

ZÁRÓDOLGOZAT

Mátrai Gergő

5/13SZOFT

BAJAI SZC TÜRRI ISTVÁN TECHNIKUM

SZOFTVERFEJLESZTŐ

ZÁRÓDOLGOZAT

AUTÓS ISKOLA MANAGER

Mátrai Gergő

2024

NYILATKOZAT A DOLGOZATRÓL

Alulírott (név) tanuló

kijelentem, hogy

..... című záródolgozatomat
(nyomtatott és elektronikus formában) a Bajai SZC Türr István Technikumá-
nak oktatói és tanulói:

- felhasználhatják (pl. hivatkozás alapjául, olvasótermi használá-
latra) későbbi munkájukhoz a szerzői jogok tiszteletben tartása
mellett.
- nem használhatják fel (titoktartási nyilatkozat csatolása mellett).

Ugyanakkor kijelentem, hogy a záródolgozat *saját munkám eredménye*.

Baja,

.....
aláírás

Tartalomjegyzék

1.	Bevezetés	1
1.1.	Github repository	1
2.	Felhasználói dokumentáció.....	2
2.1.	Általános specifikáció.....	2
2.2.	Web applikáció használata.....	2
2.3.	Rendszerigény.....	2
2.4.	Első indítás.....	3
2.5.	Regisztráció (Register)	3
2.6.	Bejelentkezés (Login).....	4
2.7.	Elfelejtett jelszó (Forgot Password?)	5
2.8.	Felhasználói felület (Dashboard).....	6
2.8.1.	Főoldal (Main Page)	6
2.8.2.	Kurzusok (Courses)	6
2.8.3.	Naptár (Calendar)	7
2.8.4.	Statisztika (Statistics).....	8
2.8.5.	Felhasználók (Users).....	9
2.8.6.	Tanulók (Students)	10
2.8.7.	Oktatók (Teachers).....	10
2.8.8.	Járművek (Vehicles).....	11
2.8.9.	Vizsgák (Exams)	12
2.8.10.	Fizetések (Payments)	13
2.8.11.	Profil (Profile).....	14
3.	Fejlesztői dokumentáció	15
3.1.	Bevezetés	15
3.2.	Fejlesztői környezet	15
3.2.1.	Visual Studio Code	15

3.2.2.	PostgreSQL	16
3.2.3.	Docker.....	16
3.2.4.	Git	18
3.2.5.	Github	18
3.3.	Felhasznált technológiák.....	18
3.3.1.	Bun.....	19
3.3.2.	React	19
3.3.3.	Next.js	20
3.3.4.	TypeScript.....	21
3.4.	Függőségek	21
3.5.	Telepítési útmutató.....	22
3.5.1.	Docker konténer létrehozása.....	23
3.5.2.	Manuális telepítés	23
3.6.	.env fájl felépítése	24
3.7.	Adatbázis szerkezet.....	24
3.7.1.	Adatbázis modellek.....	25
3.7.2.	ER modell	28
3.7.3.	Relációs séma	28
3.8.	E-mail küldés	29
3.8.1.	A projektben használt levelek	29
4.	Tesztelési dokumentáció	32
4.1.	Unit tesztek	32
4.2.	Funkcionális tesztek.....	33
5.	Összegzés.....	35
5.1.	Továbbfejlesztési lehetőségek.....	36
6.	Ábrajegyzék	37

1. Bevezetés

Az autósiskolák oktatói gyakran szembesülnek a tanulók adminisztratív és logisztikai kihívásaival, amelyek a tanfolyamok hatékonyságát és a diákok elégedettségét is befolyásolhatják. A papír alapú adminisztráció időigényes és több hibalehetőséget rejt magában. Az órarendek és a vizsgák összeállítása szintén fejtörést tud okozni az autósiskola vezetői számára. Erre nyújt megoldást az Autós Iskola Manager. A vezetők és az oktatók egy helyen, egyszerű kezelőfelületek segítségével képesek új órákat beírni a tanulóknak, kezelni a pénzügyeket, valamint az autós iskola járműveit is. A vezetők egy statisztika fülön tudják nyomon követni az iskola fontosabb adatait, mint például az aktív tanfolyamok számát vagy a vizsgák sikerességét. Az Autós Iskola Manager kezelőfelülete teljesen angol nyelvű, így lehetőséget nyújt a külföldiek számára is, hogy Magyarországon szerezzenek jogosítványt.

Keretrendszernek a Next.js-t választottam, mert már hosszú ideje foglalkoztam a Reacttal és valami új kihívást kerestem, valamint nagyon érdekelt a Next.js full-stack jellege és kíváncsi voltam, milyen lehetőségeket rejt magában.

1.1. Github repository

A projekt teljes forráskódja elérhető a következő linken:

<https://www.github.com/redbirdjs/autosiskola-manager>

2. Felhasználói dokumentáció

2.1. Általános specifikáció

A web applikáció egy olyan online felületet biztosít, ahol a használó gyorsan és könnyen tudja elvégezni az adminisztratív munkáját. (például: új óra beírása, fizetések kezelése).

2.2. Web applikáció használata

A „web applikáció” három szerepkört különböztet meg:

- Tanuló (Student)
- Oktató (Teacher)
- Admin (Admin)

Ez a három szerepkör különböző jogkörökkel rendelkezik és mindegyik szerepkör más-más funkciókat ér el a kezelőfelületen.


Új felhasználók felvételére, oktatók kinevezésére és a járművek kezelésére csak az Admin szerepkörrel rendelkező felhasználóknak van lehetőségük. Az oktató szerepkörrel rendelkező felhasználók kezelhetik a tanulóikat, új órákat vehetnek fel a számukra, a tanulók pénzügyeit kezelhetik, fizetési határidőt állíthatnak számukra, új vizsgát vehetnek fel, valamint könyvelhetik, hogy a vizsga sikeres volt-e vagy sem. Akik az oldalon regisztrálnak, automatikusan tanuló szerepkört kapnak. A tanulók új tanfolyamra jelentkezhetnek, ha még nem rendelkeznek egy már aktív tanfolyammal. Ha egy tanuló befejezett egy tanfolyamot sikeres vizsgával, akkor lehetősége van egy új tanfolyamra jelentkezni. A tanfolyamra jelentkezéskor a tanuló kiválaszthatja oktatóját, valamint, hogy melyik járművel szeretné elvégezni a tanfolyamot.

2.3. Rendszerigény

A web applikáció megnyitására valamennyi böngésző alkalmas. Ez alól kivételt képeznek a már nagyon elavult böngészők, mint például az Internet Explorer.

2.4. Első indítás

A web applikáció első indítás során feltölti azokat a táblákat, amelyek statikus, az applikáció működéséhez elengedhetetlen adatokat tartalmaznak (pl.: jogkörök, jogosítvány típusok), valamint létrehoz egy Admin jogkörrel rendelkező felhasználót egy alapértelmezett jelszóval, amelyet nem kötelező, de ajánlott módosítani.

 **Admin felhasználó adatai**


Az első indítás után az admin felhasználóba az alábbi adatokkal lehet bejelentkezni:


Email cím: admin@dsm.sbcraft.hu

Jelszó: Admin1234

ábra 1: Admin felhasználó adatai

2.5. Regisztráció (Register)

 Driving School Manager



Register

Username *

Full Name *

Email address *

Passport Number *

Password *

Repeat password *

Register

Already has an account? [Click here to login](#)

*: These fields are required.

ábra 2: Regisztráció oldal

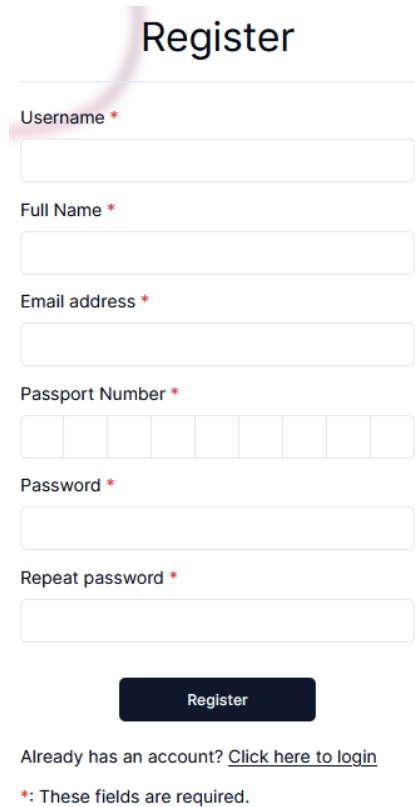
A *regisztráció* oldalon a felhasználók az adataik megadása után tanulóként regisztrálhatnak az oldalra. A sikeres regisztráció után a megadott e-mail címre egy megerősítő e-mailt is kap a felhasználó, ahol hitelesíteni tudja az e-mail címét.

A regisztráció során megadható adatok:

- Felhasználónév (Username)
- Teljes név (Full Name)
- E-mail cím (Email address)
- Útlevekszám (Passport Number)
- Jelszó (Password)
- Jelszó megerősítése (Confirm Password)

A regisztráció során a következő hibák léphetnek fel:

- Nem kitöltött kötelező mező!
- Felhasználónév túl rövid!
- Hibás e-mail cím formátum!
- Hibás útlevekszám formátum! (12345678AB)
- A jelszó nem elég erős!
- A két jelszó nem egyezik!



Register

Username *

Full Name *

Email address *

Passport Number *

Password *

Repeat password *

Register

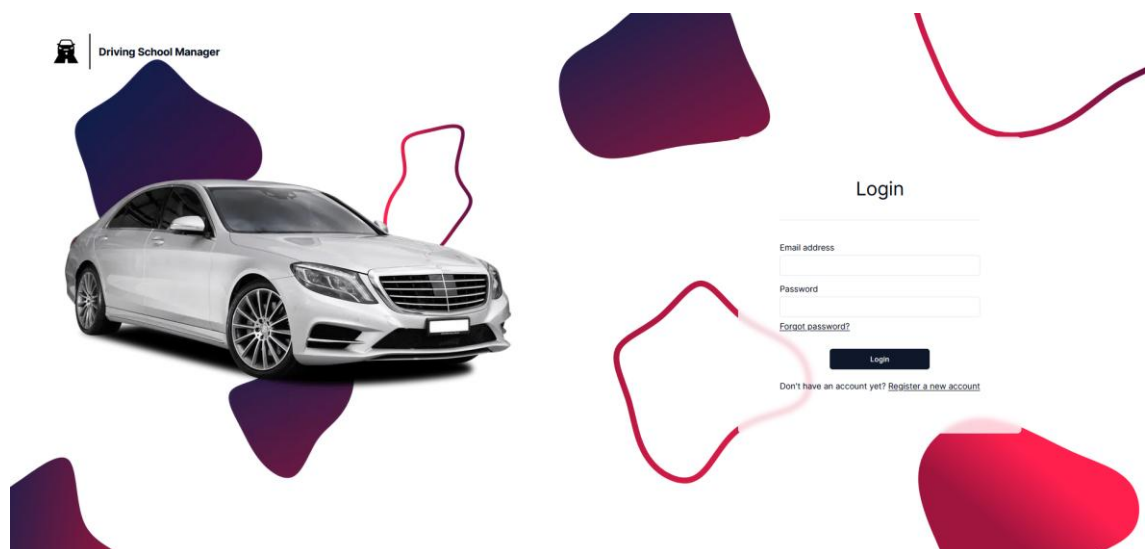
Already has an account? [Click here to login](#)

*: These fields are required.

