



győri szakképzési centrum

Jedlik Ányos
Gépipari és Informatikai
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

Záródolgozat feladatkiírás

Tanuló(k) neve: Szalóki Péter, Zoltai Balázs, Papp Zsombor

Képzés: nappali munkarend

Szak: 5 0613 12 03 Szoftverfejlesztő és -tesztelő technikus

A záródolgozat címe: Pandidakterion

Konzulens: Sándor László

Beadási határidő: 2025. 04. 15.

Győr, 2025. 04. 15.

Módos Gábor
Igazgató

Konzultációs lap

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2025.02.15.	Témaválasztás és specifikáció	
2.	2025.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2025.04.15.	Dokumentáció véglegesítése	

Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2025. április 15.

Tanulók aláírása:

Szalóki Péter Krisztián

Zoltai Balázs

Papp Zsombor



Tartalomjegyzék

1. Bevezetés	7
1.1. A probléma és a megoldása	7
2. Tervezés	7
2.1. Az ötlet	7
2.2. Használt technológiák	8
3. Csapatmunka	8
3.1. Feladatok elosztása	8
3.2. GitHub használata	9
4. Backend	9
4.1. Adatbázis	10
4.1.1. Adatbázis táblái	10
4.1.2. Egyetemi felhasználó tábla (uni_users)	11
4.1.3. Roles tábla	12
4.1.4. Todos tábla	12
4.1.5. Private Calendar tábla	12
4.1.6. Public Calendar tábla	13
4.1.7. News tábla	13
4.1.8. User_validations tábla (QR Code belépéshez szükséges)	13
4.2. Végpontok és Kontrollerek	14
4.2.1. News Controller	14
4.2.2. Public Calendar Controller	17
4.2.3. Personal Calendar Controller	22
4.2.4. Personal Todos Controller	25
4.2.5. Login Controller	28
4.2.6. QR Code segítségével történő belépés	31
4.3. Middleware-ek	32
4.4. Backend Tesztelés	32
4.5. Monitorolás	33
5. Frontend	33
5.1. Bevezetés	33
5.2. Frontend Architektúra és Technológiai Háttér	34



5.2.1.	Angular 19 Standalone Alkalmazás	34
5.2.2.	Főbb Technológiai Döntések	34
5.3.	Alkalmazás Szerkezete és Útválasztás	34
5.4.	AppComponent és Alapvető Szerkezet	35
5.5.	Globális Stílusok és Design Rendszer	35
5.6.	Szolgáltatások (Services)	36
5.6.1.	AuthService	36
5.6.2.	DataService	37
5.7.	Közös Komponensek	37
5.7.1.	Hírek Komponensek	37
5.7.2.	Calendar Komponens	40
5.7.3.	EventCard Komponens	42
5.8.	Main Komponensek	43
5.8.1.	MainComponent - Nyilvános Főoldal	43
5.8.2.	HeaderComponent - Alkalmazás Fejléc	45
5.8.3.	CourseCardList Komponens	46
5.8.4.	CourseCard Komponens	47
5.9.	Login	48
5.10.	Portál komponensek	48
5.10.1.	Portal komponens	48
5.10.2.	PortalHeader komponens	48
5.10.3.	Dashboard komponens	48
5.10.4.	ToDoList komponens	49
5.10.5.	ToDoCard komponens	49
5.10.6.	ToDoForm komponens	49
5.10.7.	NewsForm Komponens	50
5.10.8.	UserControl Komponens	50
5.10.9.	UserCardList Komponens	50
5.10.10.	UserCard Komponens	50
5.10.11.	UserSearch Komponens	50
5.10.12.	UserForm Komponens	51
5.10.13.	UserDetails Komponens	51
5.10.14.	UserEventList Komponens	51
5.10.15.	UserEventCard Komponens	51
5.10.16.	Integráció	51



6. Mobil	52
6.1. Bevezető	52
6.2. Alkalmazás főbb funkciói	53
6.2.1. Biometrikus azonosítás	53
6.2.2. QR kód alapú beléptetés	53
6.2.3. Teendőlista	53
6.2.4. Események kezelése	53
6.3. Technológiai háttér	53
6.4. QR kód beléptető rendszer működése	54
6.4.1. XAML felület	54
6.4.2. C# logika	54
6.5. Teendőlista működése	55
6.5.1. XAML felület	55
6.5.2. C# logika	55
6.6. Hibakezelés és Információközlés	56
6.7. Összegzés	56
7. Üzemeltetés kérdése	56
7.1. Docker Konténerizálás	57
8. Projekt futtatása lokálisan	57
9. Jelenlegi Állapot és a Jövő	58
9.1. Jelenlegi Állapot	58
9.2. Demo adatok bejelentkezéshez	58
9.3. Jövő	58
10.Felhasználói útmutató	59
10.1. Rövid Útmutató a Webes Alkalmazáshoz	59
10.1.1. Elérés és belépés	59
10.1.2. Globális navigáció (HeaderComponent)	59
10.1.3. Nyilvános főoldal (MainComponent)	59
10.1.3.1. Hírek	60
10.1.3.2. Esemény-slider	60
10.1.3.3. Nyilvános naptár-nézet	60
10.1.4. Portál (privát felület)	60
10.1.4.1. Dashboard felépítése	60
10.1.4.2. Teendők (Todos)	60



10.1.4.3. Személyes naptár	61
10.2. Rövid Útmutató a Mobilos Alkalmazáshoz	61
10.2.1. Be- és Kijelentkezés	61
10.2.2. QR beléptetés	61
10.2.3. Teendők kezelése	61
10.2.4. Események kezelése	62
11.Összefoglalás	62
12.Irodalomjegyzék	64

1. Bevezetés

1.1. A probléma és a megoldása

A **Pandidakterion** egy egyetemi web- és mobilalkalmazás, amely az egyetemi oktatók és hallgatók mindennapjait hivatott megkönnyíteni azzal, hogy egységesen elérhetővé tesz minden olyan adminisztrációs funkciót, amely szükséges lehet az egyetem mindennapjaiban. Így tehát az alkalmazás és a weboldal az adminisztratív feladatok egyszerűsítését, az egyetemi események/feladatok követését, és a pozitív felhasználói élmény biztosítását hivatott lehetővé tenni, egy remények szerint mások számára is intuitív módon. Célja, hogy az Egyetem működését tegye gördülékenyebbé/egyszerűbbé, emellett a diákok számára nyújtson egy áttekinthetőbb és egyszerűbb módot arra, hogy naprakészek legyenek.

A program így az alábbi problémákra jelent megoldást:

- **Platformfüggetlen elérhetőség:** Az alkalmazás bármilyen operációs rendszeren futtatható, köszönhetően a mobil applikáció fejlesztésében használt .Net MAUI technológiának. Szükség esetén bármely platformra készíthető belőle futtatható applikáció.
- **Felhasználóbarát kezelhetőség:** Az egyetemi hírek, események és feladatok áttekinthető kezelése egy egyszerű, letisztult, modern felület biztosítja a felhasználó számára.
- **Magas teherbírás és optimalizálás:** A rendszer optimalizált adatbázis-lekérdezésekkel, valamint a későbbiekben kidolgozott üzemeltetési módszerekkel hivatott biztosítani a problémamentes működést nagy terhelés alatt is.

2. Tervezés

2.1. Az ötlet

Az ötlet akkor fogalmazódott meg bennünk, amikor több egyetem weboldalát kellett böngészniük adatok gyűjtése iránt. Megfigyeltük, hogy rengeteg egyetem (köztük külföldiek is) nem fordít kellő energiát egy egységes, stabil rendszer kialakítására és a felhasználói élményre.

Tisztában voltunk vele, hogy a szakmánkban gyakori jelenség, hogy a projekt vagy funkció lefejlesztésére szánt időt drasztikusan alábecsülik a fejlesztők, ezért igyekez-

tünk olyan keretrendszert kialakítani, amely könnyen módosítható új funkciókkal, és fejlesztés közben kiemelten figyel a kód átláthatóságára és tisztaságára.

2.2. Használt technológiák

A használt technológiák kiválasztása során a következő szempontokat vettük figyelembe: ha valakinek elakadás adódna, könnyen segítséget tudnánk kérni egymástól. Végző döntésünk az alábbi megoldások mellett született:

- **Design:** Figma
- **Verziókövetés:** Git, Github
- **Backend:** Laravel
- **Adatbázis:** MySQL
- **Frontend:** Angular
- **Mobil App:** .NET MAUI
- **Üzemeltetés:** AWS EC2

3. Csapatmunka

A hatékony csapatmunka érdekében fontos volt az egyértelmű feladatmegosztás. Tudtuk, hogy mind egymás, mind tanáraink révén tudunk segítséget kérni, ha valamilyen probléma merül fel a fejlesztés során.

3.1. Feladatok elosztása

A projektünk frontendjét Zoltai Balázs valósította meg, a backendet Szalóki Péter, a mobilos alkalmazást pedig Papp Zsombor. Viszont a teljes mértékű három részre bontás lassította volna a fejlesztési fázist, így a munkánk sokszor magában foglalta az egymásnak való besegítést, tehát mindenki dolgozott majdnem minden szegmensén az applikációnak.

A tervezési fázisban és az adatbázis kialakításánál mindnyájan közösen dolgoztunk, mivel ezek a döntések az egész projekt működésére kihatnak. Egyes feladatoknál a *TDD* (Test Driven Development) módszertant alkalmaztuk, ami annyit takar, hogy előbb készítettük el a teszteket egy oldalhoz vagy funkcióhoz, minthogy azok készen lettek volna.

3.2. GitHub használata

A közös munka gördülékeny megvalósításához GitHub-ot használtuk, ami egy olyan felület, amely Git segítségével a weben lehetővé teszi a fejlesztők számára a verziókövetést és a bugok/feladatok aktuális nyomon követését.

Választásunk azért esett a GitHub-ra, mivel a kellő tapasztalattal mindegyikünk rendelkezett ahhoz, hogy problémamentesen használja ezt a technológiát, mivel iskolai környezetben már több éve alkalmazzuk.

4. Backend

A backend fejlesztéséhez a PHP egyik legnépszerűbb, fejlesztőbarát keretrendszerét, a Laravel-t választottuk. Ez egy olyan ökoszisztémát biztosított számunkra, amely segítségével rengeteg olyan funkció implementálható, amelyek más keretrendszerekben nem, vagy csak nehezen, külső függőségek segítségével lennének megvalósítható. Ami biztos, az az, hogy a Laravel csapata elsődleges céljai között tartja számon a teljesítményt, ami számunkra is egy fontos szempont volt, és nagyban befolyásolta a döntésünket a keretrendszer mellett.

Az általunk használt Laravel verzió a jelenleg legújabb, 2025. február 24-én kiadott 12-es verzió, amely nem a drasztikus változások, hanem a Laravel optimalizációs törekvései miatt vált népszerűvé. Személyes tapasztalataink is ezt támasztják alá.

A Laravel számos lehetőséget kínál:

- Az **Artisan CLI**, ami a Laravel beépített parancssori eszköze, segítségével nagyon gyorsan voltunk képesek generálni a szükséges fájlokat ami növelte a fejlesztői hatékonyságunkat is.
- A Laravel által biztosított **Full-Stack fejlesztési lehetőségekkel** nem igazán éltünk, összesen két darab oldal van amelyet a backend szolgál, ezek közül egyiket az API dokumentáció megjelenítésére, a másikat pedig monitorolásra használjuk.
- Az **API dokumentáció** generálásához a Scramble alkalmazását választottuk, mivel Laravel keretrendszerre készült, és a konfigurálása is kifejezetten egyszerű volt számunkra.

Az API dokumentáció elérhető: <http://52.28.154.228:8000/docs/api/>.

4.1. Adatbázis

Elsődleges kérdésként azt vizsgáltuk, mely adatbázis-technológia a legalkalmasabb: Fájl alapú SQLite, MySQL, PostgreSQL vagy MariaDB.

