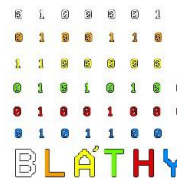




**Budapesti Műszaki Szakképzési Centrum Bláthy**  
Ottó Titusz Informatikai Technikum 1032 Budapest,  
Bécsi út 134. OM azonosító: 203058/002



# Vizsgaremek

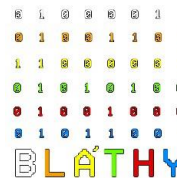
Kórus applikáció

Application for a Choir

Barkainé Verpec Mónika

Szijártó Tímea

Gecsei Zoltán



# Vizsgaremek adatlap

## A Vizsgaremek készítői:

Neve: Barkainé Verpec Mónika  
E-mail címe: monikaverpec@gmail.com

Neve: Szijártó Tímea  
E-mail címe: timea.szijarto@gmail.com

Neve: Gecse Zoltán  
E-mail címe: gecsezoltan86@gmail.com

## A Vizsgaremek témája:

Egy amatőr kórus életét megkönnyítő és rendszerező alkalmazás készítése, mely tartalmazza a tagok adatait, tagdíjbefizetéseket, kórusnaptárt, műsorokat stb.

## A Vizsgaremek címe:

Kórus applikáció

Application for a Choir

Konzulens: Gavallér György

Kelt: Budapest, 2023. november 12.

.....  
Barkainé Verpec Mónika

.....  
Gecse Zoltán

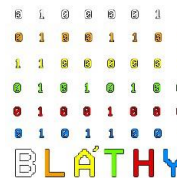
.....  
Szijártó Tímea

.....  
Gavallér György

A Vizsgaremek készítőinek aláírása, a konzulens aláírása.



**Budapesti Műszaki Szakképzési Centrum Bláthy**  
Ottó Titusz Informatikai Technikum 1032 Budapest,  
Bécsi út 134. OM azonosító: 203058/002



# Eredetiség nyilatkozat

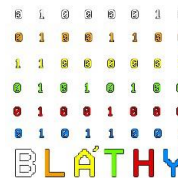
Alulírott tanuló kijelentem, hogy a vizsgaremek saját és csapattársa(i)m munkájának eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült vizsgaremekrészét képező anyagokat az intézmény archiválhatja és felhasználhatja.

Budapest, 2023.11.12.

.....  
Barkainé Verpec Mónika  
Tanuló aláírása

.....  
Szijártó Tímea  
Tanuló aláírása

.....  
Gecse Zoltán  
Tanuló aláírása



# Vizsgaremek konzultációs lap

Alulírott „Gavallér György” konzulens aláírásommal igazolom Barkainé Verpec Mónika, Szijártó Tímea, Gecse Zoltán nevű tanulók konzultációkon való részvételét<sup>1</sup>.

Dátum	Téma	Tanuló aláírása	Konzulens aláírása

<sup>1</sup> (A vizsgaremek leadásig kötelező minden tanulónak legalább három konzultációs alkalom, ennek hiányában a záródolgozat nem értékelhető!)

## Tartalomjegyzék

1. Projekt ismertetése .....	Hiba! A könyvjelző nem létezik.
2. Felhasználói dokumentáció .....	Hiba! A könyvjelző nem létezik.
a. Felületek ismertetése .....	Hiba! A könyvjelző nem létezik.
b. Funkciók ismertetése .....	Hiba! A könyvjelző nem létezik.
ii. Bejelentkezés.....	Hiba! A könyvjelző nem létezik.
iv. Naptár .....	Hiba! A könyvjelző nem létezik.
v. Kottatár.....	Hiba! A könyvjelző nem létezik.
vi. Hírek.....	Hiba! A könyvjelző nem létezik.
vii. Események:.....	Hiba! A könyvjelző nem létezik.
viii. Tagok listája .....	Hiba! A könyvjelző nem létezik.
ix. Befizetések.....	Hiba! A könyvjelző nem létezik.
c. Használati instrukciók .....	Hiba! A könyvjelző nem létezik.
3. Fejlesztői dokumentáció.....	Hiba! A könyvjelző nem létezik.
a. Munkamegosztás.....	Hiba! A könyvjelző nem létezik.
b. Telepítési dokumentáció .....	Hiba! A könyvjelző nem létezik.
c. Adatbázis dokumentáció .....	Hiba! A könyvjelző nem létezik.
i. Diagram .....	Hiba! A könyvjelző nem létezik.
ii. Mezők részletes ismertetése .....	Hiba! A könyvjelző nem létezik.
iii. Kapcsolatok indoklása ismertetése .....	Hiba! A könyvjelző nem létezik.
d. Komponens/struktúra dokumentáció.....	Hiba! A könyvjelző nem létezik.
i. Projekt elemei fejlesztés szempontjából.....	Hiba! A könyvjelző nem létezik.
3. Pl.: Controller/BusinessLogic Dokumentáció .....	Hiba! A könyvjelző nem létezik.
f. Teszt dokumentáció .....	Hiba! A könyvjelző nem létezik.
4. Továbbfejlesztési lehetőségek .....	Hiba! A könyvjelző nem létezik.
5. Összegzés.....	Hiba! A könyvjelző nem létezik.
6. Források, ábrajegyzék.....	Hiba! A könyvjelző nem létezik.

# 1. Projekt ismertetése

A vizsgaremekünk célja, hogy a csoporttagunk kórusának egy olyan alkalmazást hozzunk létre, amely segítségével könnyebb a kórustagokkal való kapcsolattartás, a különböző eseményekre való jelentkezés, az esetleges kották elérhetősége, különböző hírek megosztása. Manapság mindenhez a telefonunkat használjuk, e-mail küldése, hírek olvasása stb.

Ez a program lehetővé teszi, hogy a kórus adminjai betöltsék az információkat az adatbázisba és kezeljék a felületet, amit a kórustagok látnak, pl: a meghirdetett próbákra jelentkezzenek, vagy akár lemondják azt, mert életszerű, hogy betegség, vagy más előre nem tervezett elfoglaltság miatt lemondják a próbát, szinte csak egy kattintással. Ugyanilyen lehetőséget kínál az időnkénti fellépésekre való jelentkezés, ill. annak visszamondása, ha a kórustag mégsem tud ezen részt venni.

Nagyon hasznos lenne, ha a kórustagok az egyes kórusénekek kottáját egy helyen elérhetnék, ezért ezeknek a digitalizálása egyre sürgetőbbé vált, illetve ezeknek egy adatbázisban való tárolása, amelyeket bármikor, bárholnan meg tudnának nézni. Ez azért is életszerű, mert a próbák során is változhat egyes repertoár próbája, így nem kell fizikailag minden kottát maguknál tartani.

A kórusnak van egy éves tagsági díja, ezért felmerült az az igény, hogy az egyes tagok befizetését is el lehetne tárolni az adatbázisba, melyet le is lehetne kérdezni.

Vizsgaremekünk készítése során több probléma, kérdés merült fel, melyeket sajnos nem mind sikerült megoldani az idő rövidege miatt. Mivel épphogy a képzés végére értünk, és egy hét volt már csak a leadás végső határidejére, a sok új információ, a tanult programok elmélete és gyakorlata még nem üledett le kellőképpen. A megvalósítás során jöttünk rá, hogy egyes lépéseket más sorrendben kellett volna elvégezni.

## 2.Felhasználói dokumentáció

### a. Felületek ismertetése

Az applikáció admin és user felületből áll. Az admin felület frontendje Laravel Blade-ben íródott, a user felület frontendje pedig Vue.js-ben.

