**PROJEKTMUNKA**

Pályakövető rendszer weboldal

**Mentorok: Készítette:**

Szabó Ákos Menyhárt Martin

Dantesz Tamás Kiss Szabolcs

Labbancz Balázs

NYÍREGYHÁZA

2023

Tartalomjegyzék

[Feladat megadása 2](#_Toc133006416)

[1. Felhasználók és igényeik / Funkció lista 2](#_Toc133006417)

[1.1 Látogató 2](#_Toc133006418)

[1.2 Bejelentkezett felhasználó: 2](#_Toc133006419)

[1.3 Adminisztrátor 3](#_Toc133006420)

[2. Screenek 3](#_Toc133006421)

[3. Modellek 5](#_Toc133006422)

[4. Alkalmazások kiválasztása 8](#_Toc133006423)

[4.1 Front-end 8](#_Toc133006424)

[4.2 Backend 9](#_Toc133006425)

[4.3 Adatbázis 10](#_Toc133006426)

[4.4 Tesztelés 10](#_Toc133006427)

[5. Routing 10](#_Toc133006428)

[6. Implementáció 11](#_Toc133006429)

[7. Tesztelés 11](#_Toc133006430)

[8. Bevezetés, éles üzemmód 12](#_Toc133006431)

# **Feladat megadása**

Vizsgaremekünkben a Nyíregyházi SZC egyik új projektjét szeretnénk megvalósítani, melynek keretében egy középiskola utáni pályakövetési rendszert fogunk elkészíteni. Projektünkben egy weboldalt szeretnénk megvalósítani, amely minden felhasználója számára képes megjeleníteni, hogy ki, mikor és melyik középiskolát fejezte be és ezután hol állt munkába, vagy melyik egyetemen tanul tovább. Fontos megemlítenünk, hogy felhasználó csak SZC dolgozó lehet, így regisztrációra nem lesz lehetőség és csak az adminisztrátori joggal rendelkezők vehetnek fel új felhasználókat. Ez azt is jelenti, hogy az oldal nem publikus és tartalma csak bejelentkezés után érhető el. A felhasználóknak lehetősége lesz módosítani az adatbázist és különböző szempontok alapján szűrni azt.

# **1. Felhasználók és igényeik / Funkció lista**

Az alkalmazásban a felhasználók tanárok lesznek, akiket az OM azonosítójuk azonosít. Regisztráció olyan szinten létezik, hogy csak az admin szintű felhasználók tudnak a tanároknak fiókot létrehozni, előre megadott jelszavakkal, amiket a felhasználó később bármikor megváltoztathat. A tanárok csakis a saját osztályaikat láthatják, amikkel bejelentkezés után az alábbi műveleteket hajthatják végre.