Az adatbázis technológiának végül a MySQL-t választottuk, az alábbi okok miatt:

- Több éves tapasztalatunk van a MySQL-lel, és nem jelent számunkra problémát vele dolgozni.
- A fájl alapú SQLite adatbázist nem lennénk képesek nagyobb terhelés alatt skálázni, és ez könnyen egy bottleneckhez vezetne a programunkban.

Az adatbázis kezdő adatait a Laravel Faker segítségével töltöttük fel, amely nagyban megkönnyítette a munkánkat mind a tesztelés, mind a fejlesztés során, mivel így nem kézzel kellett felvigyük a próbaadatokat.

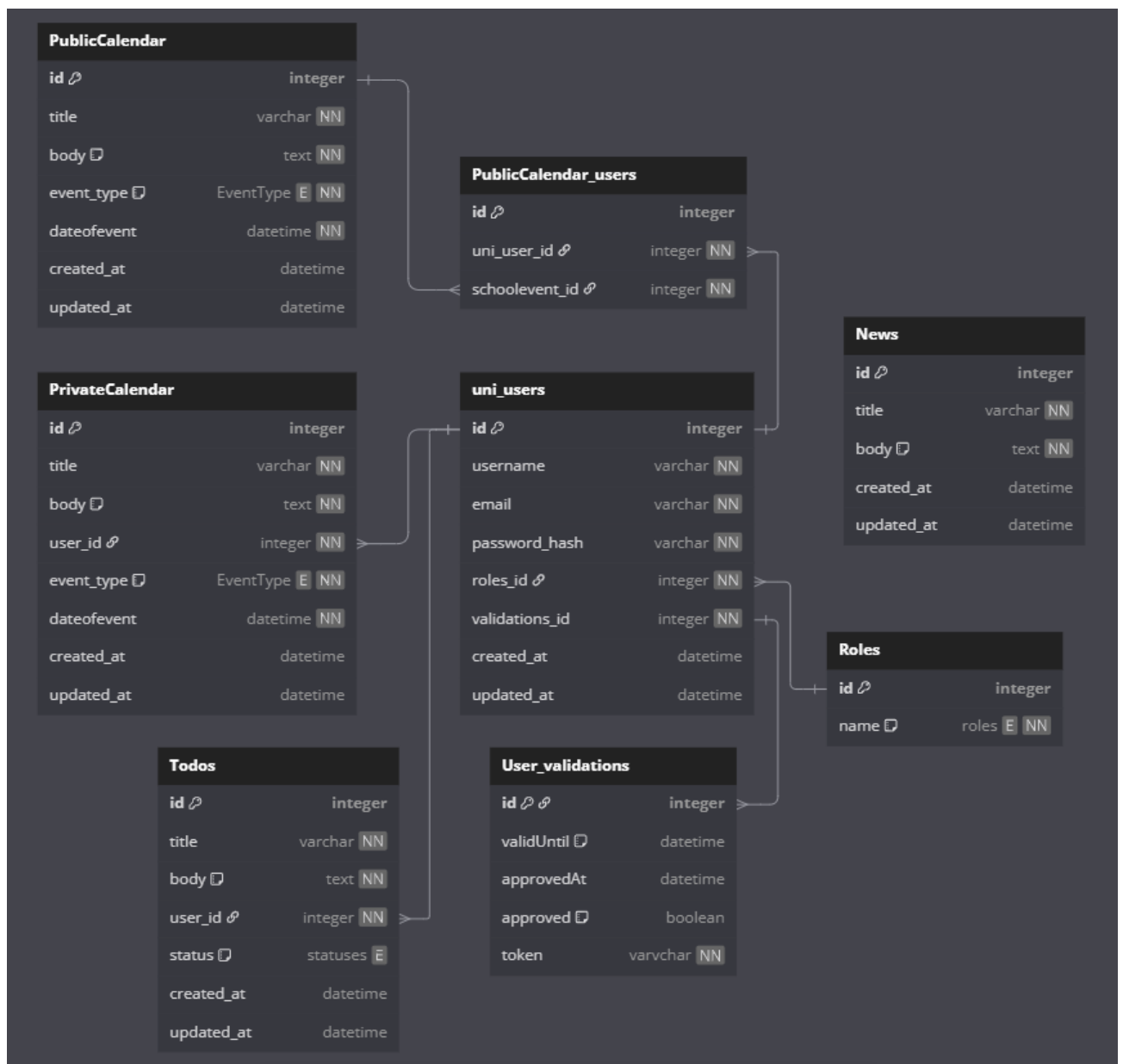
A technológia kiválasztása és a relációs adatbázis megtervezése után első feladatunk az volt, hogy kidolgozzuk a táblákat létrehozó migrációkat, hogy a Laravel képes legyen számunkra automatizálni a táblák létrehozását a fejlesztési folyamat alatt, amennyiben egy új gépen folytattuk munkánkat. Ezután a seederek kidolgozása következett, ehhez a korábban említett Faker osztályt használtuk.

Laravelben két alapértelmezett megközelítés van az adatbázissal történő interakciókra:

- Az **Eloquent ORM** lehetővé teszi rekordok manipulációját SQL lekérdezések írása nélkül.
- A **QueryBuilder** közvetlenebb SQL-szerű megközelítést tesz lehetővé, de mi ezt kevesebbszer alkalmaztuk.

4.1.1. Adatbázis táblái

Az adatbázis vizualizálására/megtervezésére DBDiagramot használtunk.



1. ábra. Adatbázis Diagram

4.1.2. Egyetemi felhasználó tábla (uni_users)

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
username	varchar	egyedi, not null
email	varchar	egyedi, not null
password	varchar	not null

Mező	Típus	Megjegyzés
roles_id	int	idegen kulcs (roles tábla id)
validations_id	int	idegen kulcs (user_validations tábla id)
created_at	date	Laravel által automatikusan kezelve
updated_at	date	Laravel által automatikusan kezelve

4.1.3. Roles tábla

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
name	varchar	not null

4.1.4. Todos tábla

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
title	varchar	not null
body	varchar	not null
status	string	enum, default: todo
uni_user_id	int	idegen kulcs (uni_users tábla id)
created_at	date	Laravel által automatikusan kezelve
updated_at	date	Laravel által automatikusan kezelve

Az status a következő megkötésekkel rendelkezik:

```
"status" => ["required", "string", Rule::in(["todo", "done",  
    "in-progress"])]
```

4.1.5. Private Calendar tábla

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
title	varchar	not null
body	varchar	not null



Mező	Típus	Megjegyzés
uni_user_id	int	idegen kulcs (uni_users tábla id)
event_type	varchar	not null
dateofevent	date	not null
created_at	date	Laravel által automatikusan kezelve
updated_at	date	Laravel által automatikusan kezelve

4.1.6. Public Calendar tábla

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
title	varchar	not null
body	varchar	not null
event_type	string	not null, enum
dateofevent	date	not null
created_at	date	Laravel által automatikusan kezelve
updated_at	date	Laravel által automatikusan kezelve

Az event_type a következő megkötésekkel rendelkezik:

```
"event_type" => ["required", "string", Rule::in(["Student Affairs",
    "Open to Public", "Registration Required", "Family Affairs"])]
```

4.1.7. News tábla

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
title	varchar	not null
body	varchar	not null
created_at	date	Laravel által automatikusan kezelve
updated_at	date	Laravel által automatikusan kezelve

4.1.8. User_validations tábla (QR Code belépéshez szükséges)

Mező	Típus	Megjegyzés
id	int	elsődleges kulcs, auto increment
validUntil	date	default: Jelenlegi dátum + 15 perc
approvedAt	date	nullable
approved	boolean	default: false
token	varchar	not null

Ezen kívül használjuk még a Laravel által biztosított `personal_access_tokens` és `sessions` táblákat.

4.2. Végpontok és Kontrollerek

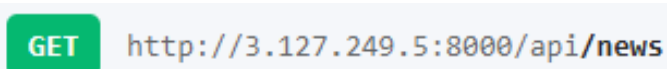
Minden végpontnál a felhasználók a saját naptárukban és feladataikban végezhetnek CRUD műveleteket. A hírek és a közös egyetemi naptár megtekintése nyilvános, módosítási műveletek (DELETE, POST, PUT/PATCH) pedig csak admin vagy teacher jogosultsággal érhetők el.

Végpontok alapértelmezett URL-je: `http://52.28.154.228:8000/api/`

Token nélküli, illetve jogosulatlan kérések esetén a backend 401 vagy 403-as státusz-kóddal válaszol. Amennyiben a felhasználó hiányos adatokat küld, arra a backend a válaszában figyelmezteti (422 Unprocessable Entity).

4.2.1. News Controller

GET News (All)



Példa kérés: `GET | /api/news`

Body: -

Korlátozás: Mindenki számára elérhető

Példa válasz (200 OK):

```
[
  {
    "id": 1,
    "title": "Lorem",
```

```
"body": "Lorem Ipsum",  
"created_at": "1993-03-18T19:42:12.000000Z",  
"updated_at": "2025-03-19T19:55:58.000000Z"  
},  
...  
]
```

GET News (Single)

GET <http://3.127.249.5:8000/api/news/{news}>

Példa kérés: GET | /api/news/2

Body: -

Korlátozás: Mindenki számára elérhető

Példa válasz (200 OK):

```
{  
  "id": 2,  
  "title": "Dolor Set",  
  "body": "Dolor Lorem Ipsum",  
  "created_at": "1995-03-18T19:42:12.000000Z",  
  "updated_at": "2025-02-19T19:55:58.000000Z"  
}
```

POST News

POST <http://3.127.249.5:8000/api/news>

Példa kérés: POST | /api/news

Body:

```
{  
  "title": "Új hír",  
  "body": "Tartalom"  
}
```

Korlátozás: Admin/teacher jogosultság

Példa válasz (201 Created):

```
{  
  "id": 3,  
  "title": "Új hír",  
  "body": "Tartalom",  
  "created_at": "2025-04-20T08:00:00.000000Z",  
  "updated_at": "2025-04-29T21:44:56.000000Z"  
}
```

PUT News

PUT

<http://3.127.249.5:8000/api/news/{news}>

Példa kérés: PUT | /api/news/3

Body:

```
{  
  "title": "Frissített cím",  
  "body": "Frissített tartalom"  
}
```

Korlátozás: admin/teacher jogosultság

Példa válasz (200 OK):

```
{  
  "id": 3,  
  "title": "Frissített cím",  
  "body": "Frissített tartalom",  
  "created_at": "2025-04-20T08:00:00.000000Z",  
  "updated_at": "2025-04-30T21:44:56.000000Z"  
}
```

DELETE News

DELETE

<http://3.127.249.5:8000/api/news/{news}>

Példa kérés: DELETE | /api/news/3

Body: -

Korlátozás: admin/teacher jogosultság

Példa válasz (200 OK):


```
{  
  "message": "News deleted."  
}
```

4.2.2. Public Calendar Controller

GET Public Event (All Events)

GET <http://3.127.249.5:8000/api/uniCalendar>

Példa kérés: GET | /api/uniCalendar

Body: -

Korlátozás: Mindenki számára elérhető

Példa válasz (200 OK):

```
[  
  {  
    "id": 1,  
    "title": "Lorem",  
    "body": "Lorem Ipsum",  
    "event_type": "Open to Public",  
    "dateofevent": "2025-05-02 19:04:54",  
    "created_at": "2025-03-19T19:55:58.000000Z",  
    "updated_at": "2025-03-19T19:55:58.000000Z"  
  },  
  ...  
]
```

GET Public Event (Single Event)

GET <http://3.127.249.5:8000/api/uniCalendar/{uniCalendar}>

Példa kérés: GET | /api/uniCalendar/1

Body: -

Korlátozás: Mindenki számára elérhető

Példa válasz (200 OK):

```
{
  "id": 1,
  "title": "Lorem",
  "body": "Lorem Ipsum",
  "event_type": "Open to Public",
  "dateofevent": "2025-05-02 19:04:54",
  "created_at": "2025-03-19T19:55:58.000000Z",
  "updated_at": "2025-03-19T19:55:58.000000Z"
}
```