A user felület összesen hat felület jelenik meg a felhasználó számára, melyek a következők:

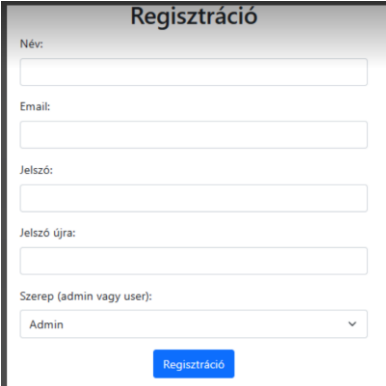
- Regisztráció felülete
- Bejelentkezés oldala
- Hiányzás bejelentő felülete
- Naptár megjelenítése
- Kottatár felülete
- Hírek oldal

Az admin oldal felületei:

- A felhasználók regisztrációja
- Események létrehozása és azok manipulálása
- Felhasználók adatai és azok manipulálása
- Befizetések naprakészen tartását szolgáló felület

### b. Funkciók ismertetése

#### i. Regisztráció.



The image shows a registration form with the title "Regisztráció". It contains the following fields and elements:

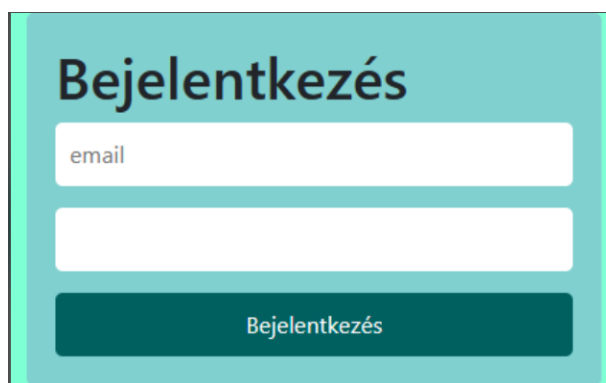
- Név: (Name) - text input field
- Email: (Email) - text input field
- Jelszó: (Password) - text input field
- Jelszó újra: (Repeat Password) - text input field
- Szerep (admin vagy user): (Role) - dropdown menu with "Admin" selected
- Regisztráció - blue button

1. ábra

A korusapp weboldala regisztrációhoz kötött, és így tudunk bejelentkezni a fiókunkba.

A regisztrációt az admin végzi el, ő adja a jelszót. Regisztráció gombra kattintva már készen is vagyunk a regisztrálással.

## ii. Bejelentkezés.

A screenshot of a login form titled "Bejelentkezés" (Login) in a teal-colored box. The form contains two input fields: the first is labeled "email" and the second is empty. Below the input fields is a dark teal button with the text "Bejelentkezés" (Login) in white.

2. ábra

Bejelentkezés során megadjuk az e-mail címünket és az admin által megadott jelszavunkat, majd ezt követően a Bejelentkezés gombra kell kattintani és ezzel meg is történt a bejelentkezésünk.

Fontos, hogy pontosan adjuk meg ezeket, mert téves felhasználónév és jelszó megadásánál nem tudunk bejelentkezni, mert különben egy felugró ablakban érkezik egy a hibaüzenet.

## iii. Hiányzás bejelentő

Hiányzás bejelentő oldalon lehetőségünk van megadni, hogy melyik próbáról fogunk hiányozni. Itt meg kell adnunk az adott próba dátumát. Egy úgynevezett "üzenetküldő" mezőben rövid szöveges indoklással is adhatunk erre magyarázatot, hogy miért nem tudunk részt venni a próbára, de ez nem kötelező, el is hagyható, opcionális.

## iv. Naptár

Ezen az oldalon ténylegesen egy naptár található. Az adott napra kattintva megjelenik, hogy van-e éppen valamilyen esemény aznap. Ez az esemény lehet egy énekpróba, fellépés, koncert,



dalverseny, vagy akár egy ruhapróba. Előfordulhat, hogy egy fellépés megköveteli az egységes dresscode-ot és ezért a kórustagok egy helyen varratják az ehhez szükséges fellépőruhát.

#### v. Kottatár

Ide gyűjtöttük az kórus élete során eddig használt kottákat, hogy egy helyen legyen, ha szükség van rá. Bármely kórustag elérheti. Tároltuk a mű zeneszerzőjének a nevét, a dal címét és magát a kottát pdf-es állományban. Lehetőségünk van keresésre a kották között, ez történhet úgy, hogy rákeresünk a zeneszerző nevére, vagy kereshetünk a dal címére. Miután megtaláltuk a keresett kottát, lehetőségünk van letölteni azt a saját eszközünkre.

#### vi. Hírek

Ezen az oldalon jelennek olyan fontos hírek, pl. hogy mikor lesz a legközelebbi kórus-találkozó. De a fellépések meghívóját is ideszerkeszthetjük, így egy esetleges külsős "vendég" látogató is értesülhet, hogy mikor és hol koncertezik a kórus, így bővíülhet a hallgatói kör is. A meghívón feltüntethetjük az adott fellépés repertoárját is, így akár még nagyobb érdeklődéssel lesz látogatottabb egy-egy koncert. Ezen az oldalon megjelenhet, ha a kórus hangfelvételen vagy tv felvételen volt, és itt lehet közzé tenni, hogy mikor lesz ez hallható a rádióban, vagy a televízióban, és melyik csatornán.

#### vii. Események:

Az admin ezen a felületen listázza ki az évad eseményeit és hozhat létre új eseményt az alábbi adatokkal: dátum pontos idővel, helyszín, cím, az éneklendő kotta sorszáma és további információk.

#### viii. Tagok listája

Ezen a felületen jelennek meg a tagok adatai (név, email cím, telefonszám, cím, születési dátum, szólam), valamint ezen adatokat tudja módosítani az admin.

#### ix. Befizetések

Ezen a felületen jelennek meg a tagok nevei, az eddig általa befizetett összeg, valamint egy mező, ahol az admin adminisztrálhatja a befizetéseket.

#### c. Használati instrukciók

Regisztrációnál fontos, hogy a jelszónak minimum 8 karakter hosszúságúnak kell lennie!

Bejelentkezésnél ügyeljünk rá, hogy a regisztrációnk során megadott email és jelszó párost adjuk meg, mert különben nem fogunk tudni bejelentkezni.

## 3. Fejlesztői dokumentáció

Az általunk készített program 3 fő részből tevődik össze.

- Adatbázis

Az adatbázis megjelenítésére a MySQL felületét használtuk (phpMyadmin)  
XAMPP Control Panel v3.3.0, Apache

- Frontend

Weboldalunk frontendjéhez a Vue JS keretrendszert alkalmaztuk, mivel ez szabványos JavaScript-, HTML-, illetve CSS-kódokra, épül, melyeket az elmúlt hónapokban tanultunk. Az admin felület Laravel Blade-ben íródott.

- Backend

A koruapp programunkat PHP programnyelven írtuk.

Weboldalunk backendjéhez PHP 8.2 hosszú távú támogatást ígérő verzióját használtunk mind a hárman. Figyeltünk rá, hogy mindegyikünkénél ez a verziószám legyen, mert így ennek megfelelően ellenőrzi a kódunkat.

Laravel

### a Munkamegosztás

A projektmunkát hárman végeztük.

Barkainé Verpec Mónika az adatbázis megtervezését, Dia programban való kialakítását, illetve MySQL programban való megjelenítését végezte és az dokumentációt.

Szűjártó Tímea végezte a backendben végzendő tevékenységeket.

Gecse Zoltán a frontendes feladatokat alakította ki, elvégezte az ehhez tartozó programozást, nagy hangsúlyt fektetve megjelenített felület létrehozására, a komponensek kialakítására.

A feladatfelosztásra kialakítottunk a projektünknek a Trello-n kórusapp néven egy oldalt. Itt rendszereztük feladatainkat, nyomon tudtuk követni, az aktuális részegységekkel melyikünk hol tart. Ellenőrizhettük az elvégzendő, a még folyamatban lévő és a már kész, befejezett teendőink listáját. Ezen kívül az elkészült fázisokat egy google drive-ra és a git-re tettük fel. Az éppen felmerült kérdéseket, problémákat és elakadásokat a Messenger csoportunkban beszéltük meg.

## b Telepítési dokumentáció

Kórusapp programunk használatára webes böngészőre lesz szükségünk, amennyiben ezzel nem rendelkezünk, telepítenünk kell egyet, ami nekünk szimpatikus, de akár többet is. Sőt ez tesztelés szempontjából még indokolt is. Lehet ez a Google Chrome, amit a Google fejlesztett, a Microsoft Edge, ami a Microsoft terméke, Firefoxot amit a Mozilla készített stb. Az oldalunk betöltéséhez természetesen még internet elérésére is szükség van.