ábra 3: Regisztráció űrlap

2.6. Bejelentkezés (Login)

A *bejelentkezés* oldalon a felhasználók az e-mail címük, valamint a hozzá tartozó jelszó megadásával léphetnek be a felhasználói felületre.



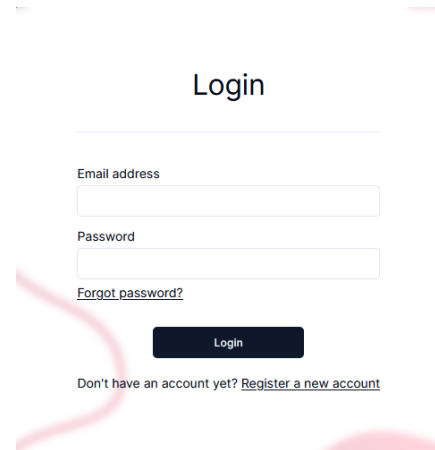
ábra 4: Bejelentkezés oldal

Bejelentkezés során megadható adatok:

- E-mail cím (Email address)
- Jelszó (Password)

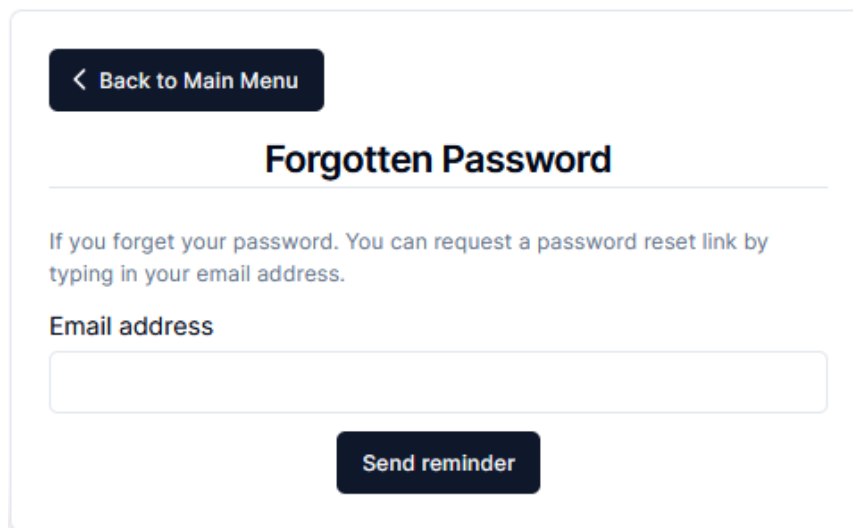
A bejelentkezés során fellépő hibák lehetnek:

- Nem kitöltött kötelező mező!
- Hibás e-mail cím formátum!
- Hibás e-mail cím és/vagy jelszó!



ábra 5: Bejelentkezés űrlap

2.7. Elfelejtett jelszó (Forgot Password?)



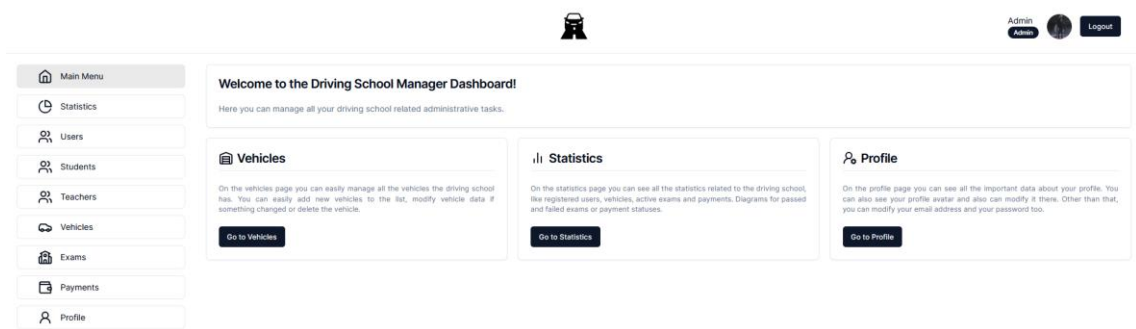
ábra 6: Elfelejtett jelszó oldal

Ha a felhasználó elfelejtette a jelszavát, lehetősége van jelszó módosítást igényelni a megfelelő e-mail cím megadását követően. A rendszer a megadott e-mail címre küld egy levelet, amelyben a gombra, vagy ha a gomb nem működik, a gomb alatt található hivatkozásra kattintva a felhasználó új jelszót tud megadni. Sikeres jelszó módosítást követően a felhasználó az új e-mail cím és jelszó párossal be tud jelentkezni a kezelő felületre.

Jelszó módosítási igénylés során fellépő hibák lehetnek:

- Hibás e-mail cím formátum!
- Az e-mail címhez nem tartozik felhasználó!
- A megadott jelszó nem elég erős!
- A két jelszó nem egyezik!

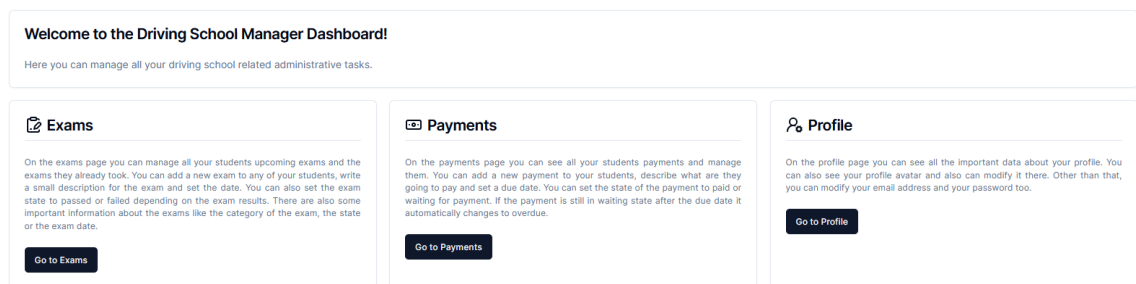
2.8. Felhasználói felület (Dashboard)



ábra 7: Felhasználói felület admin felhasználóval bejelentkezve

A felhasználói felület csak bejelentkezett felhasználók számára elérhető. Ezen a felületen szerepkörtől függően más-más navigációs gombok jelennek meg, amelyek a szerepkörhöz csatlakozó aloldalakra irányítják a felhasználót.

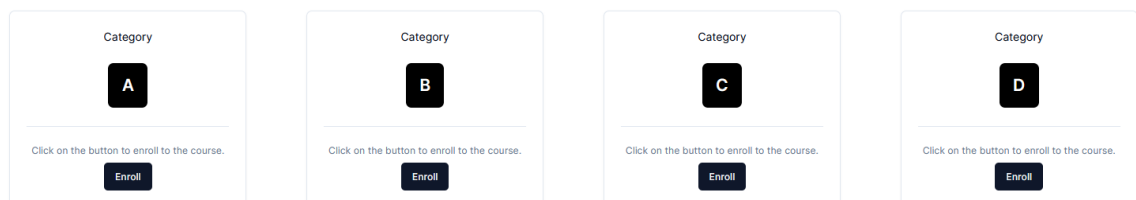
2.8.1. Főoldal (Main Page)



ábra 8: Főoldal oktató felhasználóval bejelentkezve

A bejelentkező felhasználót mindig ez az oldal fogadja. Ezen az oldalon található egy üdvözlő üzenet, valamint a szerepkörhöz tartozó fontosabb oldalakról kicsit részletes leírás és egy gomb, amely arra az aloldalra vezet.

2.8.2. Kurzusok (Courses)



ábra 9: Kurzusok oldal

A *kurzusok* oldal csak a Tanuló szerepkörrel rendelkező felhasználók számára elérhető akkor, ha nem rendelkeznek aktív kurzussal. Ha ezen feltételek közül valamelyik nem teljesül a felhasználó vissza kerül a főoldalra.

Ezen az oldalon a tanuló kiválasztja, hogy melyik kurzusra szeretne jelentkezni, valamint az oktatót és a járművet is. Az „Enroll” (beiratkozás) gombra kattintva jelentkezhet a kurzusra.

2.8.3. Naptár (Calendar)

Dashboard / Calendar

+ New Event

April 2024

month week

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9 • 2p Payment - Spense Johannesson	10	11	12	13
14	15	16	17	18	19	20
21 • 2p Payment - Nelli McBride	22 • 8:30a Exam - Nelli McBride • 8:35a Exam - Spense Johannesson	23 • 5a Driving Practise - Nelli McBride	24 • 10a Exam - Amalie Christol	25 • 10a Driving Practise - Amalie Chris	26 • 2p Payment - Amalie Christol	27
28	29	30 • 10a Exam - Nelli McBride • 2p Payment - Nelli McBride	1	2	3	4
5	6	7	8	9	10	11

ábra 10: Naptár oldal oktató felhasználóval bejelentkezve

A *naptár* oldal a Tanuló és Oktató szerepkörrel rendelkező felhasználók számára érhető el. A tanuló a hozzá tartozó feljegyzett eseményeit követheti nyomon, mint például tanórák, vizsgák, fizetési határidők. A fizetés színe változó attól függően, hogy milyen állapotban van:

- Zöld: fizetett
- Narancssárga: fizetésre vár
- Piros: nem fizetett határidőn belül

Az oktató az összes hozzá tartozó tanuló eseményeit látja a naptáron. Az oktató továbbá új eseményeket is fel tudnak venni a tanulóik számára, amelyek nem tartoznak a vizsgák vagy a fizetések közé (pl.: gyakorlati óra). Ezeknek az eseményeknek a színét kedve szerint választhatja meg az oktató.

Új esemény felvétele során megadható adatok:

- Tanuló
- Esemény dátuma
- Esemény címe
- Esemény rövid leírása
- Esemény színe

Felvétel során ezek a hibák léphetnek fel:

- Nincs tanuló kiválasztva!
- Nincs dátum kiválasztva!
- Nem megfelelő dátum formátum!
- Nincs cím megadva!

New Calendar Event

Student *

Select a student...