POST Public Event

POST <http://3.127.249.5:8000/api/uniCalendar>

Példa kérés: POST | /api/uniCalendar

Body:

```
{
  "title": "Kosárlabda Meccs",
  "body": "Kosárlabda Bajnokság Selejtező Meccse",
  "event_type": "Registration Required",
  "dateofevent": "2025-08-24 14:15:22"
}
```

Korlátozás: admin/teacher jogosultság

Példa válasz (201 Created):

```
{
  "id": 12,
  "title": "Kosárlabda Meccs",
  "body": "Kosárlabda Bajnokság Selejtező Meccse",
  "event_type": "Registration Required",
  "dateofevent": "2025-08-24T14:15:22Z",
  "updated_at": "2025-04-12T22:29:44.000000Z",
  "created_at": "2025-04-12T22:29:44.000000Z"
}
```

PUT Public Event

PUT

<http://3.127.249.5:8000/api/uniCalendar/{uniCalendar}>

Példa kérés: PUT | /api/uniCalendar/2

Body:

```
{
  "title": "Kézilabda Meccs",
  "body": "Kézilabda Bajnokság Selejtező Meccse",
  "event_type": "Registration Required",
  "dateofevent": "2025-08-24 14:15:22"
}
```

Korlátozás: admin/teacher jogosultság

Példa válasz (200 OK):

```
{
  "id": 12,
  "title": "Kézilabda Meccs",
  "body": "Kézilabda Bajnokság Selejtező Meccse",
  "event_type": "Registration Required",
  "dateofevent": "2025-08-24T14:15:22Z",
  "updated_at": "2025-04-12T22:29:44.000000Z",
  "created_at": "2025-04-12T22:29:44.000000Z"
}
```

DELETE Public Event

DELETE

<http://3.127.249.5:8000/api/uniCalendar/{uniCalendar}>

Példa kérés: DELETE | /api/uniCalendar/2

Body: -

Korlátozás: admin/teacher jogosultság

Példa válasz (200 OK):

```
{  
  "message": "Public Event was deleted."  
}
```

POST Event Signup

POST <http://3.127.249.5:8000/api/uniCalendar/{uniCalendar}/signup>

Példa kérés: POST | /api/uniCalendar/1/signup

Body: -

Korlátozás: Bejelentkezett felhasználók

Példa válasz (201 Created):

```
{  
  "message": "Unsubscribed from event."  
}
```

Megjegyzés: Ahogy a fenti példa válaszból szemlélteti, ez a végpont végzi a fel- és leiratkozást is az eseményről.

GET Event Signup

GET <http://52.28.154.228:8000/api/uniCalendar/{uniCalendar}/signup>

Példa kérés: GET | /api/uniCalendar/1/signup

Body: -

Korlátozás: Bejelentkezett felhasználók

Példa válasz (200 OK):

```
{  
  "message": "User is subscribed to event.",  
  "SubStatus": true  
}
```

Megjegyzés: Ahogy a fenti példa válaszból szemlélteti, ezzel a végponttal képesek vagyunk lekérdezni, hogy egy bizonyos eseményre fel van-e iratkozva a felhasználó.

GET All Events With Subscription Status

GET <http://52.28.154.228:8000/api/uniCalendar/withSubs>

Példa kérés: GET | /api/uniCalendar/withSubs

Body: -

Korlátozás: Bejelentkezett felhasználók

Példa válasz (200 OK):

```
[
  {
    "id": 1,
    "title": "Quis earum incidunt eos est nisi.",
    "body": "Ullam sapiente qui expedita. Amet et quo ipsum...",
    "event_type": "Open to Public",
    "dateofevent": "2025-06-11 12:46:51",
    "created_at": "2025-04-13T21:51:46.000000Z",
    "updated_at": "2025-04-13T21:51:46.000000Z",
    "subscribed": true
  },
  ...
]
```

Megjegyzés: Ahogy a fenti példa válaszból szemlélteti, ezzel a végponttal képesek vagyunk lekérdezni az összes eseményt úgy, hogy mellé extra mezőként megjelenik az, hogy fel van-e iratkozva az eseményre.

GET All Subscribed Users

GET <http://52.28.154.228:8000/api/uniCalendar/{uniCalendar}/allUsers>

Példa kérés: GET | /api/uniCalendar/uniCalendar/allUsers

Body: -

Korlátozás: admin jogosultság

Példa válasz (200 OK):

```
{
  "users": [
    {
      "user_id": 2,
```

```
"username": "teacher",
"email": "teacher@teacher.com",
"roles_id": 2,
"created_at": "2025-04-13 21:51:44",
"updated_at": "2025-04-13 21:51:44"
},
...
]
}
```

Megjegyzés: Ahogy a fenti példa válaszból szemlélteti, ezzel a végponttal képes a rendszergazda lekérdezni az összes felhasználót, akik feliratkoztak egy eseményre.

4.2.3. Personal Calendar Controller

GET Personal Events (All)

GET <http://3.127.249.5:8000/api/personalCalendar>

Példa kérés: GET | /api/personalCalendar

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
[
  {
    "id": 5,
    "title": "Reprehenderit atque officiis.",
    "body": "Necessitatibus ea voluptas rerum...",
    "uni_user_id": 2,
    "event_type": "quiz",
    "dateofevent": "2025-05-16T22:40:07Z",
    "created_at": "2025-03-19T19:55:58.000000Z",
    "updated_at": "2025-03-19T19:55:58.000000Z"
  },
  ...
]
```

```
]
```

GET Personal Event (Single)

GET <http://3.127.249.5:8000/api/personalCalendar/{personalCalendar}>

Példa kérés: GET | /api/personalCalendar/5

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{
  "id": 5,
  "title": "Reprehenderit atque officiis ab cupiditate dolorum.",
  "body": "Necessitatibus ea voluptas rerum...",
  "uni_user_id": 2,
  "event_type": "quiz",
  "dateofevent": "2025-05-16T22:40:07Z",
  "created_at": "2025-03-19T19:55:58.000000Z",
  "updated_at": "2025-03-19T19:55:58.000000Z"
}
```

POST Personal Event

POST <http://3.127.249.5:8000/api/personalCalendar>

Példa kérés: POST | /api/personalCalendar

Body:

```
{
  "title": "Vizsgára készülés",
  "body": "Analízis II. gyakorlás",
  "event_type": "study",
  "dateofevent": "2025-06-10T15:00:00Z"
}
```

Korlátozás: Bejelentkezett felhasználó

Példa válasz (201 Created):

```
{
  "id": 6,
  "title": "Vizsgára készülés",
  "body": "Analízis II. gyakorlás",
  "uni_user_id": 2,
  "event_type": "study",
  "dateofevent": "2025-06-10T15:00:00Z",
  "created_at": "2025-04-15T08:30:00.000000Z",
  "updated_at": "2025-04-15T08:30:00.000000Z"
}
```

PUT Personal Event

PUT

<http://3.127.249.5:8000/api/personalCalendar/{personalCalendar}>

Példa kérés: PUT | /api/personalCalendar/6

Body:

```
{
  "title": "Analízis II. - Gyakorló feladatok",
  "body": "Analízis II. gyakorlás",
  "uni_user_id": 2,
  "event_type": "study",
  "dateofevent": "2025-06-10T15:00:00Z"
}
```

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{
  "id": 6,
  "title": "Analízis II. - Gyakorló feladatok",
  "body": "Analízis II. gyakorlás",
  "uni_user_id": 2,
  "event_type": "study",
}
```



```
"dateofevent": "2025-06-10T15:00:00Z",  
"created_at": "2025-04-15T09:45:00.000000Z",  
"updated_at": "2025-04-15T09:45:00.000000Z"  
}
```

DELETE Personal Event

DELETE <http://3.127.249.5:8000/api/personalCalendar/{personalCalendar}>

Példa kérés: DELETE | /api/personalCalendar/6

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{  
  "message": "Event was deleted."  
}
```

4.2.4. Personal Todos Controller

GET Personal Todos (All)

GET <http://3.127.249.5:8000/api/personalTodos>

Példa kérés: GET | /api/personalTodos

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
[  
  {  
    "id": 9,  
    "title": "Non consequatur est deleniti earum aut.",  
    "body": "Dolorem molestiae nemo voluptatem...",  
    "status": "in-progress",  
    "uni_user_id": 3,  
    "created_at": "2025-03-19T19:55:58.000000Z",  
  }  
]
```



```
      "updated_at": "2025-03-19T19:55:58.000000Z"  
    },  
    ...  
  ]
```

GET Single Todo

GET <http://3.127.249.5:8000/api/personalTodos/{personalTodo}>

Példa kérés: GET | /api/personalTodos/9

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{  
  "id": 9,  
  "title": "Non consequatur est deleniti earum aut.",  
  "body": "Dolorem molestiae nemo voluptatem...",  
  "status": "in-progress",  
  "uni_user_id": 3,  
  "created_at": "2025-03-19T19:55:58.000000Z",  
  "updated_at": "2025-03-19T19:55:58.000000Z"  
}
```

POST Create Todo

POST <http://3.127.249.5:8000/api/personalTodos>

Példa kérés: POST | /api/personalTodos

Body:

```
{  
  "title": "Házifeladat beadása",  
  "body": "Analízis II. HF megoldás"  
}
```

Korlátozás: Bejelentkezett felhasználó

Példa válasz (201 Created):

```
{  
  "id": 10,  
  "title": "Házifeladat beadása",  
  "body": "Analízis II. HF megoldás",  
  "status": "todo",  
  "uni_user_id": 3,  
  "created_at": "2025-04-20T08:00:00.000000Z"  
}
```

PUT Update Todo

PUT <http://3.127.249.5:8000/api/personalTodos/{personalTodo}>

Példa kérés: PUT | /api/personalTodos/10

Body:

```
{  
  "title": "Házifeladat beadása",  
  "body": "Analízis II. HF megoldás",  
  "status": "done"  
}
```

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{  
  "id": 10,  
  "title": "Házifeladat beadása",  
  "body": "Analízis II. HF megoldás",  
  "status": "done",  
  "uni_user_id": 3,  
  "created_at": "2025-04-20T08:00:00.000000Z",  
  "updated_at": "2025-04-20T09:30:00.000000Z"  
}
```

DELETE Todo

DELETE <http://3.127.249.5:8000/api/personalTodos/{personalTodo}>

Példa kérés: DELETE | /api/personalTodos/10

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{
  "message": "Todo deleted successfully."
}
```

4.2.5. Login Controller

Login User

POST http://3.127.249.5:8000/api/login

Példa kérés: POST | /api/login

Body:

```
{
  "email": "admin@admin.com",
  "password": "admin"
}
```

Korlátozás: -

Példa válasz (200 OK):

```
{
  "message": "Login successful",
  "token": "1|nsyg3jovAo5qHFTtufpL08i10IyUFqQfXTu2WrG19552468f",
  "user": {
    "id": 1,
    "username": "admin",
    "email": "admin@admin.com",
    "roles_id": 1,
    "created_at": "2025-03-19T19:55:56.000000Z",
    "updated_at": "2025-04-08T11:53:53.000000Z"
  }
}
```

```
}
```

Logout User

POST <http://3.127.249.5:8000/api/logout>

Példa kérés: POST | /api/logout

Body: -

Korlátozás: Bejelentkezett felhasználó

Példa válasz (200 OK):

```
{
  "message": "Logout successful"
}
```

Edit User

PUT <http://3.127.249.5:8000/api/users/{user}>

Példa kérés: PUT | /api/users/1

Body:

```
{
  "username": "string",
  "password": "string",
  "password_confirmation": "string"
}
```

Korlátozás: Bejelentkezett felhasználó képes módosítani saját maga adatait (ha más felhasználót próbál módosítani akkor 403-as státuszkóddal tér vissza a backend), illetve Admin jogosultsággal rendelkező képes módosítani bármelyik felhasználót.