Telepítenünk kellett a Xampp-ot, mert az oldalunk nem rendelkezik saját domain-nel. az adatbázis és a szerver kapcsolata érdekében.

Tehát kórusapp oldalunk webes felületről elérhető, bármely eszközről csatlakozhatunk hozzá, ami rendelkezik aktív internetkapcsolattal.

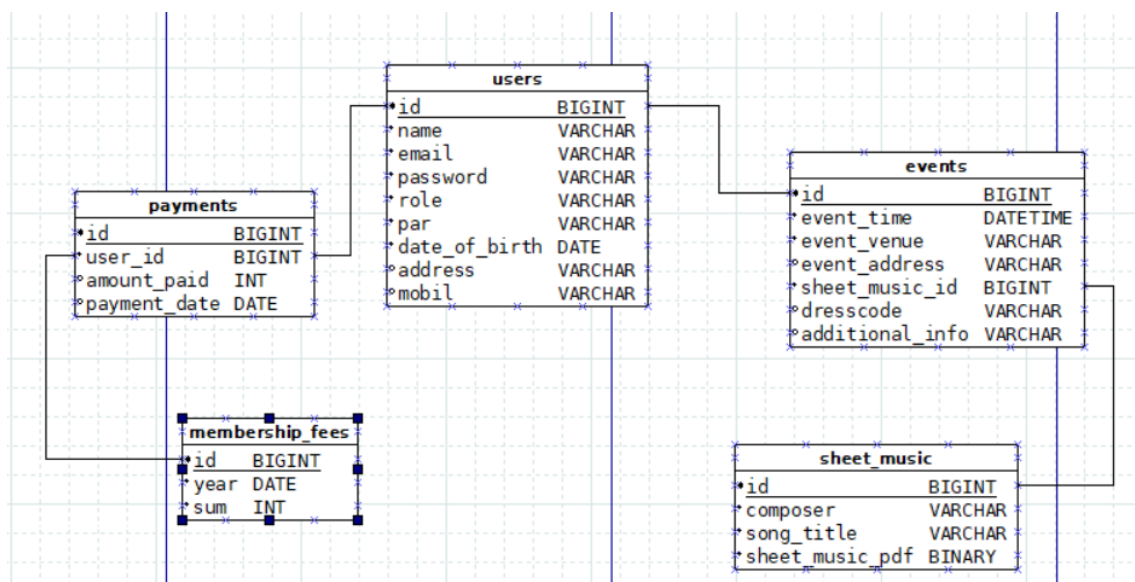
## c. Adatbázis dokumentáció

PHP Laravel programot használtunk, melyben Migration-ök segítségével hoztuk létre a táblákat, és Model-lel pedig az oszlopokat. Az adatbázist, melyet a XAMPP programban jelenítettünk meg. Így átláttuk a adatbázis táblázatait, oszlopait, bennük található mezőket. Ezt a fejlesztés során adatokkal tudtuk feltölteni, ami nagyon nagy segítségünkre volt, a különböző megírt kódok tesztelésére is.

### i. Diagram

Miután megbeszéltük, hogy a kórus életében mire lenne szükség, Dia program segítségével megterveztük az adatbázist a szükséges táblákkal és az azokhoz tartozó oszlopneveket, azok milyen típusok legyenek. Kialakítottuk az elsődleges

kulcsokat, és az idegen kulcsokat, illetve meghatároztuk a kapcsolódási pontokat. A fejlesztés során néhány változtatásra, bővítésre így is sor került, végül összesen öt tábla készült el.



3.ábra

## ii. Mezők részletes ismertetése

Adatbázisunk táblákból és azon belül mezőkből, különböző rekordokból tevődik össze, így létrejött egy összefüggő adatszerkezet.

A MySQL adatbázisokban az utf8mb4\_unicode\_ci-t használjuk, ami egy karakterkódolási és összehasonlítási szabályzat. A teljes Unicode karakterkészletet támogatja, kis- és nagybetű érzéketlen összehasonlítással bír, ezt a "ci" jelöli. Ezért mi is ezt a kódkészletet használtuk a mezőknél egységesen.

### 1. users tábla

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra
<input type="checkbox"/>	1 id 🔑	bigint(20)		UNSIGNED	Nem	Nincs		AUTO_INCREMENT
<input type="checkbox"/>	2 name	varchar(255)	utf8mb4_unicode_ci		Nem	Nincs		
<input type="checkbox"/>	3 email 📧	varchar(255)	utf8mb4_unicode_ci		Nem	Nincs		
<input type="checkbox"/>	4 email_verified_at	timestamp			Igen	NULL		
<input type="checkbox"/>	5 password	varchar(255)	utf8mb4_unicode_ci		Nem	Nincs		
<input type="checkbox"/>	6 remember_token	varchar(100)	utf8mb4_unicode_ci		Igen	NULL		
<input type="checkbox"/>	7 created_at	timestamp			Igen	NULL		
<input type="checkbox"/>	8 updated_at	timestamp			Igen	NULL		
<input type="checkbox"/>	9 role	varchar(255)	utf8mb4_unicode_ci		Nem	admin		
<input type="checkbox"/>	10 par	varchar(10)	utf8mb4_unicode_ci		Igen	NULL		
<input type="checkbox"/>	11 date_of_birth	date			Igen	current_timestamp()		
<input type="checkbox"/>	12 address	varchar(100)	utf8mb4_unicode_ci		Igen	NULL		
<input type="checkbox"/>	13 mobil	varchar(60)	utf8mb4_unicode_ci		Igen	NULL		

4.ábra

Ebben a felhasználók táblában alapvető információkat tárolunk a felhasználókhoz kapcsolódva.

id mező: típusa bigint, elsődleges kulcsot kap, mivel erre hivatkozva tudunk beazonosítani egy felhasználót. Auto\_increment-re van állítva, ami azt jelenti, hogy automatikusan a következő id értéke eggyel nagyobb értékkel növekszik, vagyis minden egyes új felhasználó esetén eggyel nő az id értéke.

name mező: típusa varchar, utf8mb4\_general\_ci illesztést kap

Ez a név mező, ez tulajdonképpen a felhasználó neve, amit regisztráláskor adott meg. Erre hivatkozva tud belépni a megadott jelszava párosításával, de pl. így tudja az oldal "köszönteni" belépéskor.

email mező: típusa varchar

Ebben az e-mail mezőben található a felhasználó e-mail címe, amit szintén a regisztráláskor adott meg. Ennek felhasználásával tudunk egy esetleges hírlevelet is kiküldeni a felhasználó számára.

password mező: típusa varchar

Ez a jelszó mező, ami nem nyilvános.

role mező (szerep mező): típusa varchar,

Itt lehet megadni a felhasználónak, hogy admin szerepet kap, vagy csak felhasználóként léphet be a programba.

Az admin kizárólagos szerepével tudja pl. a hírleveleket kezelni, a próbák és a fellépések időpontját, helyszínét megadni, milyen zeneműveket fognak énekelni, vagy akár az éves tagdíj összegét rögzíteni.

A user szereppel meg tudja nézni a felhasználó, hogy mikor és hol lesz az következő próba vagy fellépés, melyik dalokat fogják énekelni.

par mező (szólam mező): típusa varchar,

A felhasználó által énekelt szólamokat lehet berögzíteni, A -altot, M -mezot, S -szopránt jelent.

date\_of\_birth mező (születési dátum mező): típusa date,

A felhasználó születési dátumát tartalmazza ez a mező.

address mező (cím mező): típusa varchar,

A felhasználó címét láthatjuk ebben a mezőben. Ennek lekérése hasznos tud lenni, ha fellépésre megy a kórus akkor, ha esetleg valaki a fellépés helyszínének útvonalán lakik, akkor útközben be tud csatlakozni.

mobil mező (mobil telefonszám mező): Típusa varchar,

Bár a kapcsolattartásra elsősorban az e-mailt használják a kórustagok, de ha valamit sürgősen közölni kell, pl. egy fellépés vagy próba elmarad, vagy változott a helyszín vagy időpont, akkor nagyon hasznos, ha az adatbázisban tárolni tudjuk a felhasználók (kórustagok) mobil telefonszámát.

