



Jelenlét követő alkalmazás fejlesztése

Készítette

Györkis Tamás

Programtervező informatikus BSc

Témavezető

Dr. Király Roland

Egyetemi docens

EGER, 2024

Tartalomjegyzék

Bevezetés	4
1. Alkalmazás bemutatása	5
1.1. Adatbázis	5
1.2. Általános információk	6
1.2.1. Felhasználók	6
1.2.2. Általános funkciók	7
1.3. Adminisztrátori nézet, funkciók	8
1.3.1. Felhasználók kezelése	8
1.3.2. Alkalmazás konfiguráció	8
1.3.3. Félévek, termék, tantárgyak, kurzusok, órák kezelése	9
1.4. Tanári nézet, funkciók	10
1.5. Hallgatói nézet, funkciók	10
2. Felhasznált technológiák, csomagok	11
2.1. Laravel	12
2.1.1. Keretrendszer alapjai	12
2.1.2. MVC architektúra	12
2.1.3. Eloquent ORM	12
2.1.4. Blade templating engine	12
2.1.5. Laravel Fortify	12
2.1.6. Események, queue	12
2.1.7. Email küldés	12
2.2. Tailwind	12
2.2.1. DaisyUI	12
2.3. LiveWire	12
2.3.1. Ismertető, a csomag működése	12
2.3.2. LiveWire és SPA	12
2.3.3. Komponensek és data binding	12
2.3.4. Események, lapozás	12
2.3.5. Alpine.js: LiveWire és JavaScript kapcsolata	12
2.3.6. WireToast: LiveWire alapú értesítések	12

2.4.	Websocket és Pusher szerviz	12
2.4.1.	Websocket jelentősége	12
2.4.2.	Pusher	12
2.4.3.	Websocket integrálása a keretrendszerbe	12
2.4.4.	Csatornák, események létrehozása	12
2.4.5.	Események fogadása JavaScript, LiveWire esetén	12
2.5.	QR-kód generálás	12
2.6.	Naptár - FullCalendar.js	12
2.6.1.	Beállításai	12
2.6.2.	Kapcsolat a keretrendszerrel LiveWire segítségével	12
2.7.	Diagramok - Chart.js	12
2.7.1.	Beállításai	12
2.7.2.	Kapcsolat a keretrendszerrel LiveWire segítségével	12
2.8.	Progressive Web Apps	12
2.8.1.	A PWA jelentése, jelentősége	12
2.8.2.	Laravel PWA csomag	12
2.8.3.	A manifest fájl	12
2.8.4.	Service Worker	12
3.	Az alkalmazás tesztelése	13
3.1.	Manuális tesztelés	13
3.2.	Automatizált tesztelés	13
3.3.	Terheléses tesztelés	13
3.4.	Laravel Pint és Github actions	13
4.	Alkalmazás telepítése	14
4.1.	Laravel Forge	14
4.2.	Kézi telepítés lépései	14
	Összegzés	15
	Irodalomjegyzék	16

Bevezetés

Amikor szakdolgozati témaválasztás előtt álltam, sokat gondolkoztam a témán. Mindenképpen egy olyan alkalmazást szerettem volna elkészíteni, mely ténylegesen hasznos is lehet. Egyetemi éveim alatt demonstrátorként tanítottam több féléven át az egyetemen, és innen jött a felismerés, hogy jó lenne, ha a hallgatók jelenlétét, hiányzásait ne táblázatokban kelljen vezetni, hanem egy külön eszköz legyen rá készítve. Innen származik az alkalmazás ötlete.

A megvalósítás során törekedtem arra, hogy egy jól átlátható, ergonomikus weboldalt készítsek, amit – esetleges kisebb módosításokkal – ne csak egyetemi környezetben lehessen használni. Ezekből kifolyólag egy webalkalmazást készítettem, mivel ezt a leg-egyszerűbb elérni, hiszen csak egy webböngésző szükséges hozzá. A tanulók és tanárok által használt oldalak reszponzívan lettek elkészítve, ezáltal biztosítva, hogy mobiltelefonon is használni lehessen. Illetve az alkalmazás PWA funkciókkal rendelkezik, aminek jelentőségére később térek ki, de előjáróban annyit érdemes megemlíteni róla, hogy lehetővé teszi a weboldal alkalmazásként való telepítését számítógép és telefon esetén is, ezáltal egy rendes alkalmazás érzését keltve.