Példa válasz (200 OK):

```
{
  "message": "Update successful",
  "token": null, // token null, ha admin módosítja a felhasználót
  "user": {
    "id": 4,
```



```
{
  "username": "string",
  "email": "dane69@example.com",
  "roles_id": 2,
  "created_at": "2025-03-19T19:55:58.000000Z",
  "updated_at": "2025-04-13T10:30:36.000000Z"
}
```

Register User

POST

<http://3.127.249.5:8000/api/admin/register>

Példa kérés: POST | /api/admin/register

Body:

```
{
  "username": "example",
  "email": "example@example.com",
  "password": "example"
}
```

Korlátozás: admin jogosultság

Példa válasz (201 CREATED):

```
{
  "message": "Registration successful",
  "user": {
    "id": 9,
    "username": "example",
    "email": "example@example.com",
    "roles_id": 3,
    "updated_at": "2025-04-13T10:36:08.000000Z",
    "created_at": "2025-04-13T10:36:08.000000Z"
  }
}
```

4.2.6. QR Code segítségével történő belépés

Generate QRCode token

POST <http://3.127.249.5:8000/api/qrcode/generate>

Példa kérés: POST | /api/qrcode/generate

Body: -

Korlátozás: -

Példa válasz (200 OK):

```
{
  "qrcode": "q0EYL3xVJ31gXGFFcSXoY6kx98hhfyle"
}
```

Login with QRCode token

POST <http://3.127.249.5:8000/api/qrcode/login>

Példa kérés: POST | /api/qrcode/login

Body:

```
{
  "email": "user@example.com",
  "password": "string",
  "token": "q0EYL3xVJ31gXGFFcSXoY6kx98hhfyle"
}
```

Korlátozás: Bejelentkezett felhasználó (mobil alkalmazásban)

Példa válasz (200 OK):

```
{
  "status": "Success"
}
```

Megjegyzés: A mobilos applikáció által küldött kérés emailt, jelszót és a generált QR kód tokenet tartalmazza. Sikeres validáció esetén a backend Laravel Verbb segítségével üzenetet küld a frontendnek, amely tartalmazza a tokenet, ahogy azt egy átlagos login is tenné.

4.3. Middleware-ek

A backend fejlesztéséhez/működéséhez Laravel Sanctum által biztosított funkciókat és middleware-eket is használtuk a gördülékenyebb működés érdekében. A Laravel Sanctum biztosít egy olyan token rendszert és middleware-t, amellyel képesek vagyunk route-ok elérését korlátozni, és felhasználói jogosultságok mögé zárni. A projektünkben alkalmazott middleware-ek elintézik például a felhasználó tokenjének meghosszabbítását is. A token a legutóbbi használattól számított 15 percig érvényes, ezen időintervallum után inaktívnak számít a token.

4.4. Backend Tesztelés

Backend teszteléséhez PHPUnit tesztelési keretrendszert alkalmaztuk, mivel a Pest keretrendszerrel szemben számunkra ez szimpatikusabb volt. A backend tesztelése során sikeresen leteszteltük majdnem minden szegmensét azon céllal, hogy a publikálás előtt és a fejlesztés során megelőzzük a hibás kód publikálását.

Tesztelés során, hogy az adatbázisban változásokat ne tegyünk, egy ideiglenes memóriaalapú adatbázist használunk a tesztek végrehajtására. (Ezen megoldást az E2E tesztek során is alkalmazzuk)

Összesen a backenden van 54 tesztünk és 155 assertion-ünk.


```
✓ create public event endpoint works for admin
✓ update public event endpoint works for admin
✓ delete public event endpoint works for admin
✓ user can sub to public event
✓ user can unsub from public event
✓ user cant create public event
✓ user cant update public event
✓ user cant delete public event

PASS Tests\Feature\TodoTest
✓ get all todos of user
✓ get a todo of user
✓ cant get a todo of different user as not admin
✓ get all todos of different user as admin
✓ get a todo of different user as admin
✓ cant edit a todo of different user even as admin
✓ cant delete a todo of a different user even as admin
✓ user create a todo for himself successfully
✓ user edit a todo for himself successfully
✓ user delete a todo for himself successfully

PASS Tests\Feature\UserTest
✓ login successfull with correct credentials
✓ login fails with incorrect credentials
✓ logout successfull as user
✓ register works for admin
✓ register gives 422 when not all required fields are provided
✓ register unauthorized for students
✓ register unauthorized for someone that isnt logged in
✓ edit user works for admin
✓ edit user works for user aka the user can edit himself
✓ edit user gives 422 when not all required fields are provided
✓ edit user gives 422 for admin when not all required fields are provided
✓ edit user unauthorized for someone that isnt logged in
✓ qr code generation works

Tests: 54 passed (155 assertions)
Duration: 4.06s
```

2. ábra. Backend Tesztek

4.5. Monitorolás

Laravel Pulse segítségével képesek vagyunk számon tartani a backendre érkező kéréseket, és hogy azok mekkora terhelést okoznak. A teherbírás teszteléséhez artillery.io-t használtuk, és ahogy összevetettük a programunkat egy lokálisan futó fájl alapú adatbázissal rendelkező verziójához, a projektünkön majdnem négyszeres teljesítményjavulást tapasztaltunk.

5. Frontend

5.1. Bevezetés

A Pandidakterion projekt keretében kifejlesztett Angular frontend alkalmazás egy modern, felhasználóbarát webes felületet biztosít az egyetemi élet számos aspektusának kezeléséhez. Az Angular 19-es verziójára épülő *standalone single page application* (SPA) két fő nézetből áll: egy nyilvános főoldalból (*main*) és egy hitelesítést igénylő privát felhasználói felületből (*portal*), melyeket egy bejelentkezési rendszer köt össze. A webes applikáció elérhető ezen a linken keresztül: <http://52.28.154.228/>

5.2. Frontend Architektúra és Technológiai Háttér

5.2.1. Angular 19 Standalone Alkalmazás

Az alkalmazás az Angular jelenleg legújabb, 19-es verzióját használja *standalone* komponensekkel, ami lehetővé teszi a modulok nélküli, könnyebb súlyú fejlesztést.

5.2.2. Főbb Technológiai Döntések

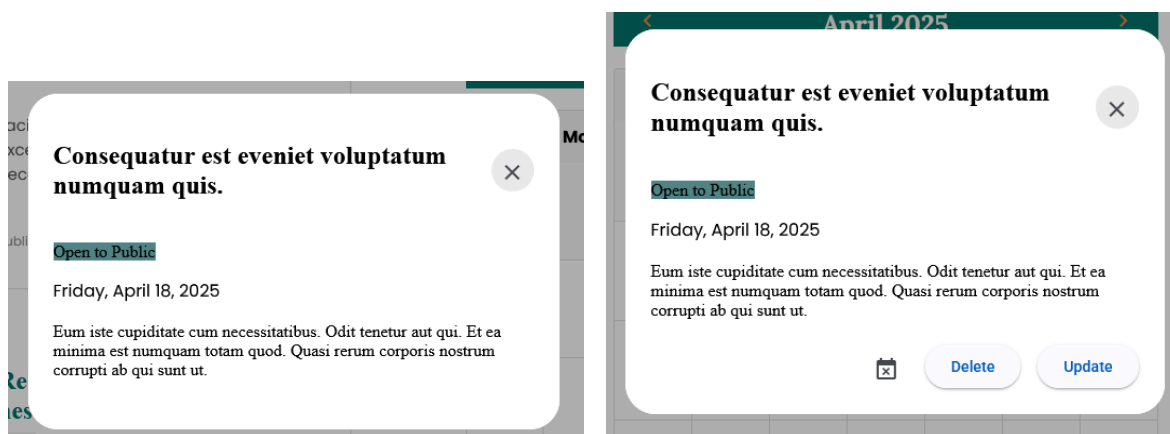
- **TypeScript**
- **Angular Material:** A Google Material Design implementációja Angular keretrendszerben, amely előre definiált, reszponzív UI komponenseket biztosít.
- **Swiper JS:** Modern, érintésre optimalizált *slider* könyvtár, amelyet a főoldali tartalmak prezentálására használtunk.
- **Jest:** Teljesítménycentrikus tesztelési keretrendszer egység- és integrációs tesztekhez.

5.3. Alkalmazás Szerkezete és Útválasztás

Az Angular alkalmazásunk egy egyszerű, de hatékony szerkezetet követ, amely két fő nézetből áll:

- **MainComponent:** A nyilvános főoldal, amely minden felhasználó számára elérhető
- **PortalComponent:** A hitelesítést igénylő felhasználói portál

A részeket felépítő gyerek komponensek gyakran megosztottak a két nézet között. Ezen esetben jellemzően a portal nézet bővítés a main nézetbeli implementációnak. Az alábbi ábrákon látható például miként bővülnek az interakciós lehetőségek az event-modal komponens esetén. Még a publikus weboldalon a komponens pusztán prezentatív jellegű, a portálon elérhetővé válnak a fel-, illetve leiratkozási, továbbá admin és teacher jogosultságú felhasználók számára az edit és delete funkciók is.



(a) event-modal a főoldalon

(b) event-modal a portál oldalon

3. ábra. Side-by-side összehasonlítása az event-modal nézetnek.

Az útválasztást az `app.routes.ts` fájlban definiáltuk:

```
10 export const routes: Routes = [
11   { path: '', component: MainComponent },
12   { path: 'login', component: LoginComponent },
13   {
14     path: 'portal',
15     component: PortalComponent,
16     children: [
17       { path: 'dashboard', component: DashboardComponent },
18       { path: 'todos', component: TodoListComponent },
19       { path: 'news', component: PortalNewsComponent },
20       { path: 'users', component: UserControlComponent },
21       { path: '', redirectTo: 'dashboard', pathMatch: 'full' }
22     ]
23   }
24 ];
25
```

4. ábra. Routes - `app.routes.ts`

5.4. AppComponent és Alapvető Szerkezet

Az alkalmazás belépési pontja az `AppComponent`. A komponens sablonja csupán egy `<router-outlet>` elemet tartalmaz, amely az aktuális útvonalhoz tartozó komponenszt jeleníti meg, a fenti (4. Routes) ábrán látható módon.

5.5. Globális Stílusok és Design Rendszer

A `styles.css` fájlban definiáltunk egy átfogó design rendszert CSS változókkal:

```
:root {
```



```
—primary-color: #66023c;  
—primary-color-rgb: 102, 2, 60;  
—secondary-color: #fff4e9;  
—secondary-color-rgb: 255, 244, 233;  
—primary-action-color: #e38946;  
—primary-action-color-rgb: 227 137 70;  
—secondary-action-color: #007269;  
  
—font-size-small: 12px;  
—font-size-medium: 15px;  
—font-size-semi-large: 19px;  
—font-size-large: 24px;  
—font-size-xl: 31px;  
}
```

A stíluslap tartalmaz még továbbá:

- Reszponzív tipográfiát
- Gombstílusokat (`.btn`, `.btn-primary`)
- Dialógusablakok stílusait
- Media query-eket különböző képernyőméretekhez

5.6. Szolgáltatások (Services)

Az alkalmazás funkcionalitását különböző szolgáltatások (services) biztosítják, amelyeket dependency injection segítségével használunk a komponensekben.

5.6.1. AuthService

A `AuthService` felelős a hitelesítési logikáért:

- Bejelentkezés/kilépés kezelése (tárolás, lekérdezés)
- Felhasználói adatok kezelése
- QR kód alapú hitelesítés támogatása
- Jogosultságkezelés (admin/teacher ellenőrzés)

5.6.2. DataService

A DataService központi pontja az alkalmazás adatkezelésének, amely a következő funkciókat biztosítja:

- Hírek kezelése (CRUD műveletek)
- Naptáresemények kezelése (személyes és nyilvános)
- Teendők kezelése
- Felhasználókezelés (keresés, szerkesztés, létrehozás)
- Eseményjelentkezések kezelése

5.7. Közös Komponensek

5.7.1. Hírek Komponensek

A hírek megjelenítésére és kezelésére több összetett komponensből álló rendszert fejlesztettünk ki.