Date *

dd / mm / yyyy , -- : --

Title *

Description

Color

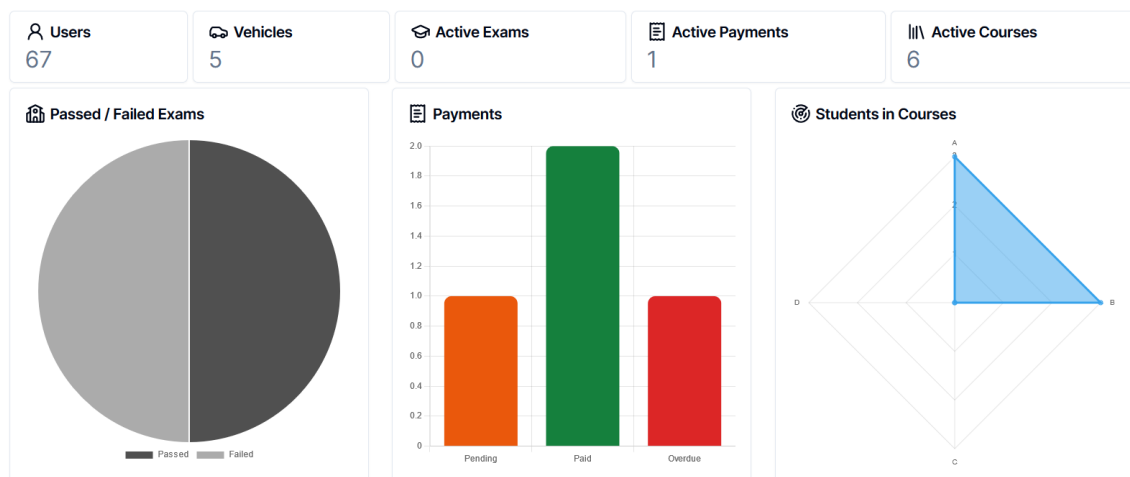
#000000

+ Add Event

*: These fields are required.

2.8.4. Statisztika (Statistics)

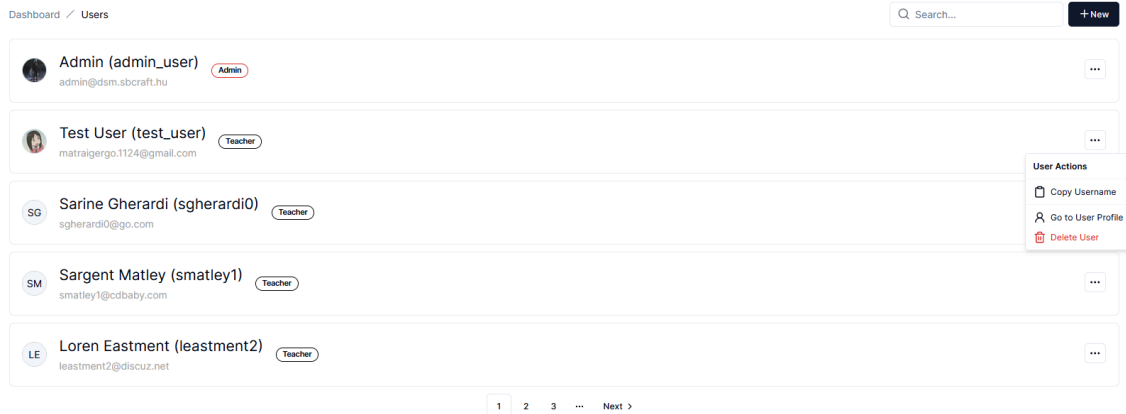
ábra 11: Új esemény felvétele



ábra 12: Statisztika oldal

A *statisztika* oldal az Admin szerepkörrel rendelkező felhasználók számára elérhető oldal. Ezen az oldalon láthatják az autós iskola legfontosabb adatait, mint például, a felhasználók, járművek számát, az aktív kurzusok és aktív vizsgák számát. Ezek mellett három grafikon mutatja a sikeres és sikertelen vizsgák számát, a fizetések állapotait (fizetésre vár, kifizetett, nem fizetett határidőn belül), valamint a tanulók elosztását a különböző kategóriákban.

2.8.5. Felhasználók (Users)



ábra 13: Felhasználók oldal

A *felhasználók* oldal az Admin szerepkörrel rendelkező felhasználók számára elérhető. Ezen az oldalon az admin felhasználók új felhasználót hozhatnak létre. Létrehozáskor meg tudják adni, hogy a felhasználó milyen szerepkörrel rendelkezzen. A létrehozás mellett a felhasználókat törölni is tudják.

Új felhasználó létrehozása során megadható adatok:

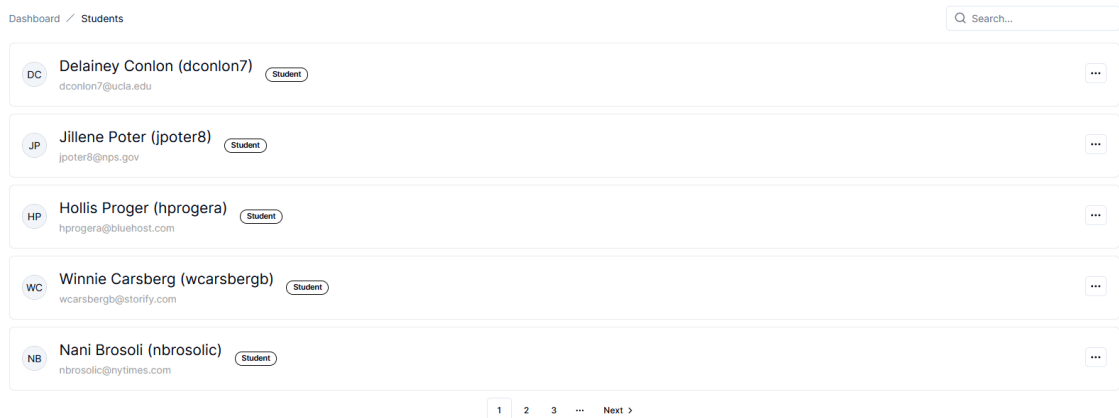
- Felhasználónév
- Teljes név
- E-mail cím
- Útlevekszám
- Szerepkör

Létrehozás során ezek a hibák léphetnek fel:

- Felhasználónév megadása kötelező!
- Teljes név megadása kötelező!
- E-mail cím megadása kötelező!
- Útlevekszám megadása kötelező!
- Hibás útlevekszám formátum!
- Nincs szerepkör kiválasztva!

ábra 14: Új felhasználó felvétele

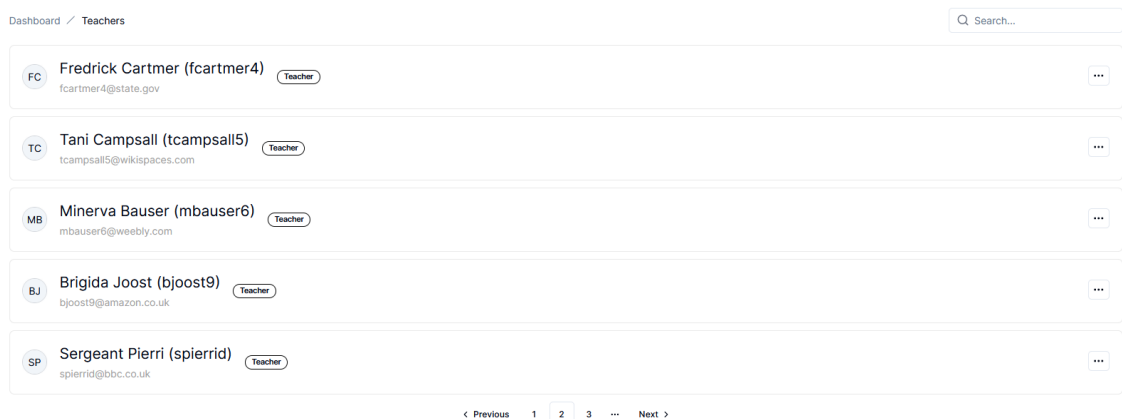
2.8.6. Tanulók (Students)



ábra 15: Tanulók oldal

A *tanulók* oldal egy az Oktató szerepkörrel rendelkező felhasználók számára elérhető oldal. A tanulók oldalon az oktatók megtekinthetik a hozzájuk tartozó összes tanuló fontosabb adatait. Ezen az oldalon a felhasználó tud név szerint keresni egy, vagy több tanulóra. A továbbiak (●●●) gombra kattintva az oktató gyorsan átugorhat a tanuló profil oldalára, ahol meg tudja tekinteni a tanuló összes fontosabb adatát.
















2.8.7. Oktatók (Teachers)



ábra 16: Oktatók oldal

Az *oktatók* oldal az Admin szerepkörrel rendelkező felhasználók számára elérhető oldal, ahol a felhasználó az Oktató szerepkörrel rendelkező felhasználókat kezelheti. Ez az oldal többnyire megegyezik a *tanulók* oldallal. Ugyanúgy rendelkezik egy keresőmezővel, ahol a felhasználó képes az oktatók között keresni, valamint a továbbiak (●●●) gombra kattintva képes átugrani az oktató profil adatlapjára, ahol további információkat tud megtekinteni a felhasználóról.

2.8.8. Járművek (Vehicles)

Dashboard / Vehicles		Q Search...	+ New
	KTM RC 390 A ABC-123		
	Kawasaki Ninja A AAA-111		
	Suzuki Vitara B BGC-123		
	Toyota Auris Hybrid B HVN-444		
	Honda Civic Lx B HZA-786		

ábra 17: Járművek oldal


A *járművek* oldal szintén az Admin szerepkörrel rendelkező felhasználók számára elérhető. Ezen az oldalon a felhasználó kezelheti az összes járművet. Az oldalon található egy kereső mező, amely segítségével a jármű több tulajdonsága alapján tudunk keresni, mint például a neve vagy a rendszáma. A felhasználónak lehetősége van új jármű felvételére, már létező jármű adatainak módosítására, valamint jármű törlésére.

Új jármű felvétele során megadható adatok:

- Jármű gyártója (Brand)
- Jármű típusa (Type)
- Jármű rendszáma (Plate)
- Jármű kategóriája (Category)
- Jármű színe (Color)
- Jármű hajtástípusa (Drive Type)
- Jármű látványkép (Preview Image)

Felvétel során ezek a hibák léphetnek fel:

- Nem kitöltött kötelező mező!
- Hibás rendszám formátum!
- Nincs kiválasztott kategória!

+  New Vehicle

Brand *

Type *

Plate *

Category *

A

B

C

D

Color *

Drive Type *

Select the drive type... ▾

Preview Image

Browse... No file selected.

+ Add new vehicle

*: These fields are required.

ábra 18: Új jármű felvétele

2.8.9. Vizsgák (Exams)

Dashboard / Exams + New

# No.	Date of the Exam	Category	Student	Description	State	
#0002	2024-04-22 08:30:00	A	Nelli McBride	Theory Exam	✓ Passed	...
#0003	2024-04-22 08:35:00	B	Spense Johannesson	Theory Exam	✓ Passed	...
#0004	2024-04-24 10:00:00	B	Amalie Christol	Theory Exam	✗ Failed	...
#0005	2024-04-30 10:00:00	A	Nelli McBride	Practise Exam	✗ Failed	...