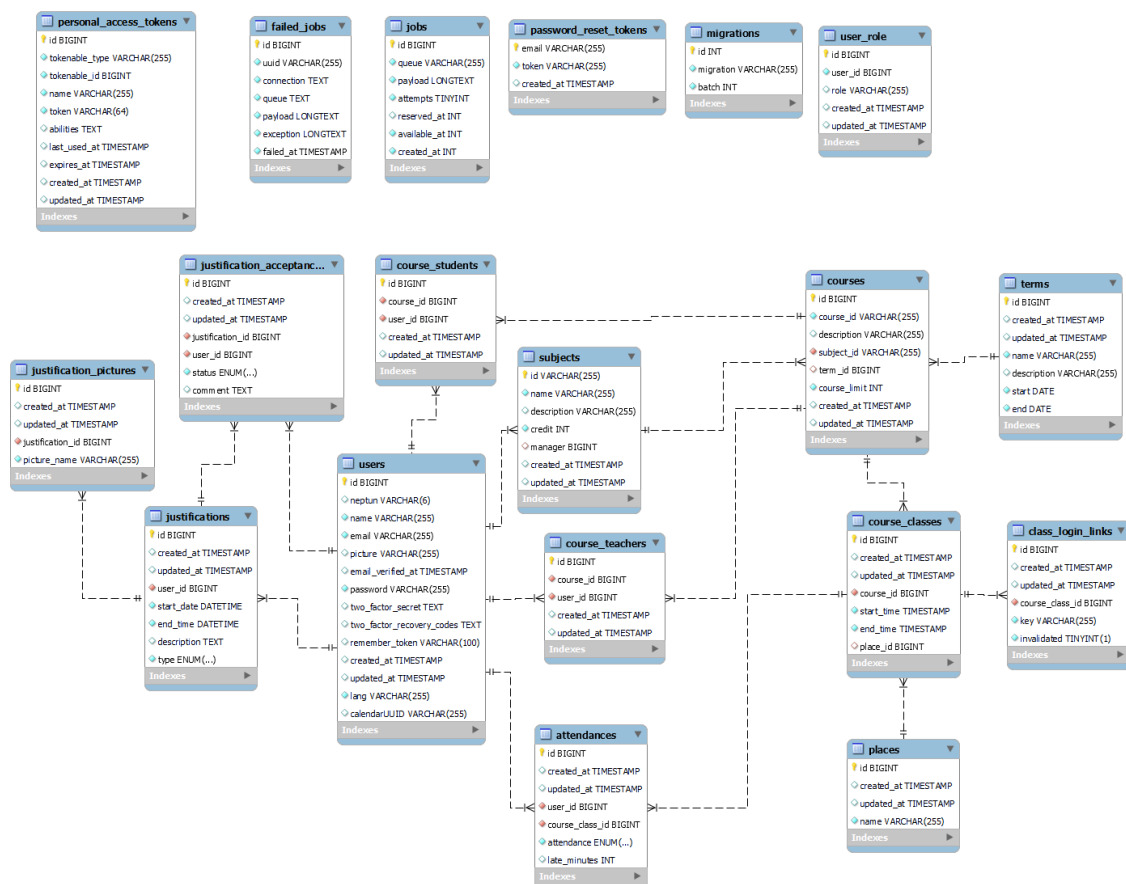
A megvalósításhoz a Laravel keretrendszert használtam, mely egy PHP alapú, MVC keretrendszer, amivel tanulmányaim során találkoztam, és egyből megkedveltem. Nem titkolt célom a szakdolgozatommal, hogy bemutassam, hogy a mai, JavaScript preferált világban, továbbra is lehetséges modern, a mai kort kielégítő weboldalt készíteni PHP segítségével, amihez különböző csomagokat alkalmaztam, amiknek működését, illetve egymással való működését a későbbiekben fogom kifejteni.

1. fejezet

Alkalmazás bemutatása

1.1. Adatbázis

Az 1.1 ábrán látható az alkalmazás által használt adatbázis, egyed-kapcsolat diagrammal ábrázolva, melyről szeretnék néhány szót ejteni.



1.1. ábra. Az alkalmazás által használt adatbázis.