## 2. events tábla

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra
<input type="checkbox"/> 1	id 🗝️	bigint(20)			Nem	Nincs		AUTO_INCREMENT
<input type="checkbox"/> 2	event_time	datetime			Nem	Nincs		
<input type="checkbox"/> 3	event_venue	varchar(100)	utf8mb4_general_ci		Igen	NULL		
<input type="checkbox"/> 4	event_address	varchar(100)	utf8mb4_general_ci		Igen	NULL		
<input type="checkbox"/> 5	sheet_music_id 🗝️	bigint(20)			Igen	NULL		
<input type="checkbox"/> 6	additional_info	text	utf8mb4_general_ci		Igen	NULL		
<input type="checkbox"/> 7	created_at	timestamp			Igen	NULL		
<input type="checkbox"/> 8	updated_at	timestamp			Igen	NULL		

5.ábra

Ez az események tábla tartalmazza a próbákat, fellépésekhez tartozó információkat.

id mező: típusa bigint, hossza 20, auto\_increment-re állítva, elsődleges kulcsot kap, mivel erre hivatkozunk egy eseményre, ami lehet próba, vagy bármilyen fellépés. Auto\_increment-re van állítva, ami azt jelenti, hogy automatikusan a következő id értéke eggyel nagyobb értékkel növekszik, vagyis minden egyes új esemény esetén eggyel nő az id értéke.

event\_time: típusa datetime

Ez a mező tárolja a felhasználó tagdíj befizetésének idejét, év, hónap, nap-ot berögzítve.

event\_venue típusa varchar, hossza 250, utf8mb4\_general\_ci kódolású,

Az esemény helyszínét tartalmazza ez a mező.

event\_address varchar 250 utf8mb4\_general\_ci

Ebben a mezőben az esemény címét tároljuk az adatbázisunkban.

additional\_info text utf8mb4\_general\_ci

Ez egy szövegbeviteli mező. Arra az esetre van, ha valamely eseménnyel kapcsolatban közlendőnk van, akkor azt itt lehet megtenni.

### 3. membership\_fees tábla

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra
<input type="checkbox"/>	1 id 🔑	bigint(20)		UNSIGNED	Nem	Nincs		AUTO_INCREMENT
<input type="checkbox"/>	2 payment_id	bigint(20)		UNSIGNED	Nem	Nincs		
<input type="checkbox"/>	3 year	date			Nem	Nincs		
<input type="checkbox"/>	4 sum	int(11)			Nem	Nincs		
<input type="checkbox"/>	5 created_at	timestamp			Igen	NULL		
<input type="checkbox"/>	6 updated_at	timestamp			Igen	NULL		

6.ábra

Ez a tagdíj tábla, ahol az éves tagdíj összegét és az adott évet rögzítjük.

id mező: típusa bigint, elsődleges kulcsot kap, hossza 20 és szintén auto\_increment-re van állítva az automatikus értéknövelés érdekében.

year mező: típusa year. Ezt célszerű lesz date-re változtatni, és teljes év, hóHa esetleg év közben változna a tagdíj összege, akkor azt is tudjuk természetesen rögzíteni, be tudjuk írni a tagdíj dátumának pl.: 2024.05.01 és akkor ehhez a dátumhoz viszonyítva tudunk befizetett tagdíjat lekérni.

sum mező: típusa int

Ebben a mezőben tudja az admin beírni az éves tagdíj elvárt összegét. Jelenleg ez 14000 Ft-ra van meghatározva.

### 4. payments tábla

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra
<input type="checkbox"/>	1 id 🔑	bigint(20)		UNSIGNED	Nem	Nincs		AUTO_INCREMENT
<input type="checkbox"/>	2 user_id 🔑	bigint(20)		UNSIGNED	Nem	Nincs		
<input type="checkbox"/>	3 amount_paid	int(11)			Nem	Nincs		
<input type="checkbox"/>	4 payment_date	date			Nem	Nincs		
<input type="checkbox"/>	5 created_at	timestamp			Igen	NULL		
<input type="checkbox"/>	6 updated_at	timestamp			Igen	NULL		

7.ábra

A befizetések tartalmazó tábla az egyes kórustag által befizetett tagdíj összegét, és a befizetés dátumát, és memebers\_id-t tartalmazza, ami idegenkulcsot kapott.

id mező: típusa bigint, mező hossza 20, és auto\_increment-re állítva az automatikus érték növelés érdekében.

amount\_paid mező: típusa int

Ez a fizetett összeg mezőt tartalmazza azt, hogy a felhasználó (esetünkben a kórustag) mennyi összeget fizetett be.

payment\_date mező: típusa date

A befizetés dátuma mező pont azt takarja, ami a nevében benne is van, a felhasználó általi befizetés dátumát.

## 5. sheet\_musics tábla

#	Név	Típus	Illesztés	Tulajdonságok	Nulla	Alapértelmezett	Megjegyzések	Extra
<input type="checkbox"/> 1	id 🗝️	bigint(20)			Nem	Nincs		AUTO_INCREMENT
<input type="checkbox"/> 2	composer	varchar(100)	utf8mb4_general_ci		Nem	Nincs		
<input type="checkbox"/> 3	song_title	varchar(100)	utf8mb4_general_ci		Nem	Nincs		
<input type="checkbox"/> 4	sheet_music_pdf	varchar(50)	utf8mb4_general_ci		Nem	Nincs		

8. ábra

Ez a kotta tábla, melyben a kottákhoz tartozó zeneszerzőket, dalok címeit, magának a kottának a pdf-ben tárolt állományát tartalmazza.

id: típusa bigint, hossza 20, beállítva auto\_increment-re, így automatikusan növeli az értéket eggyel.

song\_title mező: típusa varchar, hossza 100, kódolása utf8mb4\_general\_ci

Ebben a mezőben a dalok címei tárolódnak.

composer: típusa varchar, hossza 50, kódolása utf8mb4\_general\_ci

A zeneszerzőket találhatóak ebben a mezőben.a

sheet\_music\_pdf mező: típusa binary

## iii. Kapcsolatok indoklása ismertetése

1. A users táblához (felhasználók tábla) közvetlenül csatlakozik az events tábla (események tábla).



Ha a felhasználó jelentkezik egy eseményre -ami lehet próba, vagy fellépés-, ez az elsődleges kulcsú felhasználó id-jával és az adott esemény id-jával van összeköttetésben, ez a kapcsolat itt jelenik meg.

2. Az events tábla (események tábla) kapcsolatban van a sheet\_music (kották) táblával. Itt az események táblán belül a sheet\_music\_id (kottaId) van összekapcsosva a sheet\_music (kották) tábla elsődleges id-jával. Ez azért fontos a későbbiek során, hogy pl. egy-egy próbára vagy fellépésre az adatbázisból le lehessen kérni a kívánt kottát.

3. A users tábla (felhasználók tábla) csatlakozik a payments táblához (befizetések tábla). A users táblában található elsődleges id lett összekötve a payments tábla user\_id-jával. A kapcsolódásnak az a célja, hogy le lehessen kérdezni a felhasználó (ami a befizetőt jelöli) befizetéseit.

4. A payments táblához (befizetések tábla) kapcsoltuk a membership\_fees táblát (tagdíj tábla) Meghatároztuk az éves tagdíjat, melyet az adatbázisban rögzítettünk, ez van megjelenítve a tagdíj táblában. ennek segítségével le tudjuk kérdezni van-e esetleg tartozása a kórustagnak.

#### d. Komponens/struktúra dokumentáció

A keretrendszer komponensekből épül fel, állapotokkal rendelkezik. Fontos, hogy ezek a rendszerek azonnal automatikusan lereagálják a változásokat. Ezt úgy tudja, lereagálni, hogy virtuális DOM-ot (Declarative Rendering) használ. Igazából az egész weboldal komponensek halmazából tevődik össze.

Komponensekből épül fel a nézet, ezért a létrehoztunk egy views mappát, ahol a megjelenített komponensek láthatóak.