## Látogató

* Legyen képes bejelentkezni

## Bejelentkezett felhasználó:

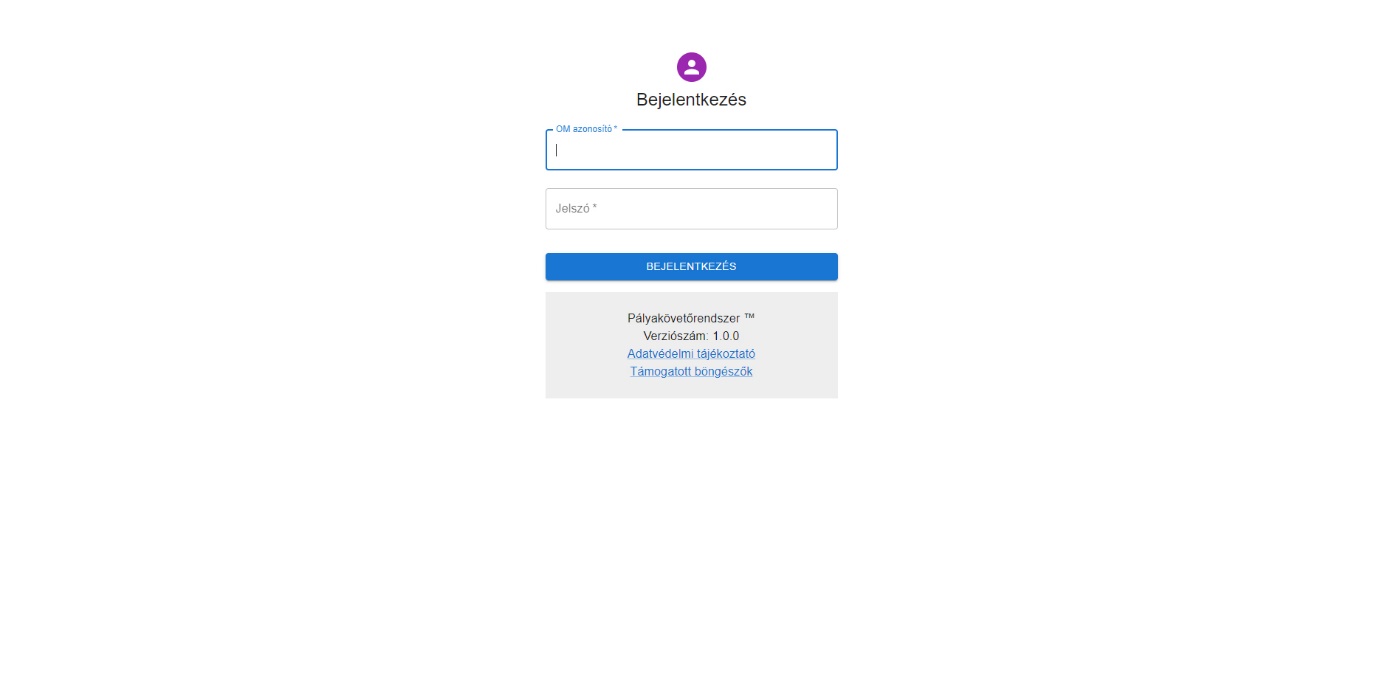
* Legyen képes osztályt választani.
* Legyen képes hozzáadni, módosítani, törölni, vagy különböző szempontok alapján szűrni a kiválasztott osztály tanulóit.
* Legyen képes importálni CSV fájlból tanulókat a kiválasztott osztályba.
* Legyen képes exportálni Excel és CSV fájlba kijelölt tanulókat és azok adatait a kiválasztott osztályból.
* Legyen képes új, üres osztályt létrehozni.
* Legyen képes jelszót változtatni.
* Legyen képes kapcsolatfelvételre az oldal üzemeltetőjével.
* Legyen képes sötét módba váltani az oldalt.
* Legyen képes kijelentkezni.

## Adminisztrátor

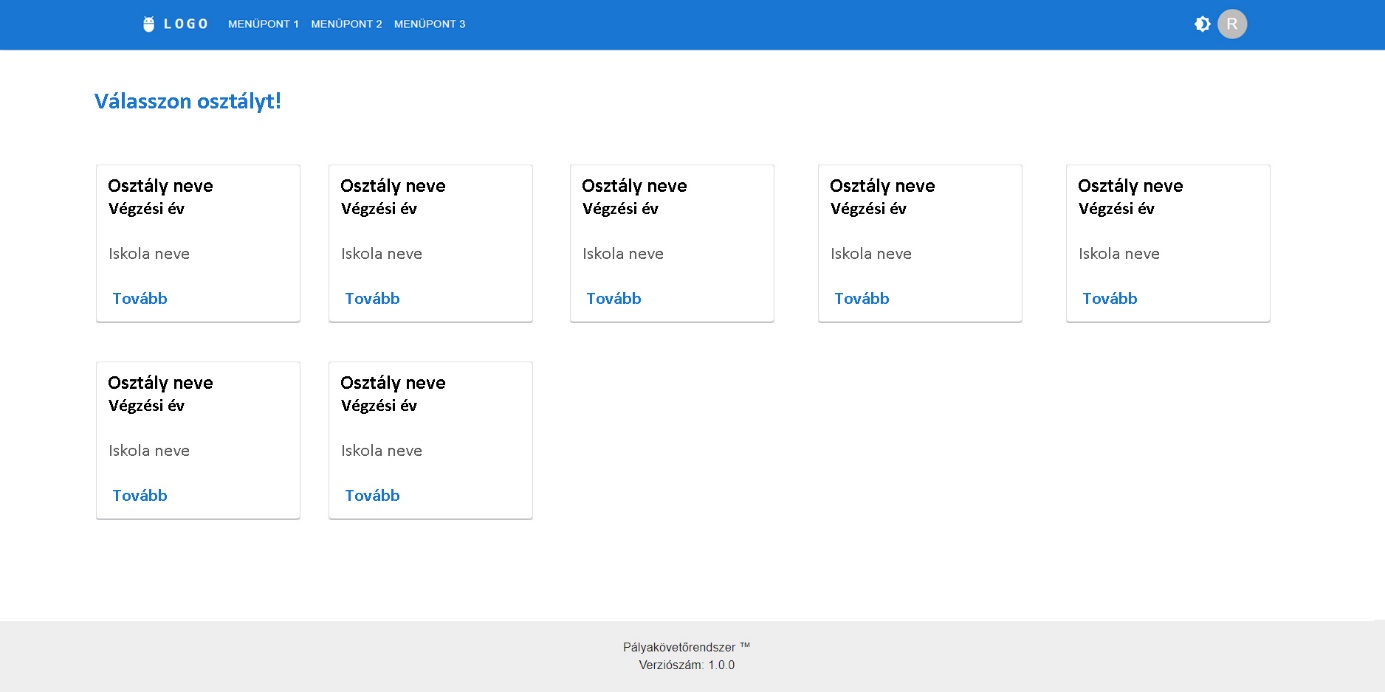
* Legyen képes új felhasználókat felvenni.
* Legyen képes létező felhasználók adatait módosítani.
* Legyen képes saját jelszót változtatni.