- *users*: ez a tábla tárolja a felhasználók adatait. Minden felhasználóról tárolunk

egy azonosítót, egy Neptun[1] kódot¹, nevet, email címet, egy titkosított jelszót, egy profilképet, a felhasználó által használt nyelvet, illetve egy UUID-t, azaz egy egyedi azonosítót, amit az órarend exportálásához lehet használni, erről később. Az azonosító oszlopra azért van szükség, mivel az alkalmazás fel van készítve arra, hogy Neptun kód nélkül rendelkező felhasználókat is tudjon kezelni.

- *terms, places*: a félévek és termek (helyek) tárolására szolgáló táblák.
- *subjects, courses, course_classe, class_login_links*: ezen táblák tárolják a tantárgyakat (amik többek között rendelkeznek egy azonosítóval, névvel, leírással, kreditértékkel, és egy tantárgyfelelőssel), a tantárgyakhoz tartozó kurzusokat (amihhez tárolok egy azonosítót, ami minden esetben egyedi, egy kurzus azonosítót, ami minden félév és tantárgy esetében kell egyedinek lennie, tartozik hozzá egy félév, illetve egy létszám limit), illetve kurzusokhoz pedig órák (amiknek van egy kezdete, vége, illetve egy terem, ahol tartják). Az utolsó tábla pedig az órákhoz tartozó bejelentkező linkeket tartalmazza, ennek jelentőségéről kicsit később.
- *attendances*: itt tárolom az egyes kurzusok résztvevőinek az órai jelenléteit, melyek lehetnek: nincs kitöltve, jelen, késés (mely esetben percben lehet tárolni a mértékét), hiányzás, igazolt hiányzás.
- *justification, justification_acceptances, justification_pictures*: ezek a táblák tárolják az igazolásokat, az igazoláshoz tartozó feltöltött képeket, illetve az igazolásban érintett tanárok válaszát, hogy elfogadják-e ó, vagy sem.
- Az egyéb, nem említett táblák inkább kapcsolótábla funkciót töltenek be, vagy a keretrendszernek vannak rá szükségei. Például a *jobs* tábla a háttérben végrehajtandó feladatokat (job) tartalmazza.

Itt érdemes még megemlíteni, hogy a fejlesztés során a MySQL nevezetű, relációs adatbázist használtam, viszont a Laravel keretrendszerből adódóan sok más típusú adatbázis szoftverrel használható az alkalmazás, mivel a táblák felépítése a keretrendszer nyújtotta módon van elkészítve.

1.2. Általános információk

1.2.1. Felhasználók

Az alkalmazás 4 különböző jogkört különböztet meg a felhasználók esetén, melyekből egyszerre többet is birtokolhat a felhasználó:

¹ A neptun kód egy 6 karakter hosszú, egyedi azonosító.

- Szuper adminisztrátor: képes a felhasználók adatainak – és jogköreinek – szerkesztésére, illetve az alkalmazás alapbeállításainak módosítására.
- Adminisztrátor: a félévek, termek, tantárgyak, kurzusok, órák, készítésére, módosítására, törlésére jogosult.
- Tanár: megtekintheti a tanított óráit, kurzusait, hallgatóit. Adminisztrálni tudja az órákon való részvételt, illetve megtekintheti és bírálhatja a hozzá érkezett igazolásokat.
- Hallgató: a hozzárendelt óráit, tantárgyait látja, megtekintheti minden kurzus esetén az egyes órák státuszát, illetve igazolásokat hozhat létre

Az alkalmazás természetesen rendelkezik bejelentkezés, regisztráció funkciókkal. A felhasználók Neptun[1] kódjuk vagy email címük, illetve a jelszavuk megadásával tudnak bejelentkezni. Regisztrálni csak abban az esetben tudnak, ha az oldal beállításai-
ban ezt engedélyezték, egyéb esetben a szuper adminisztrátorok tudnak felhasználókat létrehozni vagy importálni. Belépés után a profil szerkeszthető, a jelszó cserélhető.

1.2.2. Általános funkciók

Bejelentkezés után a főoldalon a felhasználók pár, számukra fontos vagy érdekes információt láthatnak. Tanulók esetében a mai napi óráikat, illetve adminisztrálásra váró igazolásaikat. Tanárok esetén szintén a mai óráikat, illetve a kapott igazolásaikat láthatják. Az adminisztrátorok pár statisztikát láthatnak az oldal kapcsán.