Previous Next

ábra 19: Vizsgák oldal


A vizsgák oldal az Oktató és Admin szerepkörrel rendelkező felhasználók számára elérhető. Az oktatók csak a saját tanulóikat, míg az admin felhasználók az összes tanulót kezelhetik. A felhasználó új vizsgát vehet fel, vizsgát törölhet, valamint a vizsga sikerességét állíthatja be (átment, megbukott).

Új vizsga felvétele során megadható adatok:


- Tanuló (Student)
- Vizsga rövid leírása (Description)
- Vizsga időpontja (Date)

Felvétel során fellépő hibák lehetnek:

- Nem kitöltött kötelező mező!
- Hibás dátum formátum!


+  New Exam

Course *

Select a student... 

Description *

Date *

dd / mm / yyyy , -- : -- 

+ Add Exam

*: These fields are required.

ábra 20: Új vizsga felvétele

2.8.10. Fizetések (Payments)

Dashboard / Payments + New

# No.	Description	Student	Issuer	Amount	State	Created	Due	
#0002	Cat. A Theory Book	Nelli McBride	Test User	\$25.00	✓ Paid	2024-04-19 10:07:47	2024-04-21 14:00:00	...
#0003	Cat. B Theory Book	Spense Johannesson	Test User	\$30.00	🕒 Overdue	2024-04-04 10:08:15	2024-04-10 14:00:00	...
#0004	Cat. B Theory Book	Amalie Christol	Test User	\$30.00	✓ Paid	2024-04-19 10:08:43	2024-04-26 14:00:00	...
#0005	Practise Exam	Nelli McBride	Test User	\$45.00	🕒 Pending	2024-04-19 10:09:09	2024-04-30 14:00:00	...

Previous Next

ábra 21: Fizetések oldal


A *fizetések* fül mindhárom szerepkör számára elérhető oldal, viszont szerepkörtől függően más funkcionalitása van az oldalnak. A Tanuló szerepkörrel rendelkező felhasználók csak megtekinteni tudják a különböző fizetési kötelezettségeiket, azok határidejét. Az Oktató és Admin szerepkörrel rendelkező felhasználók új fizetést hozhatnak létre, törölhetnek, valamint a fizetés státuszát módosíthatják (fizetett, fizetésre vár). Ha egy fizetés a határidő lejártá után is „fizetésre vár” státuszban van, akkor automatikusan „nem kifizetett” státuszba vált.

Új fizetés felvétele során megadható adatok:


- Tanuló (Student)
- Fizetés rövid leírása (Description)
- Fizetés összege (Amount)
- Fizetés határideje (Due)

Felvétel során ezek a hibák léphetnek fel:

- Nincs kiválasztott tanuló!
- A leírás kötelező!
- Összeg megadása kötelező!
- Határidő megadása kötelező!

+  New Payment

Student *


Select a student... 

Description *

Amount *

0

Due *

dd / mm / yyyy , -- : -- 

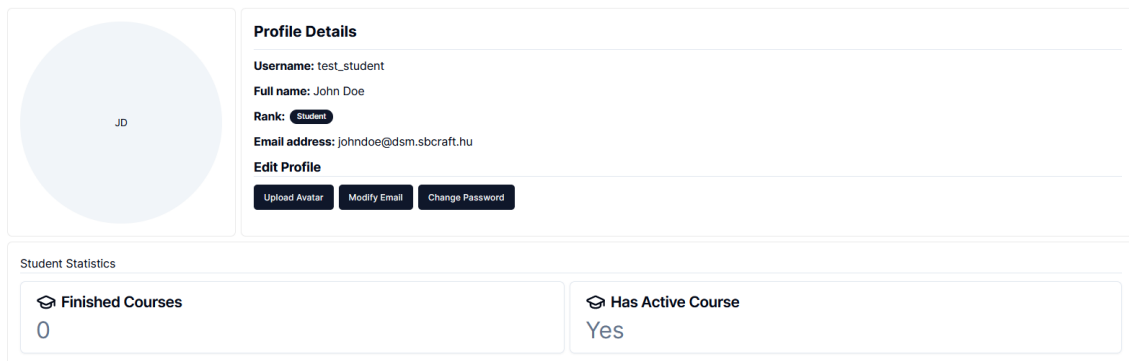
+ New Payment

*: These fields are required.



ábra 22: Új fizetés felvétele

13

2.8.11. Profil (Profile)



Profile Details	
Username: test_student	
Full name: John Doe	
Rank: Student	
Email address: johndoe@dsmsbcraft.hu	
Edit Profile	
Upload Avatar	Modify Email Change Password

Student Statistics	
<div> Finished Courses</div> <div>0</div>	<div> Has Active Course</div> <div>Yes</div>

ábra 23: Saját profil oldal

A profil oldal az összes felhasználó számára elérhető oldal. A profil oldal két különböző útvonalra osztozik:

- /profile
- /profile/[felhasználónév]

A sima profil oldalon a bejelentkezett felhasználó a saját fontosabb adatait (pl.: teljes név, szerepkör), profilképét látja, valamint a profil adatait módosíthatja, mint például az e-mail címét, jelszavát, vagy a profilképét.

A Tanuló szerepkörrel rendelkező felhasználók látják, hogy van-e aktív kurzusuk, valamint hány darab kurzust végeztek már el.

Az Oktató szerepkörrel rendelkező felhasználók látják hány tanulójuk van, valamint, hány aktív kurzusuk van.

Az Admin szerepkörrel rendelkező felhasználókhöz nem tartozik statisztika. Az ő profil oldalukon csak a fontosabb felhasználó adatok jelennek meg (pl.: teljes név, szerepkör), valamint a profilképük.

A /profile/[felhasználónév] (felhasználónév - paraméter) oldalon egy másik felhasználó adatait tudja megtekinteni. Többek között erre az oldalra vezetnek a *tanulók*, *oktatók* és *felhasználók* oldalon található „tovább a profilhoz” gombok is. Nem létező felhasználónév megadása esetén az oldal egy „Felhasználó nem található!” hibaüzenetet jelenít meg.

3. Fejlesztői dokumentáció

3.1. Bevezetés

Az Autós Iskola Manager egy Next.js¹-ben íródott web applikáció. Az autós iskolák adminisztrációja sokszor papír alapon történik, amely lassú és sok hibázási lehetőséget rejt. Ezekre a problémákra kínál megoldást az Autós Iskola Manager. Egy egyszerű felhasználói felületen tudják mind a tanulók, mind az oktatók a saját adminisztratív tevékenységüket elvégezni.

3.2. Fejlesztői környezet

- Visual Studio Code² v1.88
- PostgreSQL³ v16.2 – adatbázis
- Docker⁴ v4.29.0 – virtualizált futtatási környezet
- Git⁵ v2.34.1 – verzió kezelés
- Github⁶ – verzió kezelés, projekt management

3.2.1. Visual Studio Code

A Visual Studio Code a Microsoft által fejlesztett nyílt forráskódú, ingyenes szövegszerkesztő, amely nagyon népszerű a szoftverfejlesztők között. Keresztplatformos alkalmazás, amely azt jelenti, hogy Windowsra, Linuxra és macOS-re is elérhető.

A sok hasznos funkció és a közösség által fejlesztett kiegészítők teszik a legjobb szövegszerkesztővé a szoftverfejlesztési iparban. Néhány ilyen hasznos funkció az intelligens kódkiegészítés és hibajavítás, valamint szinte minden programozási nyelvet támogat.

¹ <https://nextjs.org/>

² <https://code.visualstudio.com/>

³ <https://www.postgresql.org/>

⁴ <https://www.docker.com/>

⁵ <https://git-scm.com/>

⁶ <https://www.github.com/>

3.2.2. PostgreSQL

A PostgreSQL egy ingyenes, nyílt forráskódú objektum-relációs adatbázis kezelő rendszer, amelyet a PostgreSQL Global Development Group fejleszt. A PostgreSQL-t gyakran csak „Postgres”-nek nevezik.

Néhány fő jellemzője:

- **Objektum-relációs adatbázis**
- **Tranzakcionális támogatás**
- **Kiterjeszthetőség**
- **Széles körű funkciók**

Az egyszerű relációs adatbázisokkal ellentétben a Postgres képes objektumokat tárolni, valamint támogatja az objektumorientált adatbázis-funkciókat, mint például a polimorfizmust vagy az öröklődést.

A Postgres támogatja a tranzakciókat, amely biztonságos adatmanipulációt, konzisztenciát és integritást jelent. A tranzakciókat az ACID (Atomicity, Consistency, Isolation, Durability) elvek szerint kezeli.

Kiterjeszthetősége abban rejlik, hogy engedi a fejlesztők számára, hogy egyedi adatbázis-funkciókat, típusokat hozzanak létre igényeiknek megfelelően, valamint támogatja a külső modulok használatát.

A Postgres nagyon sok funkciót tartalmaz, amelyek adatok tárolására, lekérdezésére, kezelésére szolgálnak. Olyan funkciók, mint például az indexelés, amely lehetővé teszi a gyors adat lekérdezést vagy a nézetek, amelyek saját adattáblák létrehozására szolgálnak a könnyebb lekérdezések érdekében.

3.2.3. Docker

A **Docker** egy nyílt forráskódú platform és technológia, amely lehetővé teszi a szoftver-alkalmazások konténerizált futtatását. A konténerizáció azt jelenti, hogy az alkalmazás összes függősége és a futtatásához szükséges környezet egyetlen önálló csomagban van. Célja, hogy megkönnyítse az alkalmazások telepítését és futtatását, valamint skálázhatóságot biztosítson.

A Docker egyaránt használt a fejlesztés során, valamint a kész alkalmazás futtatása során is.

Több előnye is van a Dockeren való futtatásnak a fizikai szerveren való futtatással szemben:

- **Konténerek**
- **Biztonság**
- **Konténerek kezelése, skálázhatóság**
- **Olcsó, erőforrás kihasználás**

A Docker úgynevezett konténereket (container) használ az applikációk futtatására. Ezek lehetnek önálló vagy akár összetett konténereket is. Az önálló konténerek csak egymagukban futnak, míg az összetett konténerek esetén több különböző konténer kerül összekapcsolásra és együtt alkotnak egy működő egységet. Minden konténer az applikációnak megfelelő futtatási környezettel rendelkezik és csak az fut rajta.

A Dockerben való futtatás biztonságosabb, mint a fizikai szerveren való futtatás, mert minden konténer úgy viselkedik, mint egy önálló virtuális gép. Teljesen el van különítve a gazda géptől, ezzel védve azt. Emellett bármikor egyszerűen biztonsági mentést tudunk készíteni a konténer összes adatáról, amit később vissza tudunk tölteni, ha bármi hiba folytán szükség lenne rá. A konténerek egy belső hálózaton közvetlenül kommunikálnak egymással, ezért nincs szükség arra, hogy például egy adatbázist tartalmazó konténer egy Internet felé nyitott porttal rendelkezzen.

