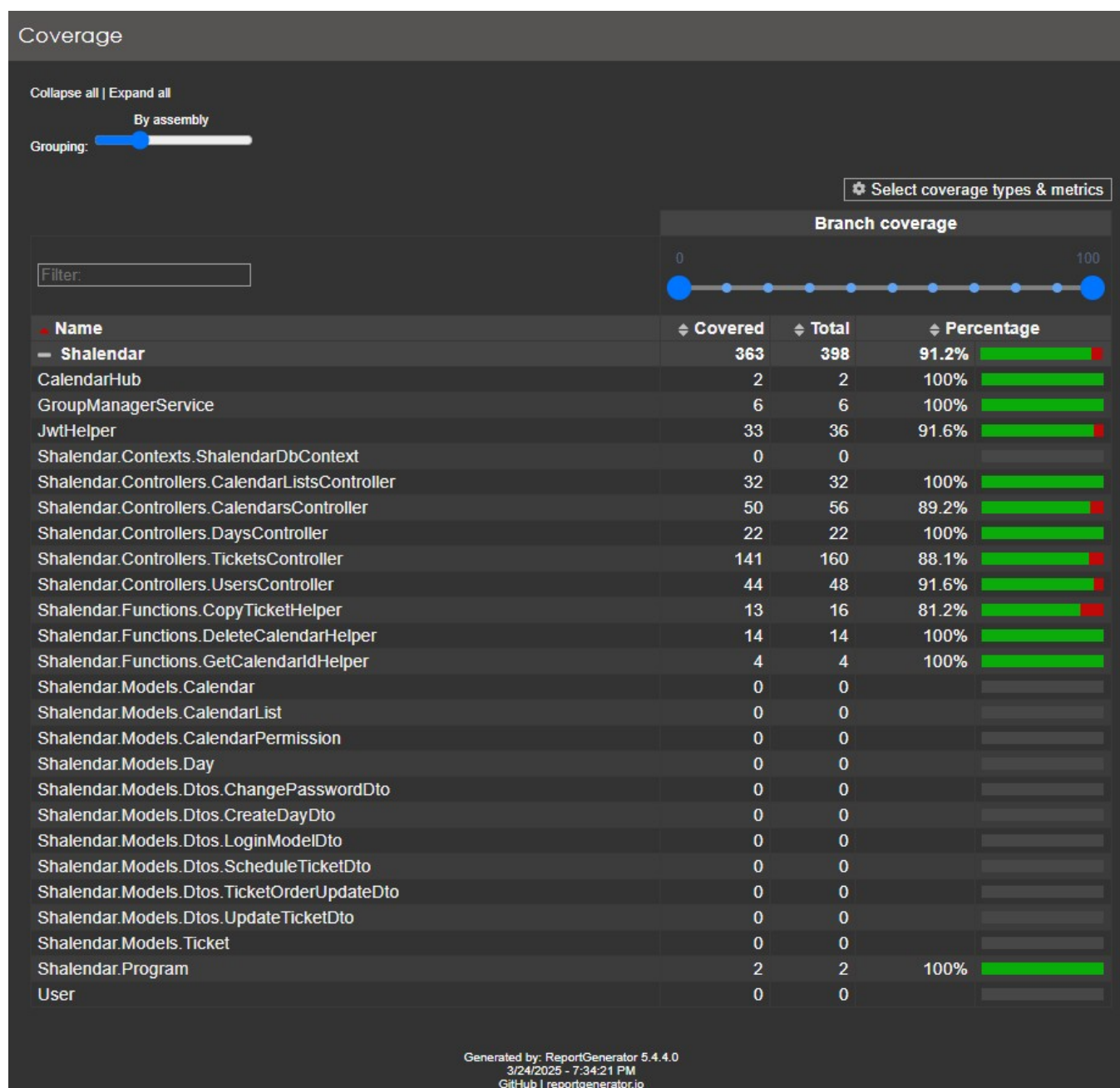
A backend tesztfájlokról coverage report készíthető a generate-coverage.bat futtatásával. Külön report készül az egység valamint integrációs tesztekéről, valamint a 2 egyesítéséről. Az alábbi képen a teljes projekt lefedettsége látszik mind a 2 típusú teszt által. A reportok az alábbi úton érhetőek el: ".../api/Shalendar.Tests/coveragereport"



4.2. ábra. Lefedettségi mutató összefoglaló



4.3. ábra. Részleges lefedettség kimutatás sorokra



4.4. ábra. Részletes lefedettség kimutatás ágakra

A kimutatások sorai tovább kattinthatóak még több részlethez jutva ezáltal.

```
1 68      DateTime selectedDate = parsedDate.Date;
    69
1 70      bool dayExists = await _context.Days
1 71      .AnyAsync(d => d.CalendarId == calendarId && EF.Functions.DateDiffDay(d.Date, selectedDate) == 0);
    72
1 73      if (!dayExists)
1 74      {
1 75          return Ok(new List<object>());
    76      }
    77
0 78      var tickets = await _context.Tickets
0 79      .Where(t => t.CurrentParentType == "ToDoList"
0 80      && t.StartTime == null
```

4.5. ábra. Legrészletesebb kimutatás példa

4.6.2. Frontend e2e teszt megtekintése

[Shalendar e2e teszt videó megtekintése \(Vimeo\)](#)



5. fejezet

Összegzés