NewsComponent - Fő Hírmenedzser A NewsComponent koordinálja a hírek kezelését:

- **Funkciók:**
 - Hírek betöltése, lapozás támogatása
 - Részletes hír megjelenítése
 - Szerkesztési és törlési műveletek kezelése
- **Főbb metódusok:**
 - `loadInitialNews()`
 - `setupNewsUpdates()`
 - `onEdit(eventData)`
 - `onDelete(eventData)`
 - `onPageChange(newPage)`
 - `onCardClick(news)`

```
41 loadInitialNews() {  
42   this.dataService.getNews().subscribe({  
43     next: (response: News[]) => {  
44       this.newsService.updateNewsList(response);  
45     },  
46     error: (error: any) => {  
47       console.error(error);  
48     },  
49   });  
50 }
```

5. ábra. Példa a news.component.ts egy függvényére: loadInitialNews()

NewsGridComponent - Hírek rácsos megjelenítése A NewsGridComponent a hírek listás/rácsos megjelenítéséért felelős:

- Hírek kártyás megjelenítése
- Lapozás kezelése
- Események továbbítása a szülő komponensnek

News

Ut quos alias delectus excepturi ut dolore none.

Nesciunt impedit autem a ipsa
expedita. Quam amet praesentium
distinctio out numquam blanditiis. Aut...

Published: May 7, 2017

A a deleniti voluptate similique animi non blanditiis magni.

Quos qui animi molestiae dolores.
Deserunt ducimus dolorem reiciendis
ullam veritatis sit amet et. Facilis...

Published: June 28, 1996

Nemo rerum a dolor voluptatem quae eaque at.

Facilis sed dolor fuga eos sunt error ad
excepturi. Excepturi a qui qui magnam
necessitatibus quis voluptatem. Sunt...

Published: February 8, 2013

Rerum et aperiam vitae nesciunt ipsum libero neque.

Saepe sequi reprehenderit asperiores
et omnis. Quas qui perferendis ullam in
ea sit facere. Suscipit est qui illum...

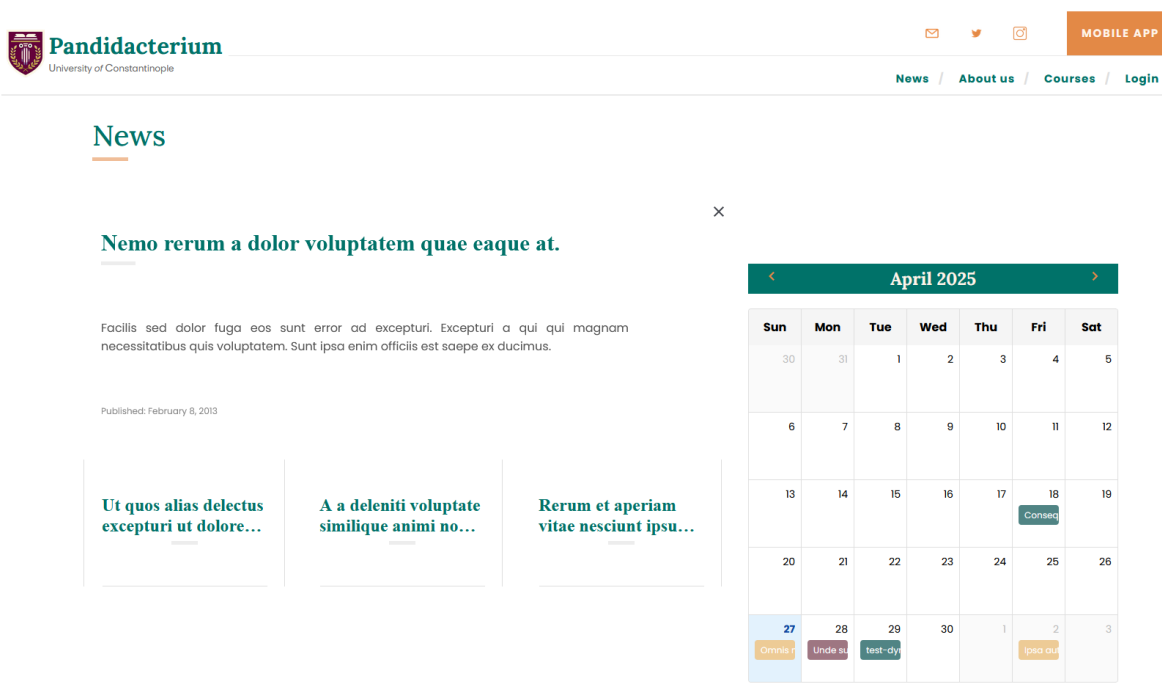
Published: June 13, 2008

April 2025						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18 Conseq	19
20	21	22	23	24	25	26
27 Omnis	28 Unde su	29 test-dy	30	1	2 Ipsa aut	3

6. ábra. Balra a news-grid, felette a header, jobbra a calendar komponensek

NewsFocusComponent - Részletes Hírnézet A NewsFocusComponent a kiemelt hír és a további előnézeti hírek megjelenítését biztosítja:

- Aktív hír kiemelése
- További hírek előnézetének megjelenítése
- Navigáció hírek között



7. ábra. Balra a news-focus, felette a header, jobbra a calendar komponensek

NewsCardComponent - Hírkártya A NewsCardComponent az egyes hírek kompakt megjelenítésére szolgál:

- Hír adatainak megjelenítése
- Szerkesztési és törlési lehetőségek admin/teacher szerepkörben a portálon
- Interaktív viselkedés (hover, kattintás)

**Ut quos alias delectus
excepturi ut dolore none.**

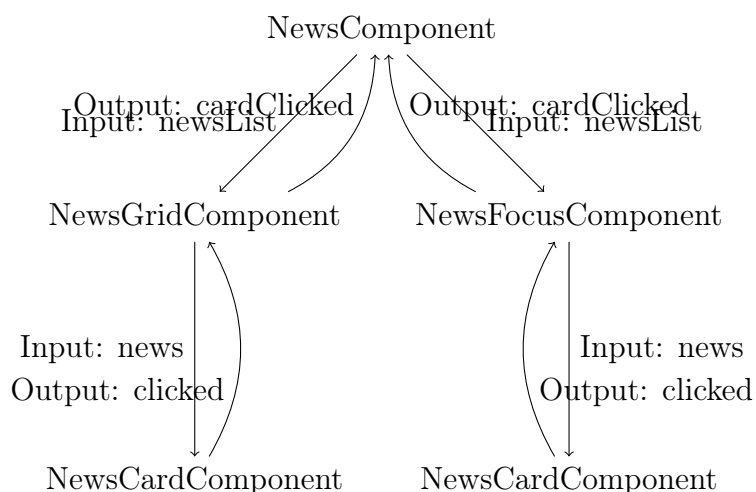
Nesciunt impedit autem a ipsa
expedita. Quam amet praesentium
distinctio aut numquam blanditiis. Aut...

Published: May 7, 2017



8. ábra. news-card komponens a portál oldalon edit / delete opciókkal

Komponensek Közötti Kommunikáció A komponensek közötti adat- és eseményáramlás:



9. ábra. A hírkomponensek hierarchiája és kommunikációja

5.7.2. Calendar Komponens

A Calendar komponens naptár megjelenítést valósít meg, események listázásával és részletes nézettel.

Funkcionalitás

- Havi navigáció
- Események színkódolt megjelenítése



- Részletes eseménynézet modal ablakban

EventDetailModal komponens Speciális viselkedések:

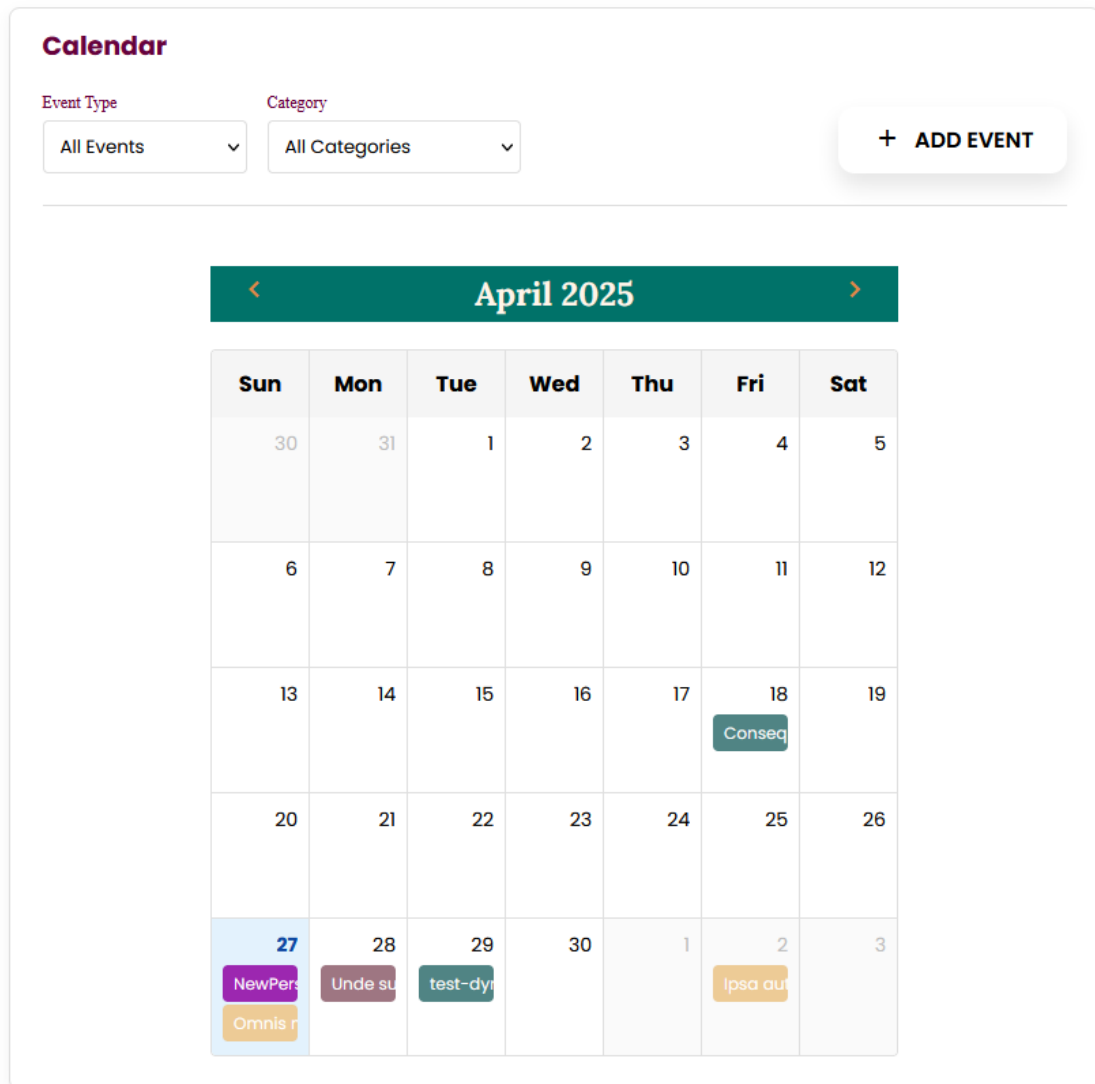
- Portál-specifikus gombok (szerkesztés, törlés, jelentkezés)
- Jogosultságkezelés adminok, tanárok, diákok számára
- Reszponzív megjelenés mobilnézetben

Adatkötések

- `eventsMap`, `loading`, `error`
- `isInPortal` állapot kezelése

Komponens-kommunikáció

- `eventAction` - események frissítése/törlése
- `subscriptionChange` - jelentkezési állapot változás



10. ábra. calendar komponens portál nézetben

5.7.3. EventCard Komponens

Az EventCard egy kompakt, vizuálisan gazdag eseménymegjelenítő kártya a főoldalon és portálon.

Technikai részletek

- CSS változók alkalmazása
- Gradiens háttér (`::before`), elválasztó (`::after`)
- Háttér transzformáció (`transform: scaleY(-1)`)

Felhasználás Az `upcomingEvents` tömb eseményeinek megjelenítésére szolgál:

```
<app-event-card *ngFor="let event of upcomingEvents" [event]="event">
```

A komponens tisztán prezentációs célokat szolgál, nem tartalmaz üzleti logikát.