Az alábbi view-kat (nézeteket) alkottuk meg:

Home.vue - A kezdő oldal, a főoldal, ami fogad minket az weboldalon.

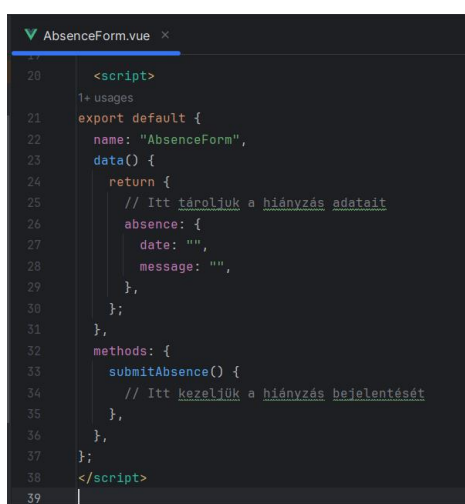
Login.vue - Ha már korábban regisztráltunk az oldalra, akkor a bejelentkezés oldalra léphetünk, ami a nézetünkben amolyan egyszerű kis "weboldalt" jelent.

Calendar.vue - ami tulajdonképpen egy tényleges naptárt tartalmaz.

MusicLibraryLink.vue - ami a kottatárat jelenti. Itt a kórustagok kereshetnek az úgynevezett "Kottatár"-ban, zeneszerzőre, dalcímre tudnak keresni.

News.vue - itt jelennek meg a kórust érintő aktuális hírek. Az aktuális dalversenyek, kórustalálkozók időpontjait, helyszíneit lehet itt megjeleníteni például.

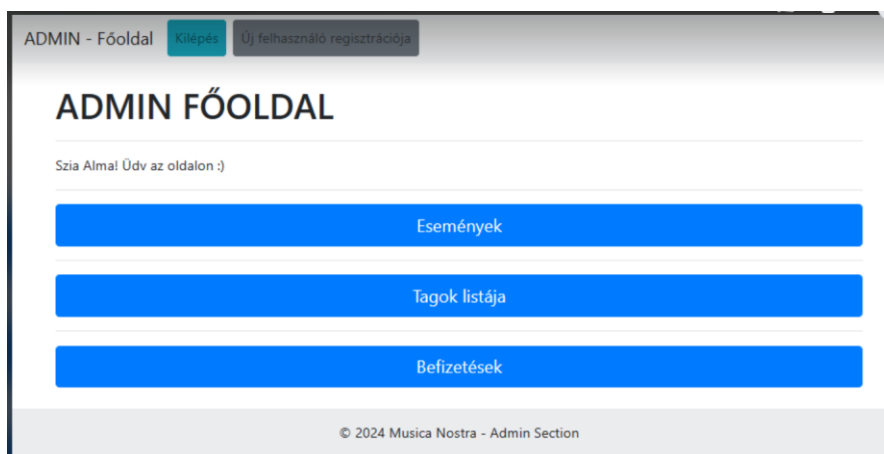
AbscenceForm.vue - Van egy Hiányzás bejelentő oldalunk, amelyet ebben a komponensben definiáltunk. Itt űrlapokat hoztunk létre, ahova a kórustag üzenetet írhat, ha nem megy egy próbára, és a Bejelentés gombra kattintva tudja elküldeni az üzenetét.



```
17
18
19
20 <script>
21 1+ usages
22 export default {
23   name: "AbscenceForm",
24   data() {
25     return {
26       // Itt tároljuk a hiányzás adatait
27       absence: {
28         date: "",
29         message: "",
30       },
31     };
32   },
33   methods: {
34     submitAbscence() {
35       // Itt kezeljük a hiányzás bejelentését
36     },
37   },
38 };
39 </script>
```

9. ábra

Az admin felület üdvözlő oldala vezet a bejelentkezéshez, ami után a főoldalra érkezünk. Innen érhetők el a különböző felületek. Az itt található navigációs sáv és gombok vezetnek a különböző nézetekhez, mint például az új felhasználó regisztrációja, az események kezelése, tagok listája és a befizetések kezelése:



10. ábra

#### i. Projekt elemei fejlesztés szempontjából

A MusicLibraryLink.vue-ban található "Kottatár" adatbázisa jelenleg csak lokálisan érhető el, a későbbiek folyamán ennek elérhetőségét meg kell oldani, például egy távolról is elérhető szerver által, vagy egy felhőben tárolt adatbázis útján.

Nézetekre, azért van szükség, hogy a router-t meg tudjuk írni, hogy a router működőképes legyen. Ahhoz, hogy használhatóak legyenek a komponensek, az index.js-be be kellett importálnunk a következőket:

```

1 //index.js
2 import { createRouter, createWebHistory } from 'vue-router';
3 import { isLoggedIn } from '@/utils/http';
4 import Home from '../views/Home.vue';
5 import Login from '../views/Login.vue';
6 import Registration from '../views/Registration.vue';
7 import Calendar from '../views/Calendar.vue';
8 import AbsenceForm from '../views/AbsenceForm.vue';
9 import MusicLibraryLink from '../views/MusicLibraryLink.vue';
10 import News from '../views/News.vue';

```

11. ábra

Így meg is érkeztek a Nézetek.

Úgy működik, hogy a createRouter fogja létrehozni magát a routert. A createRouter-t úgy hozom létre, hogy átadok neki egy objectet, hogy a history-ra vigye a createWebHistory-t, mint egy függvényt, és használja a routes-okat, amiket az array-ben létrehoztunk. Ez fogja irányítani, működtetni az egészet.



```

11
12 const router : Router = createRouter( options: {
13   history: createWebHistory(import.meta.env.BASE_URL),
14   routes: [
15     {path: '/'...},
19     {name: 'home'...},
25     {name: 'login'...},
31     {name: 'registration'...},
37     {name: 'calendar'...},
43     {name: 'absenceform'...},
49     {name: 'musiclibrarylink'...},
55     {name: 'news'...}
61

```

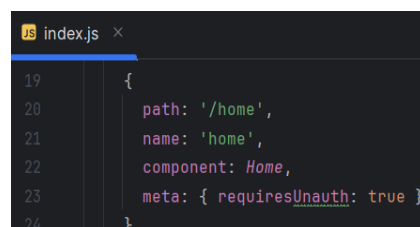
12. ábra

A router tömbnek az elemei object-ek. Ebbe vettük fel az útvonalakat, amiket szeretnénk használni, hogy a weboldalunk oldalai működjenek.

Megmondtuk, hogy mi a nevet, mi az elérési útvonala, és megmondtuk, hogy milyen komponenst töltsön be hozzá. Tehát létrehoztuk magát az objektumot a createRouterre, a router a createWebhistory-ból és a routes-ból

Itt az index.js-be adom meg, hogy mit töltsön be, melyik komponenst, megadom a nevét és az útvonalát, a path-t is.

Ha a főoldalra szeretném irányítani, akkor a következő path-re, névre, komponensre lesz szükség:



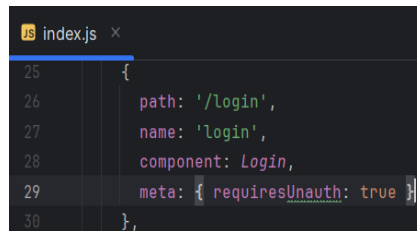
```

19 {
20   path: '/home',
21   name: 'home',
22   component: Home,
23   meta: { requiresUnauth: true }
24 },

```

13. ábra

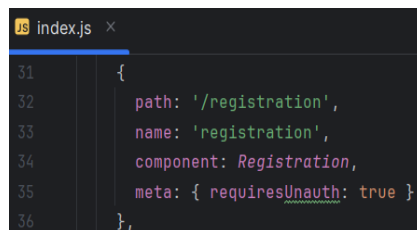
Ha a bejelentkező oldalt szeretném megjeleníteni, akkor a következők kellene az eléréshez, egy útvonalra, egy névre, és a hozzá tartozó komponensre:

A screenshot of a code editor showing the configuration for a login route in index.js. The code is as follows:

```
25 {  
26   path: '/login',  
27   name: 'login',  
28   component: Login,  
29   meta: { requiresUnauth: true }  
30 },
```

14. ábra

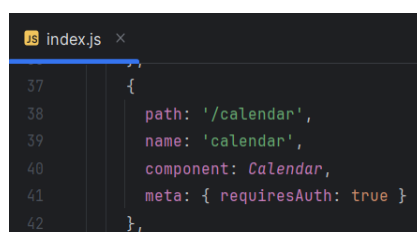
Amennyiben a regisztrációs oldalra szeretnék eljutni, akkor a következő útvonalat, nevet és komponens nevét kell behívnom.