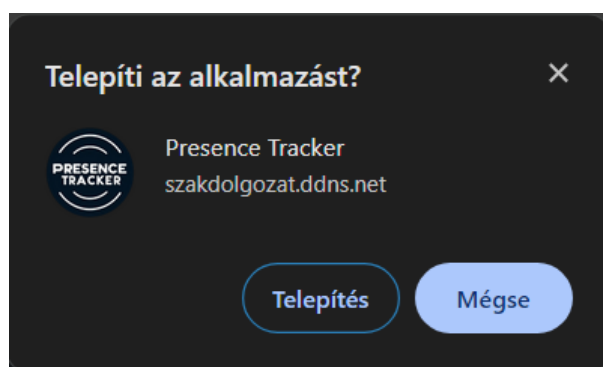


1.2. ábra. A főoldal kinézete abban az esetben, ha a felhasználó minden szerepkörrel rendelkezik. A két bal oldali oszlop a hallgatói nézethez tartozik, a két jobb oldali pedig az oktatóihoz, míg a statisztika adminisztrátorok esetén jelenik meg.

A profil kép melletti gombra kattintva a felhasználó meg tudja változtatni az oldal nyelvét, Angol és Magyar nyelv közül választva, illetve az oldalt kinézetét: világos és sötét mód között váltogatva. A nyelv beállítása eltárolódik a felhasználóhoz, így újabb bejelentkezés esetén automatikusan beállításra kerül.

Tanuló vagy tanár esetén megjelenik az *Órarend* menüpont is, ahol egy helyen láthatják a heti óráikat a felhasználók. Az egyes órák különböző színekkel jelennek meg, ezzel jelezve, hogy az adott órát a felhasználó tanítja-e, vagy hallgató esetén az órán jelen volt, hiányzott, késett, igazoltan hiányzott, vagy még nincs kitöltve a jelenlét. Lehetőség van az órarend exportálásra is, mely során egy *.ics* kiterjesztésű fájl áll elő. Ez egy széles körben használt általános naptár formátum, amit a felhasználók be tudnak importálni a gyakran használt naptár programokba[2]. Ezáltal kedvenc naptár alkalmazásukban is nyomon tudják követni óráikat.

Lehetőség van a weboldalt alkalmazásként telepíteni, ahogy az az 1.3 ábrán is látható, így gyorsan hozzáférhetnek az alkalmazás funkcióihoz, egy „rendes” alkalmazás érzését keltve ezzel. Ezt a PWA² teszi lehetővé, melynek működésére későbbiekben térek ki.



1.3. ábra. Az oldal telepítésére szolgáló felugró ablak.

1.3. Adminisztrátori nézet, funkciók

1.3.1. Felhasználók kezelése

Szuper adminisztrátor esetén a felhasználó képes kezelni az alkalmazásban tárolt felhasználókat, kivéve saját magát (személyes adatait továbbra is meg tudja változtatni). A listában képes szűrni a felhasználók között, megváltoztatni profil adataikat, jelszavukat alaphelyzetbe állítani, szerkeszteni jogosultsági szintjeiket, illetve akár törölni is őket, ha szükséges. Tud új felhasználókat létrehozni, illetve képes fájlból is importálni adatokat.

1.3.2. Alkalmazás konfiguráció

Szintén a szuper adminisztrátor körébe tartozik az oldal beállításainak kezelése. Az alábbi beállításokat tudja kezelni:

² Jelentése: Progressive Web App: egy olyan alkalmazás, ami webes technológiákat használ, de platform specifikus alkalmazásként viselkedik.[3]

- Oldal neve: az oldalon megjelenő név megváltoztatása.
- Regisztráció engedélyezése: ezzel lehet engedélyezni, hogy a felhasználók maguktól is tudjanak-e felhasználói fiókot készíteni az oldalon.
- Saját Neptun[1] kód megváltoztatása: beállítható, hogy a felhasználó meg tudja magának változtatni ezt az értéket, vagy sem.
- Kötelező Neptun kód: az oldal használható ezen kód megadása nélkül is, ez itt szabályozható.
- Alkalmazás képe: itt tölthető fel új kép, ami a fő oldalon jelenik meg látogatók esetén.