# 2. **Screenek**

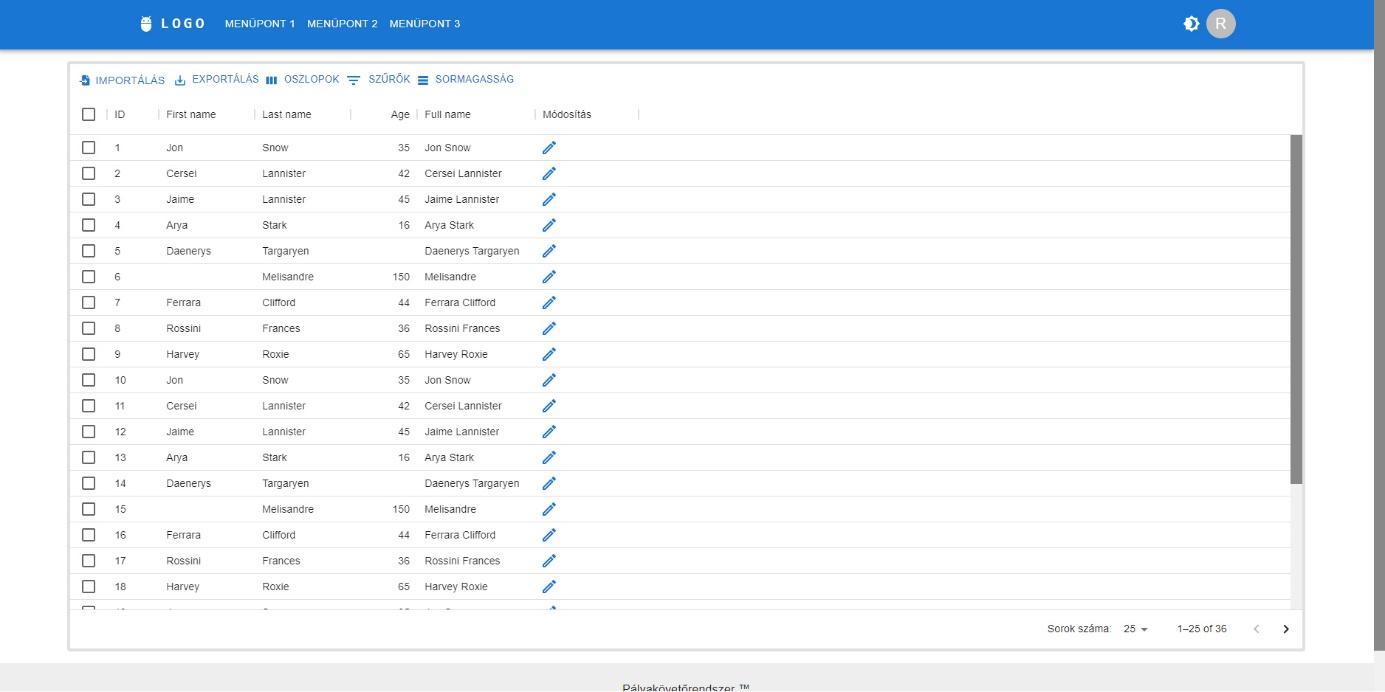
Letisztult megjelenésű bejelentkező oldal:



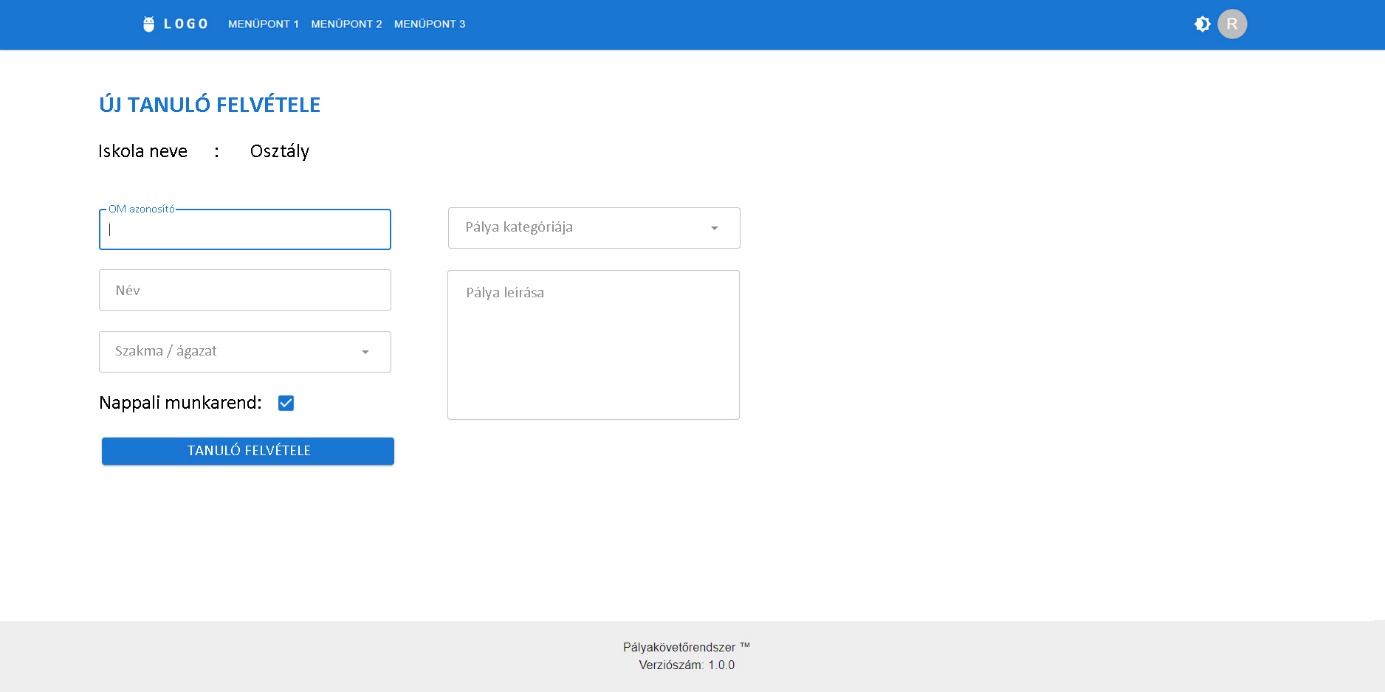
Bejelentkezés után az osztályválasztó felület:



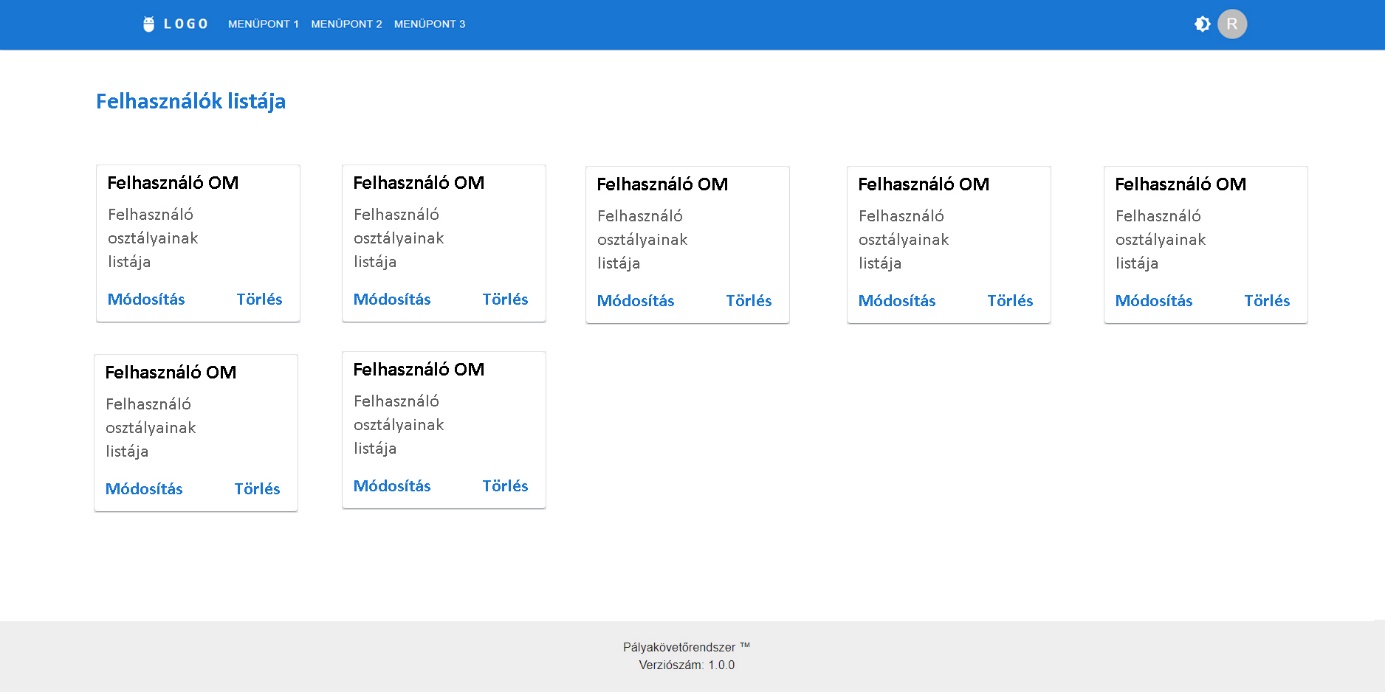
Egy kiválasztott osztály, ami tartalmazza az osztályba tartozó tanulókat:



Egy új tanuló felvétele az adott osztályba, manuálisan:



Az admin felület tartalmazza a felhasználókat, amikkel törlési és módosítási műveletek végezhetőek el:



# **3. Modellek**

Az adatok tárolására több modellre van szükség: iskola, felhasználó, osztály, tanuló, ágazat, szakma, és kategória.

**iskola** {

* id: int (2)
* nev: varchar (255)

}

**felhasznalo** {

* om\_azon: bigint (11)
* jelszo: varchar (60)
* admin: tinyint (1)

}**osztaly** {

* id: int (4)
* iskolaid: int (2)
* felhasznalo\_om: bigint (11)
* nev: varchar (255)
* vegzesi\_ev: int (4)

}

**tanulo** {

* om\_azon: bigint (11)
* nev: varchar (255)
* osztalyid: int (4)
* nappali\_munkarend: tinyint (1)
* agazatid: int (3)
* szakid: int (3)

}

**palya** {

* id: int (11)
* diak\_om\_azon: bigint (11)
* kategoriaid: int (2)
* leiras: varchar (255)

}

**agazat** {

* id: int (3)
* nev: varchar (255)
* szam: varchar (255)

}

**szakma** {

* id: int (3)
* nev: varchar (255)
* szam: varchar (255)

}

**kategoria** {

* id: int (2)
* megnevezes: varchar (255)

}

# 4. Alkalmazások kiválasztása

## 4.1 Front-end

* **React**

A React az egyik nyílt forráskódú JavaScript könyvtár. Interaktív felhasználói felületek felépítésére szolgál. Ez egy hatékony, deklaratív és rugalmas könyvtár. A Model-View- Controller (MVC) V, azaz View elemével foglalkozik. Ez nem egy teljes keret, hanem csak egy előlapi könyvtár. Lehetővé teszi összetett felhasználói felületek létrehozását vagy elkészítését, összetevőkként ismert, elkülönített és apró kóddarabok felhasználásával.