A screenshot of a code editor showing the configuration for a registration route in index.js. The code is as follows:

```
31 {  
32   path: '/registration',  
33   name: 'registration',  
34   component: Registration,  
35   meta: { requiresUnauth: true }  
36 },
```

15. ábra


Ha a naptárra lennék kíváncsi, akkor az alábbiakra lesz szükségem beírom az alábbi path-t, nevet és komponens nevét:

A screenshot of a code editor showing the configuration for a calendar route in index.js. The code is as follows:

```
37 {  
38   path: '/calendar',  
39   name: 'calendar',  
40   component: Calendar,  
41   meta: { requiresAuth: true }  
42 },
```

16. ábra

A hiányzás bejelentő űrlapot szeretném kitölteni, akkor a következő path-t, nevet, komponens nevét kell megadnom:



```

43   {
44     path: '/absenceform',
45     name: 'absenceform',
46     component: AbsenceForm,
47     meta: { requiresAuth: true }
48   },

```

17. ábra

Ha "Kottatár"-ból szeretnék egy kották lekérni, akkor a következőképpen érem el a "Kottatár" oldalát:



```

49   {
50     path: '/musiclibrarylink',
51     name: 'musiclibrarylink',
52     component: MusicLibraryLink,
53     meta: { requiresAuth: true }
54   },

```

18. ábra

Ha a "Hírek"-re lennék kíváncsi akkora a következőképpen tudom elérni, az alábbi path-t, nevet és komponenst kell neki megadnom:



```

55   {
56     path: '/news',
57     name: 'news',
58     component: News,
59     meta: { requiresAuth: true }
60   }

```

19. ábra

Majd kiexportáltuk a router-t, ezért azt mondtuk az index.js-be:

```
export default router;
```

Így már létezik a router-em, ezt a router-t fogja tudni használni. A main.js-be ezt a router-t használni (use) kellett.

Le-createl-tettük az App-ot vele, és azt mondtuk, hogy app.mount és ezenkívül, hogy app.use(router), vagyis használja a routert. Amit már rögtön importált is, mert tudta, hogy van egy router-e, ezért beimportálta szépen. Innentől kezdve használni tudta ezt a router-t.

```

1  import { createApp } from 'vue'
2  import App from './App.vue'
3  import router from './router'
4  import 'bootstrap'
5  import 'bootstrap/dist/css/bootstrap.min.css'
6
7  const app : App<Element> = createApp(App)
8  app.use(router)
9  app.mount( rootContainer: '#app')
10

```

20. ábra

```

..... WelcomeController@index
POST    api/login ..... AuthController@apiLogin
GET|HEAD api/sheetmusic ..... sheetmusic > SheetMusicController@index
GET|HEAD api/userevent ..... userevent > UserEventController@index
GET|HEAD api/users ..... UserController@getAllUsers
GET|HEAD api/users/{id} ..... UserController@getUser

Showing [6] routes

```

21. ábra

## 1. Osztályok

A backend oldalon az AuthController. php-ban egy Laravel controller osztályt definiáltunk, amely tartalmazza az autentikációs funkciókat.

```

1  <?php
2  // AuthController.php
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\User;
7  use Illuminate\Support\Facades\Hash;
8  use JWTAuth;
9  use Tymon\JWTAuth\Exceptions\JWTException;
10
11 class AuthController extends Controller
12 {
13     public function register(Request $request)
14     {
15         $request->validate([
16             'username' => 'required|string|max:255',
17             'email' => 'required|string|email|max:255|unique:users',
18             'password' => 'required|string|min:8',
19         ]);
20
21         $user = User::create([
22             'username' => $request->username,
23             'email' => $request->email,
24             'password' => Hash::make($request->password),
25         ]);
26
27         return response()->json(['user' => $user], status: 201);
28     }
29 }

```

22. ábra

A 'register' metódus felelős az új felhasználók regisztrációjáért. Először validálja a beérkezőkérést a Laravel beépített validációs szabályok segítségével. Ha a validáció sikeres, létrehozza az új felhasználót a 'User' modell 'create' metódusával.

A jelszót a 'Hash::make' függvénnyel titkosítja, mielőtt a felhasználót mentené az adatbázisba. Végül a metódus visszatér egy JSON válasszal, amely tartalmazza az új felhasználót, valamint az 'HTTP 201 Created', tehát sikeres státuszkódot.

A 'login' metódus felelős a felhasználó bejelentkezésért. először validálja a beérkező kérést, majd megpróbálja létrehozni a JWT token a 'JWTAuth::attempt' függvénnyel. Ha a bejelentkezés sikertelen, a metódus visszatér egy JSON válasszal, ami jelzi az érvénytelen hitelesítő adatokat ('HTTP 401 Unauthorized'). Ha valamilyen hiba történik a token létrehozásakor, akkor a metódus egy 'HTTP 500 Internal Server Error' státuszkódot ad vissza.

```
public function login(Request $request)
{
    $credentials = $request->validate([
        'username' => 'required|string',
        'password' => 'required|string',
    ]);

    try {
        if (!$token = JWTAuth::attempt($credentials)) {
            return response()->json(['message' => 'Invalid credentials'], status: 401);
        }
    } catch (JWTException $e) {
        return response()->json(['message' => 'Could not create token'], status: 500);
    }

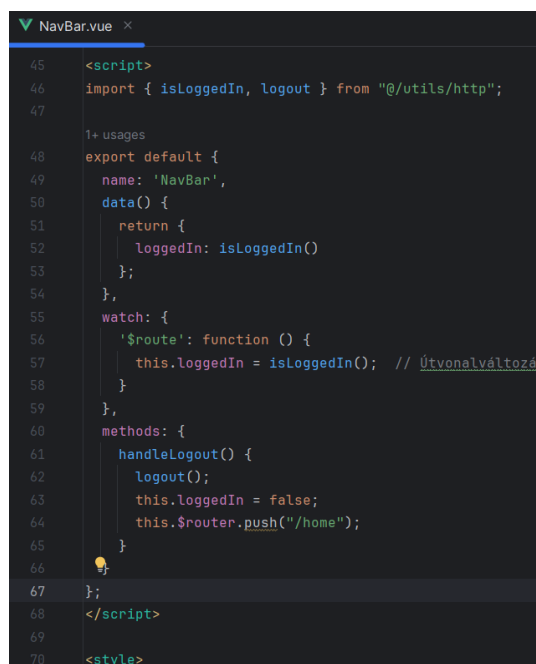
    return response()->json(compact('token'));
}
```

23. ábra

Mindkét metódus JSON választ ad vissza, ami alkalmas az API használatára. Az 'AuthController' használható a regisztrációs és a bejelentkezési folyamatok kezelésére egy Laravel alapú API-ban, esetünkben ezért használtuk.



## 2. Kapcsolatok



```
45 <script>
46 import { isLoggedIn, logout } from "@/utils/http";
47
48 1+ usages
49 export default {
50   name: 'NavBar',
51   data() {
52     return {
53       loggedIn: isLoggedIn()
54     };
55   },
56   watch: {
57     '$route': function () {
58       this.loggedIn = isLoggedIn(); // Útvonalváltás
59     }
60   },
61   methods: {
62     handleLogout() {
63       logout();
64       this.loggedIn = false;
65       this.$router.push("/home");
66     }
67   };
68 </script>
69
70 <style>
```

24. ábra

Az előző kódrészlet kezeli a bejelentkezést és a kijelentkezést frontend oldalon a navigációs sávban.

Az 'import' sor az 'isLoggedIn' és 'logout' függvényeket importálja a "@/utils/http" modulból. Ezek a függvények az autentikációhoz, ill. a felhasználói munkamenetek kezeléséhez kapcsolódó segédmetódusok.