A Docker lehetőséget biztosít a konténerek kezelésére is. Könnyedén létrehozhatunk, újraindíthatunk, leállíthatunk, vagy akár törölhetünk konténereket. Lehetőségünk van a konténerek skálázására is, valamint automatikus integrálására és folyamatos telepítésére is. Ez azért jó, mert több konténerben tudjuk ugyanazt az applikációt futtatni és így könnyedén el tudjuk osztani a terhet.

Minden Docker kép fájl (Docker Image) egy fájlokat és utasításokat tartalmazó csomag, amelyből könnyedén tudunk konténereket létrehozni. Ezek a képfájlok, tartalmazzák a futtatási környezetet, a futtatáshoz szükséges fájlokat és függőségeket, valamint az indításhoz szükséges parancsokat. Maximálisan kihasználják az erőforrásokat, így nincs szükség minden egyes applikáció esetén egy új fizikai szerver üzemben tartására, ezért sokkal olcsóbb is a használatuk.

3.2.4. Git

A Git egy nyílt forráskódú elosztott (distributed) verzió kezelő rendszer, amelyet a szoftverfejlesztők a projektjeik verzióinak nyomon követésére használnak. A Git nagyon gyorsan és hatékonyan tudja nyomon követni és kezelni a projektek kódbázisát. A Git továbbá támogatja a kollaboratív munkát csapatok között.

Minden Git által kezelt projektet „repository”-nak neveznek. Ez a repository tartalmazza a projekt összes fájlját, valamint a fejlesztés során történt összes változást.

3.2.5. Github

A Github egy online platform, amely Git repository-k megosztására és tárolására alkalmas. A Github első sorban kód megosztásra és közösségi fejlesztésre lett kialakítva, számos funkciót kínál a fejlesztők és csapatok számára, hogy hatékonyan tudják kezelni a projektjeiket.

Néhány ilyen funkció:

- Issues (Problémák)
- Pull requests (Húzási kérések)

Az Issues fülön a felhasználóknak lehetőségük van a repository-ban talált hibák jelentésére. Egy fórum szerű felületet kínál a Github felhasználói számára, ahol könnyedén tudnak egymással kommunikálni a hibát illetően, valamint minden ilyen problémát különböző címkékkel lehet ellátni attól függően, hogy egy kis vagy nagy hibáról van szó.

A Pull requestek lehetővé teszik, hogy a közösség kód módosítási javaslatot tegyen a repository-ban. A repository tulajdonosa meg tudja nézni a pull request indoklását, valamint, hogy mit módosított a felhasználó. A tulajdonos el tudja fogadni vagy el tudja utasítani a pull requestet. Elfogadás esetén a pull request hozzáadódik a kódbázishoz.

3.3. Felhasznált technológiák

- **Bun**⁷: JavaScript futtatási környezet
- **React**: Egy Facebook által fejlesztett könyvtár webes és natív interfészek létrehozására.
- **Next.js**: React alapú full-stack webes keretrendszer.

⁷ <https://bun.sh/>

- **TypeScript⁸**: Erősen típusos JavaScriptre épített programozási nyelv.
- **Prisma**: Modern, nyelvfüggetlen ORM.
- **Shadcn/ui⁹**: Egy Radix-UI-ra épülő komponens könyvtár a gyors web alkalmazások építésére.

3.3.1. Bun

A **bun** egy olyan platform, ahol a felhasználók egyszerűen és gyorsan tudnak JavaScript vagy TypeScript kódot futtatni a böngészőjükben vagy parancssorban.

Három fő célja van:

- **Gyorsaság**
- **Egyszerű API**
- **Jó fejlesztői élmény**

A bun egy sokkal gyorsabb futtatási környezetet biztosít, mint a Node.js. A Safari (Apple által fejlesztett böngésző) által használt gyorsaság orientált JavaScriptCore nevű JavaScript motort bővíti ki. Azonban a bun teljesen kompatibilis a Node.js-el, mert ugyanúgy egy package.json fájlba menti a függőségeket, valamint node_modules mappát használ. Az egyetlen különbség a lock fájlban mutatkozik meg, ami a bun esetében bun.lockb, míg a Node.js esetén package-lock.json. A bun gyorsasága ebben a fájlban rejlik, amely tartalmaz fontos adatokat a függőségekről, ami gyorsabbá teszi a letöltésüket.

Minimális, magasan optimalizált API-al rendelkezik, amely segítségével általános feladatokat tudunk végrehajtani, mint például egy webservert futtatása vagy fájlok írása.

Egy teljes eszköztárral rendelkezik JavaScript applikációk fejlesztéshez. Tartalmaz egy csomag kezelőt, egy bundlert, valamint lehet vele teszteket is futtatni.

3.3.2. React

A **React** egy nyílt forráskódú, Facebook által fejlesztett JavaScript könyvtár, amely segítségével egyszerűen tudunk létrehozni webes vagy natív (telefonos applikáció) interfészeket. Könnyen használható, nagyon hatékony és skálázható.

⁸ <https://www.typescriptlang.org/>

⁹ <https://ui.shadcn.com/>

Építőelemei a **komponensek**, amelyek újra használható egységek. Ezek a komponensek lehetnek nagyon egyszerűek, mint például egy gomb, vagy komplexek, mint egy teljes űrlap (form). A komponenseknek tudunk adni úgynevezett propokat, amelyek segítségével más funkcionalitást lehet hozzájuk rendelni, vagy más adatokat megjeleníteni. Például egy gomb esetén át tudunk adni a komponensnek egy funkciót, amelyet a gomb megnyomására lefuttatunk, vagy egy adatlap esetén át tudjuk adni a komponensnek egy felhasználó adatait, így annak a felhasználónak az adatai fognak majd megjelenni.

Egy úgynevezett **virtuális DOM**-ot (Document Object Model) használ, amely egy absztrakciója a valós DOM-nak (A DOM a weboldal objektum alapú reprezentációja, amely lehetővé teszi, hogy JavaScript segítségével módosítani tudjuk a HTML weboldalunkat futási időben). Ezt a DOM-ot használja a React a dinamikus frissítésekre. Ha egy adat módosul az oldalon, akkor a React először a virtuális DOM-ot újra generálja, majd összehasonlítja a régi és új virtuális DOM-ot, hogy megtalálja a változásokat. Ezután csak a változásokat módosítja az igazi DOM-on, ezzel biztosítva a gyorsaságot és a nagyobb teljesítményt.

A react **egyirányú adatkötést** használ, ami azt jelenti, hogy a komponensek nem feltétlenül kommunikálnak egymással. Ehelyett az adatok és események lefelé áramlanak a komponens hierarchiában, ha azt szeretnénk, hogy egy komponens megkapja egy másik komponens adatait, azt propokkal tudjuk átadni a másik komponensnek.

A react komponensek legtöbb esetben **JSX** (JavaScript XML) kiterjesztéssel rendelkeznek. A JSX egy kiterjesztése a JavaScriptnek, amely lehetővé teszi, hogy HTML kódot írassunk a JavaScript kódban. Ez nagyban megkönnyíti a fejlesztés folyamatát.

3.3.3. Next.js

A **Next.js** egy Reactra épülő **full-stack** webes keretrendszer. A full-stack azt jelenti, hogy a frontend és a backend egy applikáción belül található meg. A Next.js képes közvetlenül meghívni a backend funkciókat a frontend komponensekben. A Next.js alapvetően SSR (szerver oldali renderelés). A szerver oldali renderelés azt jelenti, hogy a szerver állítja össze a weboldalt és a kész weboldalt küldi el a felhasználó számára, ezért a komponensek két csoportra osztoznak:

- kliens oldali komponensek
- szerver oldali komponensek

A Next.js alapvetően szerver oldali komponensként kezeli az összes létrehozott komponenst és oldalt. Ha felhasználói interakcióra van szükségünk, mint például egy form kitöltésénél vagy egy gomb megnyomásánál, akkor azt a komponenst át kell helyeznünk kliens oldalra. Kliens oldalra úgy tudunk helyezni egy komponenst, hogy a „use client” direktívát használjuk a komponens fájl legelső sorában. Ezeket a kliens oldali komponenseket a Next.js úgy rendereli, hogy szerver oldalon létrehozza az oldalt úgy, hogy azzal a felhasználó még nem képes interakcióba lépni, hanem miután elküldte az oldalt a felhasználó számára, utólag úgymond „hidratálja”, azaz JavaScript segítségével interaktívvá teszi az oldalt a felhasználó számára. A hidratáció az a folyamat, amely során a statikus, nem interaktív oldalunkhoz eseménykezelők (event listener) segítségével funkciókat kapcsolunk, így interaktívvá téve az oldalt.

3.3.4. TypeScript

A TypeScript egy nyílt forráskódú programozási nyelv, amely a JavaScript kiterjesztése. A Microsoft terméke, és JavaScript alapú applikációk fejlesztésére készült. Célja, hogy nagyobb biztonságot, olvashatóságot és skálázhatóságot biztosítson.

Egy erősen típusos nyelv, amely a JavaScript hibáit próbálja meg kiküszöbölni, mint például egy változó típusának megváltoztatása futás során, így elkerülve a különböző típusok okozta hibákat. A TypeScript emellett támogatja az új ECMAScript (JavaScript hivatalos neve) verziókat, amely lehetővé teszi az újabbnál újabb funkciók használatát, mint például az arrow (nyilas) funkciók.

A TypeScript kiterjeszthető, ami azt jelenti, hogy a fejlesztők saját típusokat és interfészeket írhatnak, amelyek kielégítik az általuk fejlesztett applikáció igényeit.

Teljesen kompatibilis a JavaScripttel, mert minden TypeScript fájl JavaScriptté fordítható.

3.4. Függőségek

Az összes projekt függőség a **package.json** fájlban található meg a *dependencies* és *devDependencies* objektumokban.

- typescript v5: JavaScript szintaktikával és típusokkal.
- react v18.2.0: Egy könyvtár webes és natív interfészek létrehozására.
- react-dom v18.2.0: React csomag, amely DOM specifikus metódusokat tartalmaz.
- next v14.2.2: Egy react alapú full-stack framework.