5.8. Main Komponensek

5.8.1. MainComponent - Nyilvános Főoldal

A `MainComponent` az alkalmazás nyilvános főoldalát reprezentálja, amely minden felhasználó számára elérhető, függetlenül a hitelesítési állapottól. Ez a komponens szolgál keretként a főoldal különböző szekcióinak, és koordinálja az alárendelt komponensek működését.

Szerkezet és Layout A komponens HTML sablonja (`main.component.html`) egy hierarchikus elrendezést követ:

```
1 <div class="view--wrapper view__main--wrapper">
2   <app-header></app-header>
3   <app-slide-card-list></app-slide-card-list>
```

11. ábra. Példa a `main.component.html` template felépítésére

A főbb struktúrális elemek:

- `app-header`: Az oldal fejléc komponense
- `app-slide-card-list`: Főoldali slider komponens, amely egyetemi cikkekhez, kutatásokhoz linkelne egy valós környezetben
- `main`: A fő tartalmi terület, amelyben a szekciók helyezkednek el

Stíluskezelés A komponens stílusait a `main.component.css` fájl tartalmazza, amely a következőket definiálja:

- Reszponzív elrendezést különböző képernyőméretekhez
- Szekciók közötti elválasztókat és színátmeneteket
- Kártya alapú komponensek stílusait
- Media query-ket a különböző eszközök támogatásához

Funkcionalitás és Adatkezelés A MainComponent TypeScript kódja (`main.component.ts`) a következő főbb feladatokat látja el:

- Nyilvános események betöltése és kezelése
- Közelgő események listájának előkészítése
- Adatok formázása a megjelenítéshez
- Változásérzékelés kezelése

A komponens főbb metódusai:

- `loadPublicEvents()`: A `data-service.ts` `getPublicEvents()` metódusával lekéri a publikus naptári eseményeket, és frissíti a `publicEventsMap` tömböt.
- `updateEventsMap()`: A `Map<string, CalendarEvent[]>` struktúra frissítése a portal nézethez, elkerülve a felesleges újra-fetch-eket.
- `normalizeDate()`: Események dátumformátumának szabványosítása (yyyy-MM-dd).
- `loadUpcomingEvents()`: Az aktuális dátumhoz legközelebbi 3 publikus esemény lekérdezése az `upcomingEvents` tömbbe.

```
44  loadPublicEvents() {
45      this.loading = true;
46      this.error = null;
47
48      this.dataService.getPublicEvents().subscribe({
49          next: (events) => {
50              this.updateEventsMap(events);
51              this.loading = false;
52              this.cdr.detectChanges();
53          },
54          error: (err) => {
55              console.error('Error loading events:', err);
56              this.error = 'Failed to load events';
57              this.loading = false;
58              this.cdr.detectChanges();
59          },
60      });
61  }
```

12. ábra. Példa a `main.component.ts` egy függvényére: `loadPublicEvents()`

Szekciók és Alkomponensek A főoldal a következő főbb szekciókból áll:

- **News / Hírek:** Az egyetem legfrissebb hírei egy lapozható naptárban.
- **Upcoming Events / Közelgő események:** Az aktuális dátumhoz közeli három esemény kiemelése.
- **Featured Cards / Kiemelt tartalmak:** Kiemelt kártyák ösztöndíj programok, egyéb információk számára (jelenleg lorem ipsum szöveg).
- **About Us / Rólunk:** Az egyetem bemutatása.
- **Courses / Kurzusok:** Elérhető tanfolyamok listája egy Swiper carousel segítségével.

Az "About Us" és "Featured Card" szekciókat leszámítva minden szekció saját komponenssel került megvalósításra. A `MainComponent` feladata ezek koordinálása és az adatok biztosítása.

5.8.2. HeaderComponent - Alkalmazás Fejléc

A `HeaderComponent` az alkalmazás globális fejlécét valósítja meg, amely minden nézetben megjelenik és kulcsfontosságú navigációs funkciókat biztosít. A komponens reszponzív kialakítása lehetővé teszi az optimális megjelenést minden eszközön.

Szerkezet és Layout A fejléc HTML sablonja (`header.component.html`) két fő részből áll:

- **Logó és menügomb:** Az oldal tetején helyezkedik el
- **Navigációs menük:** Főmenü és másodlagos menü (közösségi média ikonok)

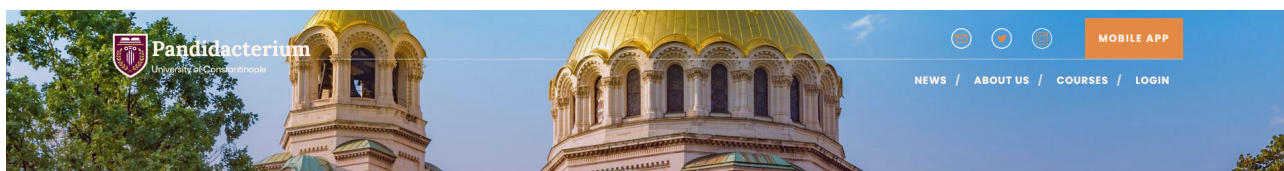
Funkcionalitás A `HeaderComponent` TypeScript kódja (`header.component.ts`) a következő főbb funkciókat implementálja:

- **Scroll viselkedés:** A fejléc megjelenésének változtatása scrollozáskor
- **Reszponzív menü:** Mobilnézetben összecukható menü
- **Felhasználói menü:** Bejelentkezett felhasználó kezelése
- **Oldalon belüli navigáció:** Smooth Scroll szekciókhoz

Felhasználói Interakciók A fejléc a következő felhasználói interakciókat kezeli:

- **Menü megnyitása/zárása:** Mobilnézetben
- **Szakaszra ugrás:** Sima görgetés a főoldal szekcióihoz
- **Felhasználói menü:** Profilkezelés és kijelentkezés
- **Közösségi média linkek:** Külső oldalakra mutató hivatkozások

A komponens az `AuthService`-t használja a felhasználói állapot kezelésére, és a `Router` segítségével navigál az alkalmazásban. A reszponzív viselkedés érdekében számos `@HostListener` dekorátorral figyeljük az ablak méretváltozását és scroll eseményeket.



13. ábra. Transzparens header az oldal tetején

5.8.3. CourseCardList Komponens

A `CourseCardList` komponens egy interaktív, görgethető kurzuslistát valósít meg a főoldalon, Swiper könyvtár segítségével. A komponens a "Courses" szekcióban jelenik meg.

Főbb jellemzők

- **Dinamikus adatforrás:** 8 előre definiált kurzus szerepel a komponensben.
- **Swiper integráció:**
 - Végtelen görgetés (loop)
 - Középre igazított diák
 - Reszponzív megjelenés (1.2–4.2 látható dia az eszköz méretétől függően)
- **Navigáció:**
 - Nyilak az oldalváltáshoz
 - Paginációs pontok

Technikai részletek

- **Swiper konfiguráció:**
 - `slidesPerView`: 'auto': Automatikus dia szám
 - `spaceBetween`: 50: Diák közötti térköz
 - `breakpoints`: Reszponzív viselkedés meghatározása különböző eszközökre
- **Angular integráció:**
 - `CUSTOM_ELEMENTS_SCHEMA` használata a Swiper web komponenshez
 - Inicializálás az `ngAfterViewInit` élelciklus hookban

Reszponzív viselkedés

- **Mobil (0–768px):**
 - 1.2 látható dia
 - Középre igazítás
- **Tablet (768–1024px):**
 - 1.6 látható dia
- **Desktop (1024px+):**
 - 4.2 látható dia

5.8.4. CourseCard Komponens

A `CourseCard` komponens egyedi kurzus kártyákat jelenít meg a `CourseCardList` komponensen belül. Minden kártya tartalmazza a kurzus címét, leírását és egy releváns ikont.

Mindkét komponens együttműködve biztosítja a kurzusok modern, felhasználóbarát megjelenítését a főoldalon, különösen mobil eszközökön, ahol a görgetés természetes interakciót biztosít.

5.9. Login

A `LoginComponent` lehetővé teszi a felhasználók számára, hogy bejelentkezzenek email és jelszó megadásával, vagy alternatív módon QR kód alapú azonosítással. A bejelentkezési adatok ellenőrzése után a felhasználó a portál felületére kerül átirányításra. QR kód használata esetén a komponens valós idejű csatornán figyeli az azonosítás sikerességét, és automatikusan kezeli a hibákat, illetve a QR kód lejáratát.

5.10. Portál komponensek

5.10.1. Portal komponens

- Fő konténer a portál felületnek
- Tartalmazza a fejléct (`PortalHeaderComponent`) és a router kimenetet
- Minimalista struktúra, csak a szükséges elemeket tartalmazza

5.10.2. PortalHeader komponens

- Navigációs fejléc a portálhoz
- Főbb jellemzők:
 - Reszponzív menü (hamburger menü mobilon)
 - Felhasználói profil ikon és dropdown
 - Session időzítő megjelenítése
 - Role-alapú menüelemek (admin/teacher látható linkek)
- Animációk: Chevron forgatás, dropdown fade-in
- Funkciók: Kijelentkezés, mobil menü kezelés

5.10.3. Dashboard komponens

- Fő irányítópult a portál belépés után
- Fő szekciók:
 - **Timeline:** Szűrhető eseménylista a közelgő eseményekkel Swiper komponenssel

- **Calendar:** Interaktív naptár szűrőkkel
- **Subscribed Events:** Feliratkozott események (csak admin/teacher)
- **Event Subscriptions:** Eseményenkénti feliratkozott felhasználók megjelenítése (csak admin)
- CRUD műveletek: Esemény létrehozás, szerkesztés, törlés

5.10.4. TodoList komponens

- Felhasználói teendők listájának megjelenítése
- Főbb funkciók:
 - Szűrés státusz, dátum és cím szerint
 - Rendezés cím vagy dátum szerint
 - CRUD műveletek (létrehozás, szerkesztés, törlés)
- Komponens kommunikáció:
 - TodoCard komponensek megjelenítése
 - TodoForm megnyitása szerkesztéshez

5.10.5. TodoCard komponens

- Egyedi teendő kártya megjelenítése
- Tartalmazza:
 - Cím és leírás
 - Státusz és dátum információk
 - Szerkesztés/törlés gombok (ha láthatóak)
- Események kibocsátása a szülő komponens felé

5.10.6. TodoForm komponens

- Teendők létrehozására/szerkesztésére szolgáló modal
- Mezők:
 - Cím (kötelező)

- Leírás
- Státusz (csak szerkesztésnél)
- Reszponzív form design Material komponensekkel

5.10.7. NewsForm Komponens

Egy dialógusablak komponens, amely lehetővé teszi hírek létrehozását és szerkesztését. A komponens két módban működhet: "add" (új hír hozzáadása) és "edit" (meglévő hír szerkesztése). A form tartalmaz egy cím (title) és tartalom (body) mezőt, mindkettő kötelező. A komponens Material Design elemeket használ, és a beviteli mezők validációját kezeli.

5.10.8. UserControl Komponens

Ez a komponens a felhasználók kezeléséért felelős. Megjeleníti a felhasználók listáját, keresési és szűrési lehetőségeket biztosít, valamint lehetővé teszi új felhasználó hozzáadását és meglévők szerkesztését. A komponens betölti a felhasználókat a DataService segítségével, és kezeli a válaszüzeneteket (sikeres vagy hibás műveletek esetén).

5.10.9. UserCardList Komponens

Egy felhasználói kártyákat megjelenítő lista, lapozási funkcióval. A komponens be-menetként kap egy felhasználólistát, és lapozással megjeleníti a felhasználókat. A lapozás gombokkal (előre, hátra) vezérelhető. A komponens dinamikusan frissíti az oldalak számát a felhasználók számának változása esetén.

5.10.10. UserCard Komponens

Egy felhasználói kártya, amely egy felhasználó alapvető adatait jeleníti meg (pl. felhasználónév, email, szerepkör, csatlakozás dátuma). A kártya tartalmaz gombokat a felhasználó szerkesztéséhez és részletek megjelenítéséhez. Az admin szerepkörű felhasználók számára további műveletek (szerkesztés) érhetők el.