A 'name' tulajdonság határozza meg a komponens adattagjait. Ebben az esetben a 'loggedIn' változó inicializálódik az 'isLoggedIn()' függvény visszatérési értékével, ami jelzi, hogy a felhasználó be van-e jelentkezve, avagy nincs.

A 'watch' blokk figyelni az útvonal ('\$route') változásait. Ha az útvonal változik akkor frissíti a 'loggedIn' állapotot a 'isLoggedIn()' függvény visszatérési értékével.

A 'methods' részben a 'handleLogout' metódus definiálása kezeli a kijelentkezést. Először meghívja a 'logout()' függvényt, majd a 'loggedIn' változót hamisra állítja, és végül átirányítja a felhasználót a '/home' útvonalra a 'this.\$router.push("/home")' segítségével.

### ii. Főbb elemek részletes ismertetése

User.php-ban egy Laravel modellt definiáltunk, amely a felhasználókat jeleníti meg az alkalmazásban.

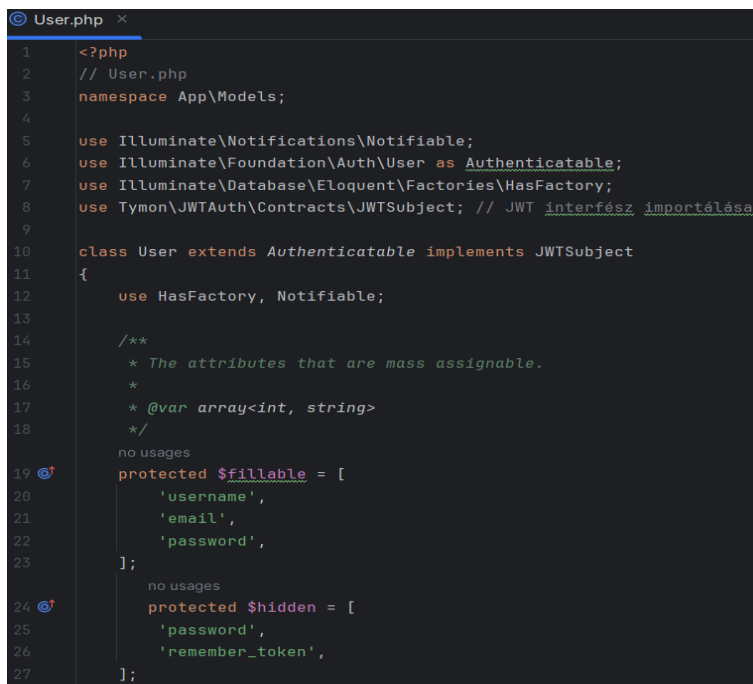
A 'User' osztály az 'Authenticatable' alapsztályt kiterjeszti, ami lehetővé teszi az alapvető hitelesítési funkciók használatát a felhasználó modellben.

Az 'Authenticatable' osztály a Laravel által biztosított alapértelmezett felhasználómodell.

A 'HasFactory' használatával a modell létrehozási funkciókkal rendelkezik, ami lehetővé teszi a ezeknek a használatát a modell teszteléséhez.

A 'Notifiable' segítségével az alkalmazás értesítéseket küldhet e-mailen keresztül a felhasználóknak. Az e-mail címeket regisztrációkor adják meg a felhasználók.

A '\$fillable' tömbben soroljuk fel azokat az attribútumokat, amelyeket tömeges töltéskor (pl. 'create()' vagy 'update()') engedélyezünk. Ezek a mezők biztonságosan beállíthatók a modellben.



```
1 <?php
2 // User.php
3 namespace App\Models;
4
5 use Illuminate\Notifications\Notifiable;
6 use Illuminate\Foundation\Auth\User as Authenticatable;
7 use Illuminate\Database\Eloquent\Factories\HasFactory;
8 use Tymon\JWTAuth\Contracts\JWTSubject; // JWT interfész importálása
9
10 class User extends Authenticatable implements JWTSubject
11 {
12     use HasFactory, Notifiable;
13
14     /**
15      * The attributes that are mass assignable.
16      *
17      * @var array<int, string>
18      */
19     no usages
20     protected $fillable = [
21         'username',
22         'email',
23         'password',
24     ];
25
26     no usages
27     protected $hidden = [
28         'password',
29         'remember_token',
30     ];
31 }
```

25. ábra

A '\$hidden' tömbben felsoroljuk azokat az attribútumokat, amelyeket nem szeretnénk megjeleníteni a felhasználók számára, például a jelszót.

### 3. Pl.: Controller/BusinessLogic Dokumentáció

A Composition API egy komponenseket leíró API, amely a VUE.js 3-hoz készült. Először a 3.0-ás Vue.js-hez jelent meg maga ez a Composition API. Valójában ez fut az Option API alatt

is (a háttérben). A kettő egyszerre fut egymás mellett és a Lifecycle-nek lesz fontos szerepe. Az alábbiakat tartalmazza: Reactivity API, Lifecycle hooks, Dependency Injection. Tulajdonságokat tartalmaz, értékeket tárol.

A Composition API elengedi ezt az oblijeg megközeleítést és azt mondja, hogy ami eddig objektum volt a data, a watch, a components stb. ő azt mondja, hogy innentől kezdve speciális függvényben valósítja meg, és innentől kezdve tudunk szabadabb kódrendszereket alkotni. Tudunk akár olyan kódrendszert alkotni, aminek nincs semmi megjelenítője, csak használni tudom bizonyos metódusait bizonyos helyeken.

Itt már ki tudunk emelni logikát, tudok olyan kódrészleteket alkotni, aminek nem kell, hogy legyen megjelenítője, hogy működni tudjon. Úgyhogy újrafelhasználhatóvá tudom tenni a kódomat és több különböző függvényt tudok használni különböző helyen. A nagyobb projektekhez és az összetett komponensekhez ajánlott ezt használni. Ez azt jelenti, hogy ha már mélyebben benne vagyunk a webfejlesztés világában és valamilyen weboldalt csinálunk egy nagy cégnek, akkor érdemesebb a Composition API-t használni, mert míg az Option API szigorú rendszerben tart, addig a Composition API azt teszi amit mi mondunk neki. Elengedi a kezemet, mindent én írok meg, mindent úgy határozok meg, ahogy én akarom.

Egyszerre tudjuk az Option API-t és a Composition API-t használni. Az export default maga az Option API. Az Option API-n belül van egy data(), és a Composition API-t úgy tudom hívni, hogy meghívom a setup() függvényt, és amit a setup() függvénybe írok, az Composition API, amit a data-ba írok, az meg az Option API, mert ugye ez egy zárt rendszer. Mint ahogy ez látszik, ez a két API egymás mellett teljesen működőképes és nincs nagyon különbség a kettő között, csak majd egy picit érdekesebb dolgokra képes.

Amikor használtuk a data-ban valamilyen változót, ő az állapotváltozásokra le tudta követni. Beírt valamit a felhasználó és a felhasználó által megadott dolgokra reagálva le tudta követni az Option API. A cél a reaktivitás, hogy anélkül, hogy hozzá kellene nyúlni, az állapotváltozásokat lekezelje. Az export default az Option API része.

Ahhoz, hogy ez az egész működjön, és a template-ben el tudjam érni magát az értékeket és a benne lévő tartalmakat, ahhoz egy objekt-ként ki kellett return-ölnöm azt, amire szükségem volt.