(<https://hu.education-wiki.com/9050114-what-is-react>)

* **Javascript**

A JavaScript (röviden JS) egy objektumalapú scriptnyelv, amelyet elterjedten használnak weboldalakon, a web programozási nyelve. A Javascript futásideje egyszálú, ami azt jelenti, hogy egyszerre csak egy darab kódot képes végrehajtani. Ennek a megoldására használunk async függvényeket.

* **Html**

A HTML (Hypertext Markup Language) egy leíró nyelv, melyet weboldalak elkészítésére használhatunk.

* **MUI**

A MUI egy nyílt forráskódú keretrendszer (framework) amely egy React alapú komponens könyvtár, ami a Google Material Design elveit implementálja**.** Nagyon könnyen, és minimális energia befektetéssel tudunk jól kinéző, bármilyen képernyőméreten szépen megjelenő weboldalakat készíteni a MUI segítségével.

* **axios**

Az Axios egy könyvtár, amelyet HTTP -kérések küldésére használnak a böngészőből Node és Express.js platformon keresztül.

## 4.2 Backend

* **Node.js**

A Node.js egy nyíltforrású, többplatformos, szerveroldali JavaScript futtatókörnyezet a **Google Chrome V8 JavaScript***-*motoron alapulva. Leginkább webes applikációk, dinamikus weboldalak fejlesztésére használják, de akár szerveroldali feldolgozó-szkriptek készítésére is kiválóan alkalmas.

(http://maas.hu/learning/it/prog/javascript/nodejs/#gsc.tab=0)

* **Express**

Az **Express** (amit gyakran **Express.js**-nek, vagy **ExpressJS**-nek is hívnak) egy szerver oldali (back end) applikáció fejlesztői keretrendszer Node.js-hez. Használata és terjesztése ingyenes, szabad szoftver az MIT licensze alatt. Elsősorban webes applikációkat és API-kat fejleszthetünk vele, és rendkívül sikeres életútja során viszonylag hamar az általános webes applikációs keretrendszerré vált Node.js alatti fejlesztések esetében. (http://maas.hu/learning/it/prog/javascript/express/)

* **Bcrypt** – hashelés (jelszó)

A Bcrypt egy népszerű és megbízható módszer a jelszavak kivonására.

* **Nodemon – no restart**

A nodemon egy olyan eszköz, amely segít a **Node.js** alapú alkalmazások fejlesztésében azáltal, hogy automatikusan újraindítja a csomópont-alkalmazást, ha a rendszer fájlváltozásokat észlel a könyvtárban.

* **MySQL – Node Module**

A MySQL egy rendkívül népszerű **relációs adatbázis-kezelő rendszer***.* A MySQL-t gyakran használják adatbázisként a Node-dal épített projektekhez és alkalmazásokhoz.

## 4.3 Adatbázis

* **XAMPP**

Egy platformfüggetlen webszerver-szoftvercsomag, azok számára, akik adatbázissal támogatott webszervert szeretnének üzemeltetni, a webes alkalmazások készítését, tesztelését és futtatását oldja meg.

* **MariaDB**

A MariaDB Server az egyik legnépszerűbb nyílt forráskódú relációs adatbázis.

## 4.4 Tesztelés

* **Postman**

A Postman egy komplett eszköztár API fejlesztők számára. Olyan eszköz, amelyet az API-k megbízható teszteléséhez használhatunk. A Postman leegyszerűsíti az API életciklusának minden lépését, és egyszerűsíti az együttműködést.

# 5. Routing

**POST auth/register**

|  |  |
| --- | --- |
| tervezett feladat | Regisztráció |
| bemenet | OM azonosító, jelszó |
| kimenet | Visszajelzés a sikerességről |
|  |  |

**POST auth/login**

|  |  |
| --- | --- |
| tervezett feladat | Bejelentkezés |
| bemenet | OM azonosító, jelszó |
| kimenet | Felhasználónév és egy jogosultságot meghatározó érték (isAdmin), sikerességről való visszajelzés |

**GET auth/logout**

|  |  |
| --- | --- |
| tervezett feladat | Kijelentkezés |
| bemenet | - |
| kimenet | Visszajelzés a sikerességről |