- prisma v.5.10.2: Adatbázis ORM.
- tailwindcss v3.3.0: CSS framework.
- postcss v8: Egy toolkit, amely segítségével módosíthatjuk a CSS-t JavaScript segítségével.
- autoprefixer v10.0.1: Egy PostCSS plugin, amely automatikusan hozzáadja a böngésző specifikus CSS prefixeket.
- radix-ui v1.0.5: Előre designolt komponensek, erre épül a shadcn/ui.
- shadcn/ui v1.0.0: Radix UI-ra épült react komponens könyvtár.
- eslint v8: JavaScript szintaktika ellenőrző.
- zod v3.22.4: TypeScript alapú séma validáció könyvtár formok könnyű kezeléséhez.
- bcrypt v5.1.1: Egy Blowfish rejtjelre épülő jelszó titkosító funkció.
- jsonwebtoken (JWT) v9.0.2: Egy nyílt forráskódú standard (RFC 7519), amely kompakt és biztonságos adatátvitelt biztosít JSON objektumok formájában.
- resend v3.2.0: E-mail SMTP szerver.
- react-email v2.1.0: React alapú e-mail komponensek, e-mail designolásra.
- use-debounce v10.0.0: Felhasználó bemenet késleltetésre használt könyvtár.
- clsx v2.1.0: Feltételes CSS classok építése.
- lucide-react v0.356.0: Egy közösség által létrehozott, nyílt forráskódú ikon csomag.
- react-table v8.15.3: React tábla komponens.
- chart.js v4.4.2: Nyílt forráskódú diagramm könyvtár.
- react-chartjs-2 v5.2.0: React chartjs komponensek.
- moment v2.30.1: Dátum kezelő könyvtár
- sharp v0.33.3: Kép optimalizáló könyvtár buildeléshez

3.5. Telepítési útmutató

A web applikációt kétféleképpen tudjuk telepíteni. Az első és legegyszerűbb módszer a Docker image (képfájl) létrehozása az applikációból és annak futtatása konténerben. A második módszer az egyéni feltelepítés, amelyet személy szerint nem ajánlok, mert nagyon hosszadalmas folyamat, valamint a konténerizált applikáció egyben biztonságosabb is, mint a gazda gépen való futtatás, mert a konténer teljesen izolálva van a számítógéptől.

3.5.1. Docker konténer létrehozása

A konténer létrehozásának feltétele, hogy a Docker telepítve legyen a számítógépünkön.

Telepítés lépései:

- 1.) Készítsünk egy másolatot a github repository-ról. (szükséges program: Git)

```
git clone https://github.com/redbirdjs/autosiskola-manager
```

- 2.) Csomagoljuk ki a fájlokat.

- 3.) A mappában futtassuk le a következő parancsot:

```
docker compose --build -d
```

A parancs futásának befejezése után létrejön 3 konténer, egy a web applikációnak, egy az adatbázisnak és egy a kép kiszolgálónak.

3.5.2. Manuális telepítés

A manuális telepítéshez szükséges programok:

- Git verzió kezelő
- Bun vagy Node.js¹⁰ JavaScript futtatási környezet
- PostgreSQL adatbázis szerver

Telepítés lépései Bunt használva:

- 1.) Készítsünk egy másolatot a github repository-ról.

```
git clone https://github.com/redbirdjs/autosiskola-manager
```

- 2.) Töltsük ki a .env fájlt a megfelelő adatokkal, mint például a saját adatbázis szerverünk adatai vagy a Resend¹¹ email kiszolgáló API kulcsa.

- 3.) Lépünk be a mappába és futtassuk a következő parancsot:

```
bun install
```

- 4.) Indítsuk el az adatbázis szerverünket.

- 5.) Migráljuk az adatbázis modellt, az adatbázis szerverünkre.

```
bun run prisma:migrate
```

- 6.) Töltsük fel a szükséges adatokat az adatbázisba.

```
bunx prisma db seed
```

¹⁰ <https://nodejs.org/en>

¹¹ <https://resend.com>

7.) Készítsünk egy production buildet a web applikációról.

```
bun run build
```

8.) Futtassuk a web applikációt.

```
bun start
```

3.6. .env fájl felépítése

Az env fájl (környezeti változókat tartalmazó fájl) olyan információkat tartalmaz, amelyeket sűrűn használnak a fejlesztés során (pl.: kiszolgáló elérési útvonala) vagy titkos adatok, amelyeket nem szeretnénk megosztani a felhasználókkal (pl.: adatbázis kapcsolat, e-mail kiszolgáló token).

```
DATABASE_URL="postgresql://<username>:<passwd>@<db-host>:<db-  
port>/<db-name>?schema=<db-schema>"  
RESEND_KEY="<resend-api-key>"  
REF_SECRET="<refresh-token-secret>"  
REF_EXPIRE="1d"  
SITE_URL="http://localhost:3000"  
IMAGE_PROVIDER="http://localhost:8080"
```

DATABASE_URL	Prisma adatbázis elérési útvonal.
RESEND_KEY	Resend Email SMTP API kulcs.
REF_SECRET	Session kezeléshez használt refreshToken rejtjele.
REF_EXPIRE	refreshToken lejárat ideje.
SITE_URL	A web applikáció elérési útvonala.
IMAGE_PROVIDER	Külső kép kiszolgáló útvonala.

3.7. Adatbázis szerkezet

Az adatbázist a Prisma¹² ORM (Object-Relational Mapper) kezeli. A Prisma ORM egy nyelvfüggetlen TypeScript-ben íródott ORM, amely képes adat modelleket (adatbázis táblák) létrehozni, valamint kezelni azokat. A Prisma ORM azért nyelvfüggetlen, mert képes valamennyi adatbázis nyelvvel működni, mint például a PostgreSQL, a MySQL vagy, hogy egy NoSQL adatbázist említsek képes kezelni a MongoDB-t is. Egy **prisma.schema** fájl alapján hozza létre az adatbázis táblákat és ennek a fájlnek a segítségével jön létre a query builder (lekérdezés építő) könyvtár is.

¹² <https://www.prisma.io/orm>

A query builder könyvtár nagyban meg tudja könnyíteni a fejlesztés folyamatát, mert ahelyett, hogy mi írnánk meg minden egyes SQL lekérdezést, ezt a Prisma egy objektum séma alapján automatikusan elvégzi nekünk. Csak annyi dolgunk van, hogy létrehozunk egy Prisma klienst a projektünkben, és ezen a kliensen keresztül közvetlenül elérjük az összes adatbázis modellünket és tudunk velük lekérdezéseket végezni. Az összes Prisma kliens folyamat aszinkron, ami azt jelenti, hogy a háttérben megkezdődik az adatbázis lekérdezés és a kódunk tovább fut. Amint véget ért a lekérdezés egy úgynevezett „callback” segítségével visszaküldi a lekérdezés eredményét, amelyet meg tudunk jeleníteni a felhasználók számára vagy további műveleteket tudunk elvégezni vele.

3.7.1. Adatbázis modellek

Az összes adatbázis modellt és azok kapcsolatait a **prisma.schema** fájl tartalmazza.

Az **elsődleges kulcsokat** vastag betűstílussal, az idegen kulcsokat pedig szaggatott vonallal jelöltem.

users – Felhasználók adatai

id	Egész	A felhasználó ID-je
username	Szöveg	Felhasználónév
email	Szöveg	E-mail cím
realName	Szöveg	Felhasználó teljes neve
passportNumber	Szöveg	Útlevekszám
password	Szöveg	Titkosított jelszó
verifyToken	Szöveg (lehet NULL)	E-mail hitelesítő token
passwordToken	Szöveg (lehet NULL)	Jelszó módosító token
avatarPath	Szöveg	Profilkép útvonal
<u>rankId</u>	Egész	A felhasználó rang ID-je

ranks – Jogköröket tartalmazó tábla

id	Egész	A rang ID-je
name	Szöveg	A rang neve

calendar – Egyéni naptár események

Ez a tábla tartalmazza az egyénileg létrehozott eseményeket, amelyek nem vizsgák vagy fizetések.

id	Egész	Az esemény ID-je
title	Szöveg	Az esemény címe
description	Szöveg (lehet NULL)	Az esemény leírása
date	DateTime	Az esemény időpontja
color	Szöveg	Az esemény színe

courses – Kurszus adatok

id	Egész	Kurszus ID-je
theory	Egész	Elméleti százalék
practise	Egész	Gyakorlati százalék
<u>studentId</u>	Egész	Tanuló ID-je
<u>teacherId</u>	Egész	Oktató ID-je
<u>vehicleId</u>	Egész	Jármű ID-je
<u>categoryId</u>	Egész	Kategória ID-je

payments – Fizetések adatai

id	Egész	Fizetés ID-je
description	Szöveg	Fizetés leírása
amount	Valós	Fizetés összege
created	DateTime	Létrehozás dátuma
due	DateTime	Határidő
state	Egész	Fizetés státusza
<u>courseId</u>	Egész	Kurszus ID-je

vehicles – Járművek adatai

id	Egész	Jármű ID-je
plate	Szöveg	Jármű rendszáma
color	Szöveg	Jármű színe
driveType	Szöveg	Jármű váltó típusa
brand	Szöveg	Jármű gyártója
type	Szöveg	Jármű típusa
imageUrl	Szöveg	Jármű kép útvonala
<u>categoryId</u>	Egész	Kategória ID-je

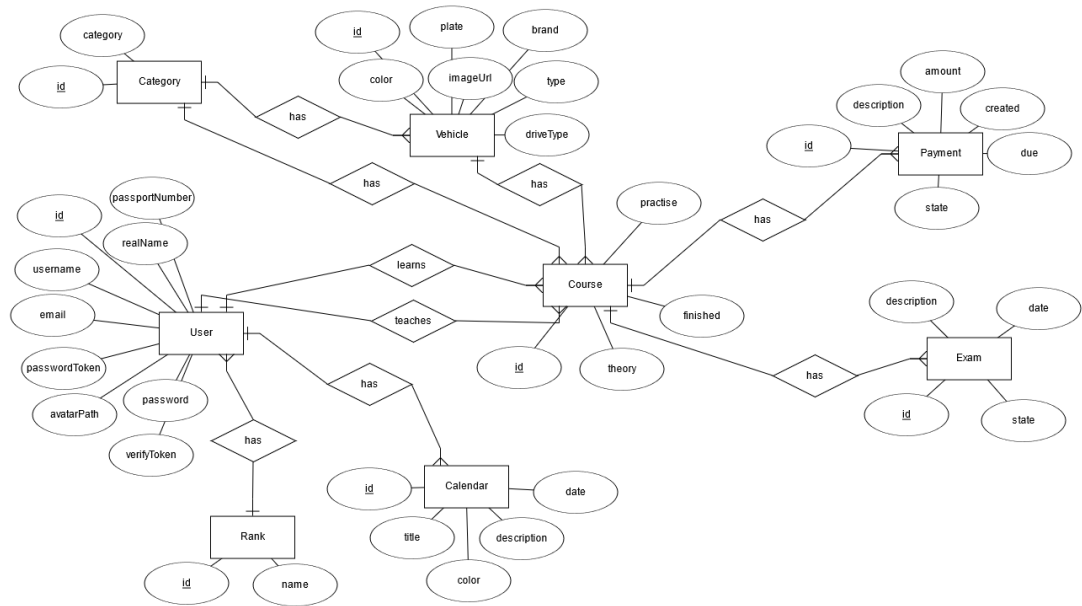
categories – Jogosítvány kategóriákat tartalmazó tábla