A regisztrációs hibát egy "alert"-ben, felugró ablakban jelenítjük meg a felhasználó számára, aminek a kódját egy methods-ban lett megírva, a Registration.vue-ban:

```

55 },
56 methods: {
57   async register() {
58     try {
59       const response = await axios.post( url: "/register", this.formData);
60       console.log("Regisztrációs válasz:", response.data);
61       if (response.data.success) {
62         this.$router.push("/login");
63       } else {
64         console.error("Regisztrációs hiba: ", response.data.message);
65         alert("Regisztrációs hiba: " + response.data.message);
66       }
67     } catch (error) {
68       console.error("Regisztrációs kivétel: ", error.response);
69       alert(
70         "Hiba történt: " + (error.response.data.message || "Ismeretlen hiba")
71       );

```

26. ábra

e. API dokumentáció

i. Végpontok

1. Útvonalak

```

<ul class="navbar-nav">
  <li class="nav-item" v-show="!loggedIn">
    <router-link class="nav-link" to="/registration">Regisztráció</router-link>
  </li>
  <li class="nav-item" v-show="!loggedIn">
    <router-link class="nav-link" to="/login">Bejelentkezés</router-link>
  </li>
  <li class="nav-item" v-if="loggedIn">
    <router-link class="nav-link" to="/calendar">Naptár</router-link>
  </li>
  <li class="nav-item" v-if="loggedIn">
    <router-link class="nav-link" to="/absenceform">Hiányzásbejelentő</router-link>
  </li>
  <li class="nav-item" v-if="loggedIn">
    <router-link class="nav-link" to="/musiclibrarylink">Kottatár</router-link>
  </li>
  <li class="nav-item" v-if="loggedIn">
    <router-link class="nav-link" to="/news">Hírek</router-link>
  </li>
</ul>

```

27. ábra

A frontend oldalon lévő layout mappában található NavBar.vue komponensbe létrehoztunk `<router-link>` -eket, aminek a segítségével a különböző oldalakhoz való navigálást oldottuk meg. A nav-link kapott egy osztályt: `class= "nav-link"`.

A `"/registration"`-ra mutató `<router-link>` a Regisztráció oldalára vezet.

A `"/login"`-ra mutató `<router-link>` a Bejelentkezés oldalára vezet.

A `"/calendar"`-ra mutató `<router-link>` a Naptár oldalára vezet.

Az `"/absenceform"`-ra mutató `<router-link>` a Hiányzásbejelentő oldalára vezet.

A `"/musiclibrarylink"`-ra mutató `<router-link>` a Kottatár oldalára vezet.

A `"/news"`-ra mutató `<router-link>` a Hírek oldalára vezet.

Ezt a `<router-linket>` így még nem fogom elérni, importálnom kell az `utils` mappában lévő `http.js` file-ból. Innentől kezdve tudja, hogy ez a `<router-link>` létezik.

A `<router-link>` -ben a `/registration` (per jel) utáni hivatkozást megnézi, hogy az `index.js`-ben lévő mire hivatkozik a `/registration` (per jel) után, és a hozzá tartozó `Registration` komponenst betölti majd, ami a Regisztrációs oldal.

A `<router-link>` -ben a `/login` (per jel) utáni hivatkozást megnézi, hogy az `index.js`-ben lévő mire hivatkozik a `/login` (per jel) után, és a hozzá tartozó `Login` komponenst betölti majd. Ez a Bejelentkezés laphoz vezet minket.

A `<router-link>` -ben a `/calendar` (per jel) utáni hivatkozást megnézi, hogy az `index.js`-ben lévő mire hivatkozik a `/calendar` (per jel) után, és a hozzá tartozó `Calendar` komponenst betölti majd. Így jutunk a Naptár oldalára.

A `<router-link>` -ben a `/absenceform` (per jel) utáni hivatkozást megnézi, hogy az `index.js`-ben lévő mire hivatkozik a `/absenceform` (per jel) után, és a hozzá tartozó komponenst betölti majd. Így érhető el majd a Hiányzásbejelentő oldala.

A `<router-link>` -ben a `/musiclibrarylink` (per jel) utáni hivatkozást megnézi, hogy az `index.js`-ben lévő mire hivatkozik a `/musiclibrarylink` (per jel) után, és a hozzá tartozó `MusicLibraryLink` komponenst betölti majd. Ennek segítségével találhatjuk meg a Kottatár oldalát.

A `<router-link>` -ben a `/news` (per jel) utáni hivatkozást megnézi, hogy az `index.js`-ben lévő mire hivatkozik a `/news` (per jel) után, és a hozzá tartozó `News` komponenst betölti majd. Evvel megérkezhetünk a Hírek oldalára.

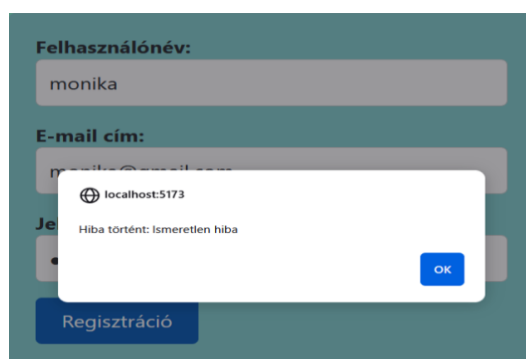
2. Paraméteres kérések lehetőségei

3. Válasz.

#### f. Teszt dokumentáció

A frontend és a backend oldalon is végeztünk tesztek a projekt program írása során. A [www.postmen.com](http://www.postmen.com) oldalát használtuk az egyes elkészült részek tesztelésére.

Hibás, rosszul megadott jelszónál egy felugró ablakban jelzi is a rendszer, hogy "Hiba történt".



28. ábra

Bejelentkezéskori téves név vagy jelszó megadásánál is hibaüzenettel jelez a program.

## 4. Továbbfejlesztési lehetőségek

Számos további ötlet merült fel, amik további fejlesztési lehetőséget jelentenek, de idő és még tudás hiányában ez még nem tudott megvalósulni, de nyitva maradt legalább ennek lehetősége, és tudjuk, hogy az ötlet, az elképzelésünk a kórus életében sok pozitívummal kecsegtet.

Fontos fejlesztés lenne, ha a felhasználó által beírt név és jelszó párosítás már létezik, azt jelezze a program. Biztonságot adna, ha regisztráláskor egy visszaigazoló e-mail érkeznek az újonnan regisztrált e-mail fiókjába.

Felmerült az igény a távolabbi jövőre nézve egy chat ablak létrehozása, ahol a kórustagok egymásnak tudnának üzenni.

## 5. Összegzés

A téma kiválasztásánál még nem ismertük se a Vue.js-t, se a PHP-t, Laravel-t, így mondhatni vakon találtunk ki egy valós életben is használható projektet. Csak az oldal kinézetéről és kórus számára az esetleges igényekről volt némi halvány elképzelésünk. A megvalósítás hatalmas kérdőjelként lebegett előttünk. Ahogy teltek a hetek és haladtunk a tananyaggal, még mindig nem állt össze, hogy is kellene létrehozni a kigondolt projektet. Sajnos már az elején nehézségbe ütköztünk a user és az admin bejelentkezés backend és frontend oldali összekapcsolásával.

## 6. Források, ábrajegyzék

Források:

[www.mysql.com](http://www.mysql.com)

[vuesj.org](http://vuesj.org)

[getbootstrap.com](http://getbootstrap.com)

[laravel.com](http://laravel.com)

[www.w3schools.com](http://www.w3schools.com)

Ábrajegyzék:

1. ábra - Regisztráció ablaka
2. ábra - Bejelentkezés ablaka
3. ábra - Dia program
4. ábra - users tábla
5. ábra - events tábla
6. ábra - membership\_fees tábla
7. ábra - payments tábla
8. ábra - sheet\_musics tábla
9. ábra - AbscenceForm.vue - Hiányzás bejelentő kód
10. ábra - index.js - importálандók
11. ábra - index.js - createRouter
12. ábra - index.js - /home kódrészlet
13. ábra - index.js - /login kódrészlet

- 14. ábra - index.js - /registration kódrészlet
- 15. ábra - index.js - /calendar kódrészlet
- 16. ábra - index.js - /absenceform kódrészlet
- 17. ábra - index.js - /musiclibrarylink kódrészlet
- 18. ábra - index.js - /news kódrészlet
- 19. ábra - main.js -createapp
- 20. ábra - AuthController.php -register
- 21. ábra - AuthController.php - login
- 22. ábra - NavBar - script
- 23. ábra - User osztály
- 24. ábra - Registration.vue - regisztrációs hiba - kód
- 25. ábra - Útvonalak, router-linkek
- 26. ábra - Regisztráció - tesztelés, hibaüzenet
- 27. ábra - Útvonalak
- 28. ábra – Hibaüzenet