5.10.11. UserSearch Komponens

Egy keresőkomponens, amely lehetővé teszi felhasználók szűrését felhasználónév vagy email alapján, valamint szerepkör és rendezési szempont szerint. A komponens debounce technikát használ a keresési mezőben, hogy csökkentse a felesleges lekérdezések számát. A szűrők törlésére is van lehetőség.

5.10.12. UserForm Komponens

Egy űrlapkomponens, amely új felhasználó hozzáadására vagy meglévő szerkesztésére szolgál. Az űrlap tartalmazza a felhasználónév, email (csak új felhasználó esetén), jelszó és jelszó megerősítés mezőket. A jelszó erősségét validálja (szám, nagybetű, speciális karakter). Az űrlap Material Design elemeket használ, és a bevitt adatok validációját kezeli.

5.10.13. UserDetails Komponens

Ez a komponens egy felhasználó részletes adatait jeleníti meg. Tartalmazza a felhasználó kártyáját, személyes naptári eseményeket és teendőket (Todos). A komponens betölti a felhasználóhoz tartozó teendőket a DataService segítségével, és kezeli a betöltési állapotot és hibákat. A komponens visszaugrást tesz lehetővé a felhasználólistára egy "Vissza a listához" gomb segítségével.

5.10.14. UserEventList Komponens

Egy eseménylista komponens, amely egy felhasználó személyes eseményeit jeleníti meg. Két módon működhet: vagy betölti a felhasználó eseményeit a megadott userId alapján, vagy megjeleníti a közvetlenül átadott eseményeket. A komponens tartalmaz egy betöltési spinner-t és üzenetet, ha nincsenek események. Az események leiratkozását is kezeli, ha a showUnsubscribe bemeneti értéke igaz.

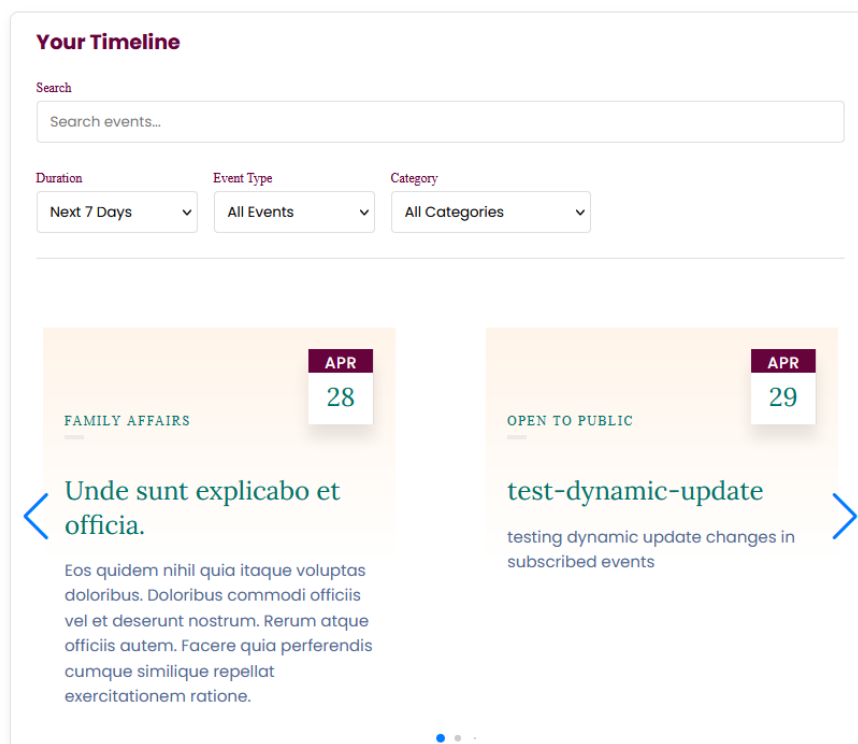
5.10.15. UserEventCard Komponens

Egy eseménykártya komponens, amely egyetlen naptári esemény részleteit jeleníti meg. Az esemény címét, dátumát, típusát, leírását és létrehozási/módosítási dátumát mutatja. Ha a showUnsubscribe bemeneti értéke igaz, akkor megjelenít egy leiratkozás gombot, amely eseményazonosító kibocsátásával jelez a szülő komponensnek a leiratkozási szándékról. A komponens Material Design kártya elemét használja az esztétikus megjelenítéshez.

5.10.16. Integráció

- A komponensek az AuthService-t használják jogosultságkezelésre
- Az eseménykezelés a DataService-n keresztül történik
- Reszponzív design minden komponensben (mobil/tablet/desktop)

Dashboard



14. ábra. dashboard komponens részlet

6. Mobil

6.1. Bevezető

A mobilalkalmazásunk egy .NET MAUI alapú mobilalkalmazás, amely kifejezetten a webes alkalmazás kiegészítésére készült. Azért a .NET MAUI mellett döntöttünk, mert ezt a környezetet tanultuk az iskolában, így ehhez biztosan értünk mindhárman, így ez sem vet gátat az effektív munka számára.

Mindenki képes átlátni és értelmezni a kódot, így nem csak a végtermékről ítélnünk, hanem képesek vagyunk jelezni a hibát egymásnak, és javaslatot tehetünk annak megoldására is.

A rendszer három fő funkcionalitással rendelkezik:

- QR kód alapú beléptető rendszer
- Teendők (todo) kezelési felület
- Egyetemi események böngészése és azokra való jelentkezés

Célunk, hogy az egyetemisták egyszerűen és hatékonyan tudják nyomon követni feladataikat és a közös/egyetemi eseményeket, akár egy mobilos alkalmazás segítségével, illetve amennyiben be szeretnének jelentkezni a webes alkalmazásba, a QR kód olvasó funkciónk segítségével azt egyszerűen legyenek képesek megtenni.

6.2. Alkalmazás főbb funkciói

6.2.1. Biometrikus azonosítás

A biometrikus azonosítás funkció lehetővé teszi, amennyiben már egyszer az adott készülékről bejelentkezett valaki a hagyományos e-mail cím jelszó módszerrel, a következő alkalommal elég csak a telefon ujjlenyomat-olvasóját használni a bejelentkezéshez.

6.2.2. QR kód alapú beléptetés

A QR kód beléptetés segítségével a felhasználók gyorsan be tudnak jelentkezni. Az alkalmazás a telefon kamerája használatával beolvassa a kódot, és egy serveroldali API hívással érvényesíti a belépést.

6.2.3. Teendőlista

A teendőlista lehetővé teszi a felhasználók számára, hogy feladatokat vegyenek fel, szerkesszenek vagy töröljenek. Az adatok egy szinkronizált szolgáltatáson keresztül kezelhetők az alkalmazásban.

6.2.4. Események kezelése

Az eseménylista segítségével képesek a felhasználók egyszerűen jelentkezni és jelentkezésüket visszavonni a közös/egyetemi eseményekről. Az eseményekről minden szükséges információt megtalálhatnak a felhasználók a felsorolásban, névszerint a dátumát, típusát, nevét és a leírását is.

6.3. Technológiai háttér

- **Platform:** .NET MAUI
- **Nyelv:** C#, XAML
- **QR szkennel:** ZXing.Net.MAUI

- **HTTP hívások:** HttpClient
- **Biztonság:** SecureStorage tokenek, jelszavak tárolására
- **Adatkezelés:** Saját TodoService/AuthService, REST API kapcsolat

6.4. QR kód beléptető rendszer működése

6.4.1. XAML felület

A QRCodePage.xaml tartalmazza a felhasználói felületet:

- Egyetemi logó
- Cím ("Scan a QR Code to Login")
- CameraBarcodeReaderView komponens
- Eredmény megjelenítő Label

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
3             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4             xmlns:zxing="clr-namespace:ZXing.Net.Maui.Controls;assembly=ZXing.Net.Maui.Controls"
5             x:Class="UniMobile.Views.QRCodePage"
6             Title="QRCodePage">
7      <VerticalStackLayout Padding="20" Spacing="20">
8          <Image Source="logo_2.png" HeightRequest="50" HorizontalOptions="Start"></Image>
9          <Label Text="Scan a QR Code to Login"
10             HorizontalOptions="Center"
11             FontSize="20"
12             TextColor="#E38946" />
13      </VerticalStackLayout>
14      <VerticalStackLayout Padding="40" Spacing="20">
15          <zxing:CameraBarcodeReaderView x:Name="cameraBarcodeReaderView"
16             HeightRequest="300"
17             WidthRequest="300"
18             IsDetecting="True"
19             BarcodesDetected="CameraBarcodeReaderView_BarcodesDetected" />
20          <Label x:Name="ResultLabel"
21             Text="Waiting for QR code..."
22             HorizontalOptions="Center"
23             FontSize="14" />
24      </VerticalStackLayout>
25  </ContentPage>
```

15. ábra. XAML részlet

6.4.2. C# logika

- A QR kód beolvasása után az esemény a CameraBarcodeReaderView_BarcodesDetected metódus kerül meghívásra.

- Token és felhasználói adatok lekérése
- JSON POST kérés az API-hoz
- Visszajelzés megjelenítése a felületen a felhasználó felé

```
var request = new HttpRequestMessage(HttpMethod.Post, "http://3.127.249.5:8000/api/qrcode/login")
{
    Content = content
};
request.Headers.Authorization = new AuthenticationHeaderValue("Bearer", authToken);
```

16. ábra. C# kód részlet

6.5. Teendőlista működése

6.5.1. XAML felület

- Button a teendő felvételhez.
- CollectionView a teendők listázásához.
- SwipeView elem törlési lehetőséggel. (Balra húzás esetén)
- Teendő kiválasztása esetén módosítási törlési lehetőség.

```
<SwipeItem Text="Delete"
Backgroundcolor="Red"
Invoked="onDeleteSwipeItemInvoked"
CommandParameter="{Binding .}" />
```

17. ábra. XAML részlet

6.5.2. C# logika

- LoadTodos: betölti a listát a szerverről a TodoService segítségével.
- OnAddTodoClicked: navigál a részletes szerkesztő oldalra
- onDeleteSwipeItemInvoked: teendő törlése megerősítés után

```
private async void OnDeleteSwipeItemInvoked(object sender, EventArgs e)
{
    if (sender is SwipeItem swipeItem && swipeItem.CommandParameter is Todo todoToDelete)
    {
        bool confirm = await DisplayAlert("Delete", $"Are you sure you want to delete '{todoToDelete.Title}'?", "Yes", "No");
        if (!confirm)
            return;