1.3.3. Félévek, termek, tantárgyak, kurzusok, órák kezelése

Minden adminisztrátor képes ezen adatok kezelésére.

Félévek esetén egy nevet, illetve egy kezdő és vég dátumot szükséges megadni. Ellenőrizve van, hogy a félévek nem ütközhetnek egymással. A listában egy pipa jelzi a jelenlegi félévet, ha van ilyen.

A termeknél elegendő egy nevet megadni, ezeket lehet az egyes órákhoz hozzárendelni.

Tantárgyak esetén egy tárgy kódot, egy nevet, egy opcionális leírást, egy kredit értéket, illetve egy tantárgyfelelőst tárol, akinek rálátása van a tárgy összes kurzusára. Ezen belül minden kurzus egy tárgyhoz kapcsolódik. Ezek szintén rendelkeznek egy kóddal, ami minden tárgy és félév esetén egyedi, szintén egy opcionális leírással, illetve nulla, egy, vagy több tanárral. Minden kurzus egy félévhez van rendelve. A kurzusokhoz hozzárendelhetők a hallgatók, és csak hallgatók.

A kurzusokhoz órák hozhatóak létre, amik a féléven belül lehetnek. Minden óra külön kezdés és vég időponttal rendelkezik, illetve egy teremmel, ahol tartják. Az óra létrehozásakor lehetőség van arra is, hogy a félév végéig heti ismétléssel hozza létre az órákat – természetesen később egyenként lehet őket szerkeszteni, törölni –, ezáltal megkönnyítve az adminisztrációt.

1.4. ábra. Új óra hozzáadása.

1.4. Tanári nézet, funkciók

1.5. Hallgatói nézet, funkciók

2. fejezet

Felhasznált technológiák, csomagok

2.1. Laravel

2.1.1. Keretrendszer alapjai

2.1.2. MVC architektúra

2.1.3. Eloquent ORM

2.1.4. Blade templating engine

2.1.5. Laravel Fortify

2.1.6. Események, queue

2.1.7. Email küldés

2.2. Tailwind

2.2.1. DaisyUI

2.3. LiveWire

2.3.1. Ismertető, a csomag működése

2.3.2. LiveWire és SPA

2.3.3. Komponensek és data binding

2.3.4. Események, lapozás

2.3.5. Alpine.js: LiveWire és JavaScript kapcsolata

2.3.6. WireToast: LiveWire alapú értesítések

2.4. Websocket és Pusher szerviz

2.4.1. Websocket jelentősége

2.4.2. Pusher

2.4.3. Websocket integrálása a keretrendszerbe

2.4.4. Csatornák, események létrehozása

2.4.5. Események fogadása JavaScript LiveWire esetén

3. fejezet

Az alkalmazás tesztelése

3.1. Manuális tesztelés

3.2. Automatizált tesztelés

3.3. Terheléses tesztelés

3.4. Laravel Pint és Github actions

4. fejezet

Alkalmazás telepítése

Az alkalmazás a szakdolgozat védés, illetve a záróvizsga időszak alatt elérhető a **https://szakdolgozat.ddns.net** címen, ahol ki lehet próbálni az alkalmazást. Pár alapértelmezett felhasználó elérhető:

- Szuper adminisztrátor:
 - Felhasználónév: *SADMIN*
 - Jelszó: *superadmin*
- Adminisztrátor:
 - Felhasználónév: *ADMIN0*
 - Jelszó: *admin*
- Tanár:
 - Felhasználónév: *TEACHE*
 - Jelszó: *teacher*
- Hallgató:
 - Felhasználónév: *STUDEN*
 - Jelszó: *student*

4.1. Laravel Forge

4.2. Kézi telepítés lépései

Összegzés

Irodalomjegyzék

- [1] SDA INFORMATIKA ZRT: *Neptun alkalmazás leírása*
<https://sdainformatika.hu/termekek>, Megtekintés dátuma: 2024.03.21.
- [2] FILEINFO.COM: *ICS fájl leírása*.
<https://fileinfo.com/extension/ics>, Megtekintés dátuma: 2024.03.21.
- [3] MDN WEB DOCS: *Progressive web apps*
https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps,
Megtekintés dátuma: 2024.03.21.