**GET classes/class\_chooser**

|  |  |
| --- | --- |
| tervezett feladat | Osztályok kilistázása, ami az adott felhasználóhoz tartozik |
| bemenet | - |
| kimenet | Osztályok renderelt listája |

**POST classes/createClass**

|  |  |
| --- | --- |
| tervezett feladat | Üres osztály létrehozása |
| bemenet | Az iskola id-je, a tanár OM azonosítója, az osztály neve, és az osztály végzési éve |
| kimenet | Az új üres osztály kártyájának kirenderelése |

# 6. Implementáció

Az implementáció a tervezést követő folyamat. A tervezés során megtörténik a feladatokra bontás, valamint az egyes feladatokban érintett funkcionalitás és szerkezet modellezése. Az implementáció során a modellekből készítünk működő, futtatható kódot.   
(https://okt.sed.hu/rf2/gyakorlat/tananyag/implementacio/implementacio/)

# 7. Tesztelés

A felhasználó szempontjából a rendszer futtatásához elegendő bármely olyan operációs rendszer, amelyen futtatható HTML, CSS és JavaScript technológiával készült alkalmazás megjelenítésére képes böngésző.

A fejlesztés alatt a rendszer tesztelése asztali számítógépeken, illetve laptokon több böngészőben (Google Chrome 112.0.5615.138, Microsoft Edge 112.0.1722.48) ráadásul több asztali operációs rendszereken, név szerint Windows 10 Professional és Windows 11 Profession operációs rendszerek alatt történt. Továbbá mobil eszközön Android és iOS operációs rendszer alatt is teszteltük.

A szerveroldalról nézve a rendszernek szüksége van egy olyan platformra, ami MySQL szervereket képes futtatni, a fejlesztés során egy helyi localhost-al oldottuk meg.

A feladat végrehajtása során folyamatosan jelen volt a tesztelés. Már a statikus oldal megtervezésekor és készülésekor tesztelve volt a megjelenés reszponzivitása, az olvashatóság, a láthatóság.

Az adatbázis tartalommal feltöltése után az oldal tartalmi része is tesztelve, ellenőrizve volt. Ezt Postman-ban valósítottuk meg, amely azonnali visszajelzést adott a fejlesztés sikerességéről, esetleges hibáiról. Ilyen volt például a felhasználók jelszavának tesztelése:

Új felhasználó felvétele esetén (POST kérés) a jelszót hash-eljük. A fejlesztés során előre felvett admin felhasználónak például Admin\_123 jelszót adtunk meg, melyet átalakítottunk. A bejelentkező oldal esetén a frontend beküldi az adatot, a backend hash-eli, és a hash-elt jelszó adott pillanatban egyezik az adatbázisból kivett felhasználóhoz tartozó jelszóval, akkor a felhasználót átengedi.

# 8. Bevezetés, éles üzemmód

* Az elkészült munka megtekintéséhez szükséges a XAMPP letöltése. Látogassunk el a következő oldalra: <http://www.apachefriends.org/en/xampp.html>, és onnan töltsük le az operációs rendszerünknek megfelelő XAMPP-verziót.
* Telepítsük fel. A letöltött futtatható állományt indítsuk el, és kövessük a képernyőn megjelenő utasításokat. Windows alatt a javasolt telepítési könyvtár c:\xampp.
* töltse le a Windows telepítőt a Node.js webhelyről. Miután kiválasztotta az igényeinek megfelelő verziót, futtassa a telepítőt.
* Indítunk kettő Visual Studio Code alkalmazást, egyikben a frontend mappát nyitjuk meg, a másikban a backend mappán belül a server mappát kell megnyitni az Open Folder menüponttal.
* mindkét Visual Studio Code-ban indítunk egy Terminált a Terminal menu/New Terminal-lal. (Vagy a Command Line is megfelel)
* Sikeres telepítések után a xampp-ot indítsuk el és az Apache-t és MySQL-t is Startoljuk, és a MySQL Admin gombját válasszuk.
* A phpMyAdmin-ba importáljuk be a kapott adatbázist.
* Mindkét Visual Studio Code-ban a Terminal-ban npm -i --legacy-peer-deps paranccsal telepítsük az összes modult.
* Mindkét Visual Studio Code-ban npm start paranccsal indítsuk az alkalmazást.