        bool success = await _todoService.DeleteTodoAsync(todoToDelete.Id);
        if (success)
        {
            await DisplayAlert("Deleted", "Todo deleted.", "OK");
            LoadTodos();
        }
        else
        {
            await DisplayAlert("Error", "Failed to delete todo.", "OK");
        }
    }
}
```

18. ábra. C# kód részlet

6.6. Hibakezelés és Információközlés

- Felhasználói és rendszerhibák megjelenítése DisplayAlert segítségével
- Kamera hiba, hiányzó adat, sikertelen API hívás esetén a felhasználó számára egy értelmezhető és egyszerű indok/válasz nyújtása.

6.7. Összegzés

A UniMobile egy egyszerű, modern és biztonságos alkalmazás, amely valódi problémákra ad választ az egyetemi mindennapok során. A biometrikus azonosítás gyorsá és megbízhatóvá teszi a bejelentkezést, míg a teendőlista segít rendet tartani a kaotikus mindennapokban.

7. Üzemeltetés kérdése

Tisztában vagyunk azzal, hogy az üzemeltetés gyakran egy másik csapat feladata, de a tapasztalatszerzés miatt mi is szeretnénk magunkévá tenni ezt a feladatot is.

Üzemeltetési lehetőségek:

- Az iskolánk által biztosított szerver
- Az AWS által üzemeltetett EC2 szervergép
- Laravel Forge (ez egy nagyobb költséggel járó alternatíva)

- Laravel Cloud (sajnos ahhoz, hogy ingyenesen legyünk képesek igénybe venni ezt a szolgáltatást csak SQLite alapú adatbázist használhatunk)

Így végül, mivel stressz-tesztet is óhajtottunk végrehajtani a rendszeren, ezért a saját EC2-re esett a választásunk. Itt a frontend üzemeltetéséhez apache web szervert használunk. A backendnél viszont egy kicsit több dolgot vettünk számba, hogy a lehető legjobban optimalizált legyen. Először is a Laravel által biztosított “php artisan serve” egy fejlesztői környezetben használatos kiszolgálást biztosít, amelynek hátránya, hogy azt nem lehet úgy skálázni, mint az apache, nginx, php fpm vagy a Laravel Octane segítségével lehet. Végül a backend egy FrankenPHP alapú Laravel Octane által kerül kiszolgálásra.

7.1. Docker Konténerizálás

DevOps tudásunk révén projektünket konténerizáltuk, hogy az áttérés másik gépre vagy operációs rendszerre zökkenőmentes legyen. Ehhez három konténert definiáltunk:

- MySQL szerver – adatbázis szolgáltatás
- Laravel backend – API és üzleti logika
- Angular frontend – weboldal kiszolgálása

8. Projekt futtatása lokálisan

Amennyiben a projekt backend és frontend oldalát lokálisan szeretné futtatni, két lehetősége van:

- Docker segítségével a gyökér könyvtárban futtat egy "docker compose up" parancsot.
- Windows eszközön lefuttatja a két batch file-t (install.bat és a run.bat), illetve egy lokális MySQL szerver futtatása is szükséges.

Előfeltétel viszont a backend számára a .env fájl, ami nem megtalálható a github projektünkben. Ezért a lokális futtatáshoz szükséges .env fájl Záródolgozat mellékleteként került leadásra. Ezt a fájlt a szükséges bemásolni a backend_Uni mappába.

9. Jelenlegi Állapot és a Jövő

9.1. Jelenlegi Állapot

A projektben sikeresen megvalósítottuk az összes olyan funkcionalitást, amelyet a kezdetekben tűzünk ki magunknak:

- Egyetemi kalendárium
- Egyetemi hírportál
- Személyes kalendárium
- Személyes feladatok
- Univerzális kezelőfelület
- Mobilos applikáció
- QR Code-al való belépés
- Admin és Teacher kezelőfelülete

Ezenek mellett sikeresen implementáltuk a docker-konténerizálást is.

9.2. Demo adatok bejelentkezéshez

Szerepkör	Email	Jelszó
Admin	admin@admin.com	admin
Teacher	teacher@teacher.com	teacher
User	user@user.com	user

9.3. Jövő

A jövőben a projekt célja, hogy továbbra is minden, tervezéskor meghatározott feladatot ellásson, és egy valódi egyetem mindennapjaiban is alkalmazható legyen. A rendszernek a felhasználói élményt nem szabad sértenie/rontania, miközben optimalizált és hatékony marad.

A közeljövőben implementálandó funkciók:

- CI/CD Pipeline a fejlesztési munka megkönnyítésére.
- Kubernetes alapú üzemeltetés a még intenzívebb terhelések kezelésére.
- Órák felvétele/kezelése.
- Mobilos értesítések teendőkhöz/eseményekhez.
- Naptár integráció mobil alkalmazásban (napár nézet).

10. Felhasználói útmutató

A következő részben szeretnénk egy átfogó tájékoztatót nyújtani azoknak, akik nem fejlesztőként, hanem felhasználóként olvassák a dokumentációt.

10.1. Rövid Útmutató a Webes Alkalmazáshoz

10.1.1. Elérés és belépés

- 1) Nyisd meg a böngészőben a webes alkalmazást a következő címen: `http://52.28.154.228/`.
- 2) A fejléc jobb felső sarkában kattints a **Login** gombra.
- 3) Add meg e-mailed és jelszavad, majd nyomj a **Login** gombra.
- 4) Sikeres belépés után a portál (privát felhasználói felület) főoldalára (Dashboard) kerülsz.

10.1.2. Globális navigáció (HeaderComponent)

Minden oldalon a lap tetején találod a fix fejléctet, benne:

- **Logó** (visszavisz a nyilvános főoldalra),
- Főmenü-elemek (**Main**, **Portal**, **Events**, **Todos**),
- Profil-menü (felhasználóneved, **Sign Out**).

10.1.3. Nyilvános főoldal (MainComponent)

A főoldal azoknak szól, akik még nincsenek bejelentkezve, vagy csak áttekinteni szeretnék az egyetemi híreket és eseményeket.

10.1.3.1 Hírek

- A hírek kártyás (grid) elrendezésben jelennek meg.
- Kattints egy hírkártyára a teljes tartalomért.
- Lapozz a következő oldalakra a grid alján lévő vezérlőkkel.

10.1.3.2 Esemény-slider

- A legközelebbi, közelgő események Swiper-sliderben jelennek meg.
- Lépegetsz balra/jobbra, vagy az egérrel húzd a kártyákat.

10.1.3.3 Nyilvános naptár-nézet

- Görgess le a „Calendar” szekcióhoz: havi nézetben látod az eseményeket.
- Kattints egy eseményre annak részleteinek megjelenítéséhez.

10.1.4. Portál (privát felület)

Bejelentkezés után a **Dashboard** (irányítópult) nyílik meg:

10.1.4.1 Dashboard felépítése

- **Timeline** – közelgő események listája, fel-/leiratkozási gombokkal.
- **Calendar** – interaktív naptár, kattintással részletesen láthatod az eseményeket, illetve hogy az eseményre fel vagyunk-e iratkozva
- **Event Subscriptions** (adminoknak) – listázza, kik iratkoztak fel egy eseményre.

10.1.4.2 Teendők (Todos)

- Nyisd meg a **Todos** menüpontot.
- Új tétel: **Add Todo**, töltsd ki a címet és leírást, majd **Save**.
- Tétel törlése: húzd el balra, vagy kattints a **Delete** gombra.
- Szerkesztés: kattints a tételre, módosítsd, majd **Save**.

10.1.4.3 Személyes naptár

- **Personal Calendar** menüben saját eseményeidet hozhatod létre, szerkesztheted, törölheted.
- Új esemény: **Add Event**, add meg a címet, leírást, típust és dátumot.
- Módosítás/törlés ikonok a részletező modalban.

10.2. Rövid Útmutató a Mobilos Alkalmazáshoz

10.2.1. Be- és Kijelentkezés

- 1) Nyisd meg az alkalmazást, majd jelentkezz be e-mail és jelszó megadásával, amennyiben ezt egyszer már megtetted, és a telefonod is támogatja, képes leszel ujjlenyomatazonosító vagy arc-szkenner segítségével is belépni a későbbiekben.
- 2) Ezután át leszel irányítva a Teendők menüpontra.
- 3) Amennyiben ki szeretnél jelentkezni, nyisd meg a "Profil" menüpontot.
- 4) Kattints az "Sign Out" gombra.
- 5) Töltsd ki a címet és a leírást.
- 6) Ebben a menüpontban látod a felhasználóneved és az emailed is, amivel bejelentkeztél.

10.2.2. QR beléptetés

- 1) Válaszd ki a "QR Login" menüpontot .
- 2) Tartsd a kamerát a kód fölé.
- 3) Sikeres beolvasás után visszajelzést kapsz, és átirányít a Teendők menüpontra.

10.2.3. Teendők kezelése

- 1) Nyisd meg a "Todos" menüpontot.
- 2) Kattints az "Add Todo" gombra.
- 3) Töltsd ki a címet és a leírást.

- 4) Mentés után megjelenik a listában.
- 5) Törléshez húzd el balra az elemet, vagy nyomj rá, és ott válaszd ki a "Delete" gombot.
- 6) Szerkesztéshez nyomj rá az elemre, írd át, majd nyomj a "Save" gombra.

10.2.4. Események kezelése

- 1) Nyisd meg az "Events" menüpontot.
- 2) Böngéssz a közeli események között.
- 3) A gomb megnyomása után értesít az alkalmazás, hogy éppen fel- vagy leiratkoztál az eseményről, és a gomb színe és szövege is megváltozik.

11. Összefoglalás

A **Pandidakterion** projekt célja az egyetemi élet megkönnyítése volt, hogy mind a hallgatók, mind az oktatók számára egyszerűbbé és hatékonyabbá tegye a mindennapi adminisztratív feladatok kezelését/végrehajtását. A projekt során sikerült egy olyan platformfüggetlen rendszert kialakítani, amely mind a webes, mind a mobil felületen magas szintű biztonságot, skálázhatóságot és felhasználóbarát élményt nyújt.

A csapat megfelelően használta a modern technológiákat a fejlesztési fázisban, név szerint a **Laravel** keretrendszert a backendhez, az **Angular**-t a frontendhez, valamint a **.NET MAUI**-t a mobilalkalmazás fejlesztéséhez. A rendszer architektúrája kiemelkedően kezeli a nagyobb terheléseket, köszönhetően az AWS EC2 szerveren üzemelő konténerizált környezetnek, amely rendkívül optimalizált futtatási megoldásokat tartalmaz, illetve az optimalizált adatbázis-lekérdezéseknek. A fejlesztési folyamat során a **GitHub**-on keresztüli együttműködés segítette a csapatunk hatékonyságát.

A projekt legnagyobb erőssége a valós problémákra történő fókuszálás: a platformfüggetlenség, a feladatok és események áttekinthetősége, valamint a QR-kódos belépés mind konkrét igényekből/észrevételekből alakultak ki programunk funkcióivá. A mobilos alkalmazás kiegészíti a webes felületet, lehetővé téve a felhasználók számára, hogy akár webes kliens nélkül is hozzáférjenek az alapvető funkciókhoz.



A jövőben a rendszer további funkciókkal bővíthet, például órarend-kezeléssel, telefonos értesítésekkel és még effektívebb üzemeltetési módszerekkel (kubernetes). A jelenlegi állapottal is meg vagyunk elégedve, sikerült rengeteg olyan funkciót implementálni, amelyek megvalósítása alatt sok tapasztalatot szereztünk.

12. Irodalomjegyzék

1. Laravel. Backend PHP keretrendszer. Elérve: 2025. április 7. url: <https://laravel.com/>.
2. Laravel Reverb. Laravel Reverb - Websocket kommunikációs eszköz. Elérve: 2025. április 10. url: <https://laravel.com/docs/12.x/reverb>.
3. Laravel Octane. Laravel Octane - Laravel Teljesítményfokozó csomagja. Elérve: 2025. április 12. url: <https://laravel.com/docs/12.x/octane>.
4. FrankenPHP. FrankenPHP - Modern PHP applikációs szerver. Elérve: 2025. április 12. url: <https://frankenphp.dev/docs/>.
5. PHPUnit. PHPUnit - PHP tesztelési környezet. Elérve: 2025. április 13. url: <https://docs.phpunit.de/en/12.1/>.
6. DBDiagram - Adatbázis vizualizáló. Elérve: 2025. április 13. url: <https://dbdiagram.io/home>.
7. .NET MAUI - Multi-platform alkalmazás fejlesztő keretrendszer. Elérve 2025. április 11. url: https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-9.0&WT.mc_id=dotnet-35129-website.
8. Scramble - Scramble dedoc API végpont vizualizáló. Elérve: 2025. április 13. url: <https://scramble.dedoc.co/>
9. AWS EC2 - Felhő alapú szervergép. Elérve: 2025. április 13. url: <https://aws.amazon.com/ec2/>
10. Docker - Docker. Elérve: 2025. április 13. url: <https://www.docker.com/>
11. Angular - Frontend keretrendszer. Elérve: 2025. április 13. url: <https://angular.dev/overview>
12. SwiperJs - Egyszerűen kezelhető slider. Elérve: 2025. április 25. url: <https://>



`swiperjs.com/get-started`

13. Apache - Http szerver frontend üzemeltetéséhez. Elérve: 2025. április 25. url:

`https://httpd.apache.org/`

14. Jest - Frontend tesztelési környezet. Elérve: 2025. április 25. url: `https://jestjs.io/`