id	Egész	Kategória ID-je
category	Szöveg	Kategória neve

exams – Vizsgák adatai

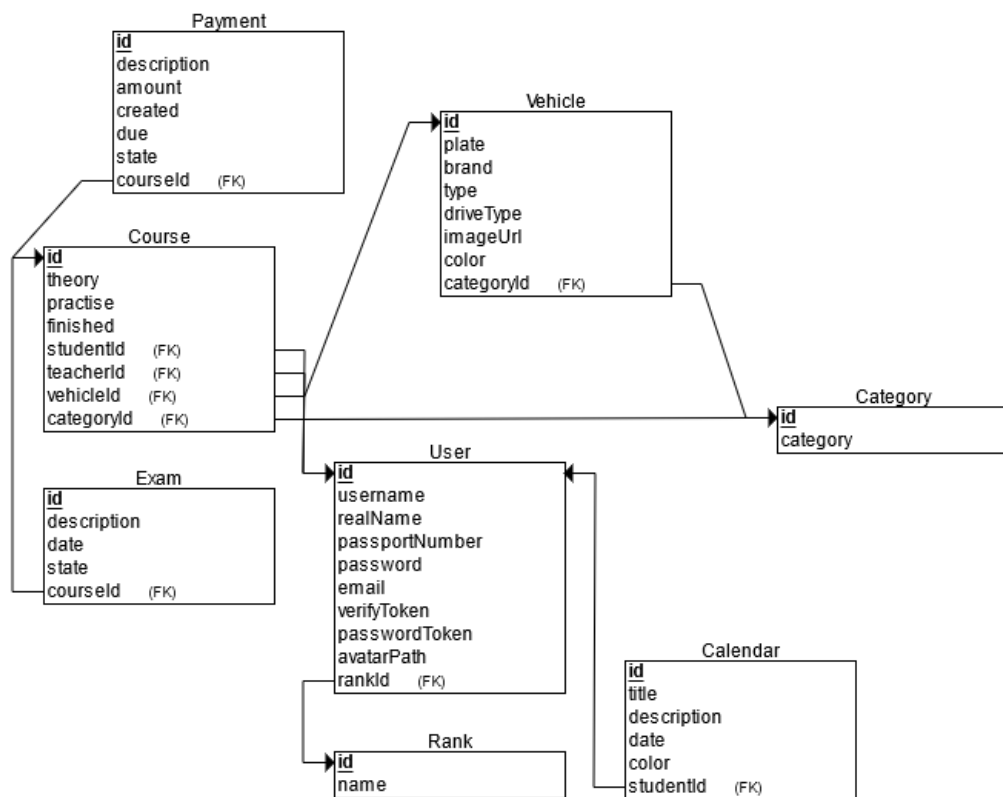
id	Egész	Vizsga ID-je
date	DateTime	Vizsga dátuma
description	Szöveg	Vizsga leírása
state	Egész	Vizsga státusza
<u>courseId</u>	Egész	Kurzus ID-je

3.7.2. ER modell



ábra 24: Adatbázis ER modell

3.7.3. Relációs séma



ábra 25: Adatbázis relációs séma

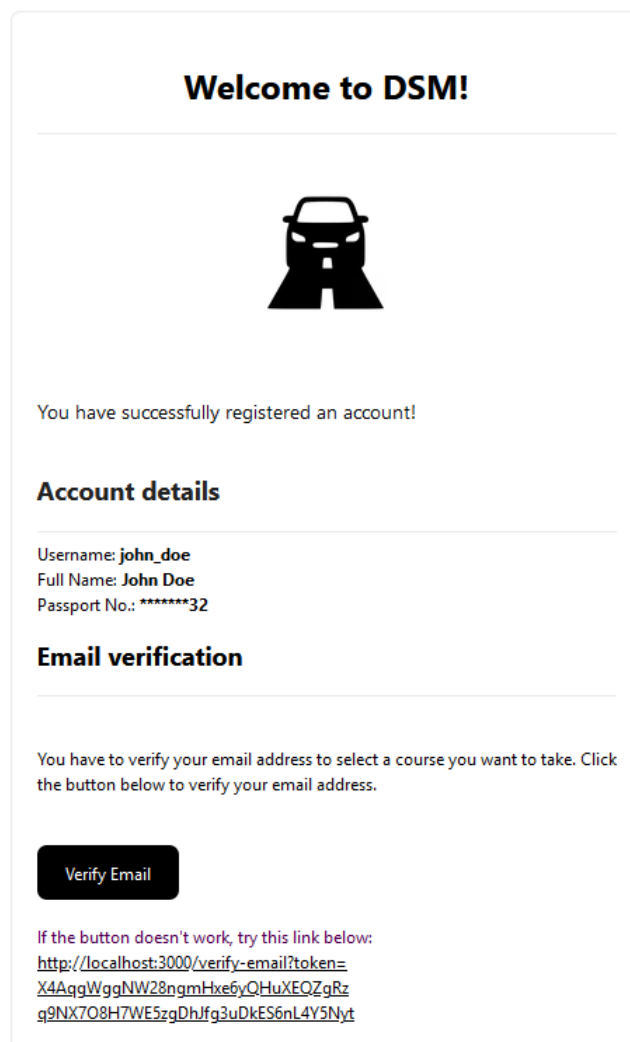
3.8. E-mail küldés

Az e-mail küldésért a Resend függőség felel. A Resend egy e-mail API, amely segítségével könnyedén tudunk saját domain segítségével leveleket küldeni bármilyen e-mail-re. A használatához szükség van egy saját domain-re, ezért a tulajdonomban lévő „sbcraft.hu” domain-ben létrehoztam egy aldomaint, amelyet használok ebben a projektben. A Resend létrehoz egy hozzáférési kulcsot, amely segítségével használni tudjuk az API-t.

3.8.1. A projektben használt levelek

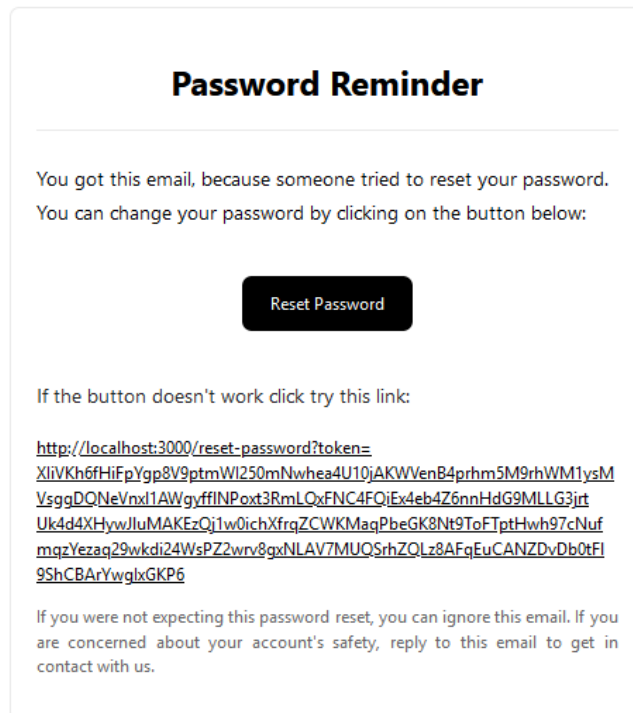
A leveleket React Email segítségével hoztam létre. A React Email egy React alapú könyvtár, amely segítségével könnyedén tudunk React használata segítségével paraméterezett levélmintákat létrehozni.

Sikeres regisztráció



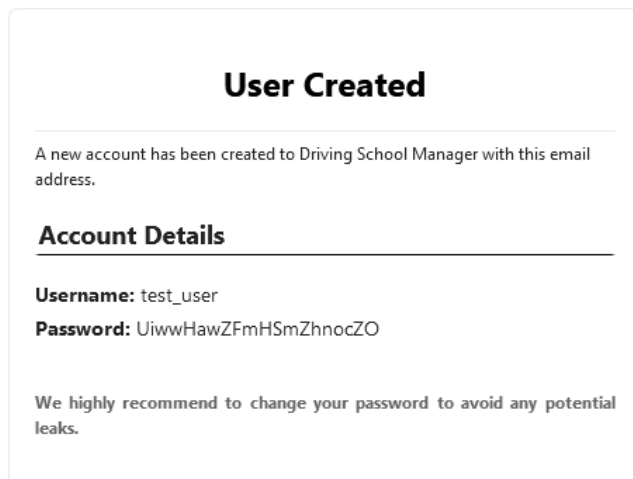
ábra 26: Sikeres regisztráció levél

Jelszó emlékeztető



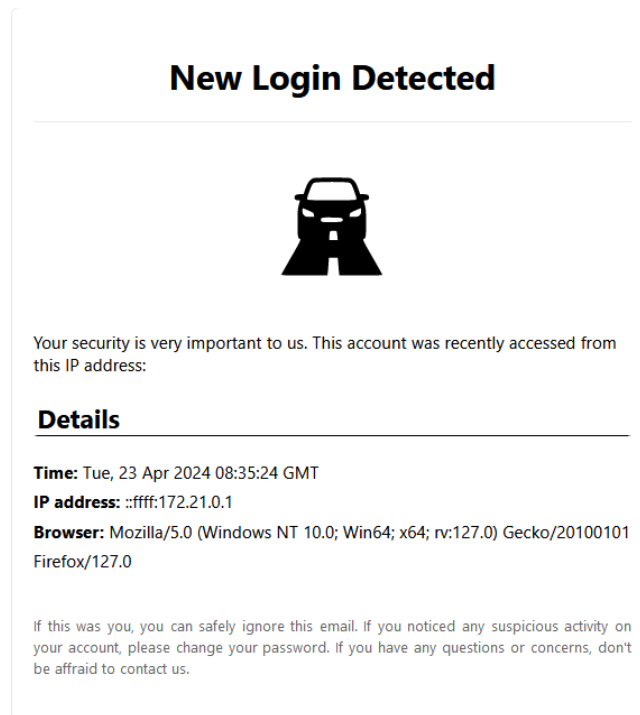
ábra 27.: Jelszó emlékeztető levél

Új felhasználó létrehozva (Admin által)



ábra 28.: Új felhasználó létrehozva levél

Új bejelentkezés



ábra 29: Új bejelentkezés levél

4. Tesztelési dokumentáció

A tesztelés egy nagyon fontos része az applikáció fejlesztésnek. Három féle tesztelést végeztem a projekten:

- Unit tesztek
- Funkcionális és UI tesztek
- SEO tesztek

A unit tesztek az applikáció legkisebb önálló egységeit tesztelik, hogy azok megfelelő eredményt adnak-e minden teszteset során.

A funkcionális és UI teszteket Katalon Studio-val végeztem el. A funkcionális tesztek során az oldalon használható valamennyi funkciót leteszteljük hibás és helyes adatokkal is.

A SEO (Search Engine Optimization) tesztek segítségével meg tudjuk állapítani, hogy egy weboldal mennyire teljesít jól a keresőmotorok találati listáján. Ezek a tesztek sok pontot lefednek, mint például a weboldal betöltési sebessége, telefonos kompatibilitás vagy tartalom minősége (kinézet, kontraszt). Egy weboldal minél nagyobb pontszámot ér el ezen a SEO teszten, annál nagyobb esélye van arra, hogy a keresőmotor azt az oldalt jelenítse meg a keresés után.

4.1. Unit tesztek

A unit teszteket a Bun beépített tesztelési könyvtárával végeztem el.

```
import { expect, test } from 'bun:test'
import { capitalizeLetter } from '@lib/utils'

// Első karakter nagybetűssé alakítás tesztek
tabnine: test | explain | document | ask
test("teszt felhasználó" → "Teszt felhasználó", () => {
  const str = capitalizeLetter('teszt felhasználó', false);
  expect(str).toBe('Teszt felhasználó');
});

tabnine: test | explain | document | ask
test("teszt felhasználó" → "Teszt Felhasználó", () => {
  const str = capitalizeLetter('teszt felhasználó', true);
  expect(str).toBe('Teszt Felhasználó');
});
```

ábra 30: capitalizeLetter unit tesztek

```

import { expect, test } from 'bun:test'
import { randomString } from '@lib/utils'

// Random string generálás tesztek
tabnine: test | explain | document | ask
test('10 karakter hosszú random string', () => {
  const str = randomString(10);
  expect(str.length).toBe(10);
});

tabnine: test | explain | document | ask
test('50 karakter hosszú csak számokat tartalmazó string', () => {
  Test Regex...
  const str = randomString(50, /[A-z]/g);
  expect(str.length).toBe(50);
  Test Regex...
  expect(str).toMatch(/[0-9]/g);
});

tabnine: test | explain | document | ask
test('20 karakter hosszú, csak kisbetűs karaktereket tartalmazó string', () => {
  Test Regex...
  const str = randomString(20, /[A-Z0-9]/g);
  expect(str.length).toBe(20);
  Test Regex...
  expect(str).toMatch(/[a-z]/g)
});

```

ábra 31: randomString funkció unit tesztek

```

bun test v1.1.3 (2615dc74)

randomString.test.ts:
✓ 10 karakter hosszú random string [0.54ms]
✓ 50 karakter hosszú csak számokat tartalmazó string [0.11ms]
✓ 20 karakter hosszú, csak kisbetűs karaktereket tartalmazó string [0.05ms]

capitalizeLetter.test.ts:
✓ "teszt felhasználó" → "Teszt felhasználó" [0.14ms]
✓ "teszt felhasználó" → "Teszt Felhasználó" [0.27ms]

5 pass
0 fail
7 expect() calls
Ran 5 tests across 2 files. [39.00ms]

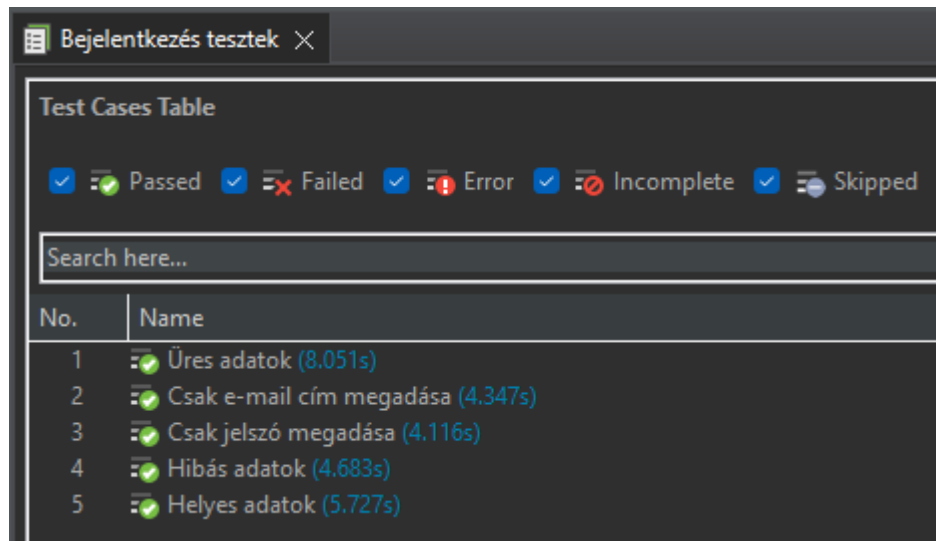
```

ábra 32: Unit tesztek eredményei

4.2. Funkcionális tesztek

A funkcionális teszteket a Katalon Studio szoftverrel végeztem el. A Katalon Studio egy tesztelő szoftver, amely kifejezetten automatizált tesztelésre használnak. Lehetővé teszi, hogy a felhasználók könnyedén készítsenek és végrehajtsanak teszteket webes, mobil alkalmazásokon, valamint API tesztelésre is használható.

A Katalon Studio-val végzett tesztek részletes leírása a forrásfájlok között a **tests/katalon** mappában találhatóak.



The screenshot shows the 'Bejelentkezés tesztek' window in Katalon Studio. It features a 'Test Cases Table' with a search bar and a list of five test cases, all of which have passed. Each test case entry includes a status icon (a green checkmark), a name, and a duration in seconds.

No.	Name
1	Üres adatok (8.051s)
2	Csak e-mail cím megadása (4.347s)
3	Csak jelszó megadása (4.116s)
4	Hibás adatok (4.683s)
5	Helyes adatok (5.727s)

ábra 33: Bejelentkezés tesztek

5. Összegzés

A projekt során számos dolgot megtanultam a React Hookok megfelelő használatáról és a Next.js nyújtotta funkciókról, valamint számos újdonságot tanultam a JavaScript használatával kapcsolatban is, hogy hogyan tudok kevesebb kóddal hatékonyabb eredményt elérni. A TypeScriptet is ebben a projektben használtam először komolyabban. Az egyedi típusok és interfészek néha fejtörést okoztak, amikor komplex interfészeket kellett létrehozni egy-egy adatbázis lekérdezéshez, de hogy TypeScriptet választottam a sima JavaScript helyett hosszútávon megkönnyítette a fejlesztés folyamatát. Nagyon egyszerűvé tette a hibák keresését, valamint segített az automatikus kiegészítésben is.

A shadcn/ui, amit a weboldal designolására használtam szintén egy új dolog volt, amit már régen meg szerettem volna tanulni. A komponensek telepítése és a felhasználás nagyon egyszerű volt. Könnyedén személyre szabható az összes komponens, ezért nagyon egyszerű a használatuk. Viszont az egyes shadcn komponensek egymásba ágyazása már bonyolultabb volt. Azok a komponensek, amelyek megjelentek, mint például felugró ablakok vagy oldalról becsúszó menük, akkor jöttek létre, amikor rákattintottam a gombra és mikor becsukódtak, akkor törlődtek, ezért, ha a szerver kicsit lassabban válaszolt, volt mikor nem jelent meg egy hibaüzenet, mert a funkció, ami kezelte a hiba megjelenítését a válasz megérkezésekor már nem létezett.

Voltak olyan komponensek, amelyek kötelezően csak kliens oldalon létezhetnek, ezekbe a komponensekbe elég nehéz volt szerver oldali funkciókat beilleszteni, de ezt a problémát is sikerült áthidalnom.

A legnehezebb feladatot a grafikonok személyre szabása jelentette számomra. A bár részletes, de nehezen megérthető dokumentáció miatt voltak olyan esetek, mint például a radar diagramm esetében, hogy az a paraméter, ami a dokumentációban szerepelt, az a letöltött könyvtárban már más néven volt jelen. A másik nagy fejtörést az okozta, amivel akkor szembesültem, mikor elkészítettem az első production buildet a projektből. A Next.js buildeléskor létrehozza az összes statikus oldalt és attól a ponttól kezdve nem tud helyileg képeket megjeleníteni. Bár a kép a megfelelő helyen van, a Next.js nem tudja megjeleníteni azt. Ezt a problémát úgy sikerült megoldanom, hogy létrehoztam egy web-szervert, ami csak arra szolgál, hogy ezeket a képeket küldi vissza a Next.js számára, amelyeket már meg tud jeleníteni.

A Docker, a Dockerfile és compose.yml megfelelő használatát is a projekt során tanultam meg. A kezdeti 4-5 GB-os képfájlt sikerült optimalizálással lefaragnom a projekt tényleges méretére, 1,7 GB-ra.

Szerintem sikerült a lehető legtöbbet kihoznom az általam használt keretrendszerből és az összes függőségből, amit az adatok kezelésére és megjelenítésére használtam.

5.1. Továbbfejlesztési lehetőségek

Természetesen mindig van hely bővítésnek és végtelenségig lehetne tovább fejleszteni az applikációt. Próbáltam a lehető legjobban törekedni a skálázhatóságra.

Néhány ilyen fejlesztési lehetőség lehet:

- Részletesebb kurzus leírás a tanulók számára.
- Kurzussal kapcsolatos statisztikák a tanulók számára.
- Bővített statisztika az admin felhasználók számára.
- Statisztika az oktatók számára.
- Vizsgák részletes nyomon követése.
- Oktatók értékelése a kurzus befejeztével.

6. Ábrajegyzék

ábra 1: Admin felhasználó adatai	3
ábra 2: Regisztráció oldal	3
ábra 3: Regisztráció űrlap	4
ábra 4: Bejelentkezés oldal	4
ábra 5: Bejelentkezés űrlap	5
ábra 6: Elfelejtett jelszó oldal	5
ábra 7: Felhasználói felület admin felhasználóval bejelentkezve	6
ábra 8: Főoldal oktató felhasználóval bejelentkezve	6
ábra 9: Kurzusok oldal	6
ábra 10: Naptár oldal oktató felhasználóval bejelentkezve	7
ábra 11: Új esemény felvétele	8
ábra 12: Statisztika oldal	8
ábra 13: Felhasználók oldal	9
ábra 14: Új felhasználó felvétele	9
ábra 15: Tanulók oldal	10
ábra 16: Oktatók oldal	10
ábra 17: Járművek oldal	11
ábra 18: Új jármű felvétele	11
ábra 19: Vizsgák oldal	12
ábra 20: Új vizsga felvétele	12
ábra 21: Fizetések oldal	13
ábra 22: Új fizetés felvétele	13
ábra 23: Saját profil oldal	14
ábra 24: Adatbázis ER modell	28
ábra 25: Adatbázis relációs séma	28
ábra 26: Sikeres regisztráció levél	29
ábra 27: Jelszó emlékeztető levél	30
ábra 28: Új felhasználó létrehozva levél	30
ábra 29: Új bejelentkezés levél	31
ábra 30: capitalizeLetter unit tesztek	32
ábra 31: randomString funkció unit tesztek	33
ábra 32: Unit tesztek eredményei	33

ábra 33: Bejelentkezés tesztek	34
--------------------------------------	----