

## Szent István Katolikus Technikum és Gimnázium

Oktatás Segítő Alkalmazás

Dokumentáció

Sike István, Szabadka Konrád, Hadari Dávid

Tartalom

[Szent István Katolikus Technikum és Gimnázium 0](C:\\Users\\Rebi\\Downloads\\OSA_Documentáció.docx" \l "_Toc163512587)

[Mi az az OSA? 3](#_Toc163512588)

[Funkciói 3](#_Toc163512589)

[Felhasználói dokumentáció 4](#_Toc163512590)

[Bejelentkezés: 4](#_Toc163512591)

[Admin: 5](#_Toc163512592)

[Tanár: 5](#_Toc163512593)

[Diák: 6](#_Toc163512594)

[Használt technikák 6](#_Toc163512595)

[Frontenden használtak: 6](#_Toc163512596)

[Backenden használtak 7](#_Toc163512597)

[Adatbázishoz használtak 7](#_Toc163512598)

[Adatbázis 8](#_Toc163512599)

[**Adatbázis** 8](#_Toc163512600)

[Frontend szerver 15](#_Toc163512601)

[Felhasználói felület 15](#_Toc163512602)

[Logó 18](#_Toc163512603)

[Színek 20](#_Toc163512604)

[Script 20](#_Toc163512605)

[CSS 22](#_Toc163512606)

[Backend server 25](#_Toc163512607)

[Backend ismertetése 25](#_Toc163512608)

[Backend Fájljai 25](#_Toc163512609)

[app.js 26](#_Toc163512610)

[db.js 28](#_Toc163512611)

[dbModels mappa 28](#_Toc163512612)

[Routes mappa 29](#_Toc163512613)

[Controllers mappa 30](#_Toc163512614)

[logRegisterController.js 30](#_Toc163512615)

[adatmodositasController.js 33](#_Toc163512616)

[mulasztasokController.js 35](#_Toc163512617)

[osztalyzatokController.js 36](#_Toc163512618)

[profilController.js 38](#_Toc163512619)

[oraDokumentalas.js 42](#_Toc163512620)

[Felhasználó hitelesítés 43](#_Toc163512621)

[middleware/hitelesitesMiddle.js 43](#_Toc163512622)

[Fekete lista 44](#_Toc163512623)

[Backend végpont táblázat 45](#_Toc163512624)

[Tesztelés 49](#_Toc163512625)

[Használt tartalom 49](#_Toc163512626)

# Mi az az OSA?

Napjaink erősen digitalizált világában az oktatást sem hathatjuk ki a korszerűsítésből ezzel a céllal állt neki eme feladatnak a csapatunk hogy alkosson erre a feladatra alkalmas szoftvert. Így született meg az OSA egy alkalmazás amely webes, asztali és telefonos alkalmazásként futtatva egy olyan felületet biztosít a diákoknak és tanároknak ami megkönnyíti a mindennapjaikat az oktatásban mint például: gyorsan elérhetik a számukra fontos információkat vagy könnyedén eltudják végezni az adminisztrációs feladataikat.

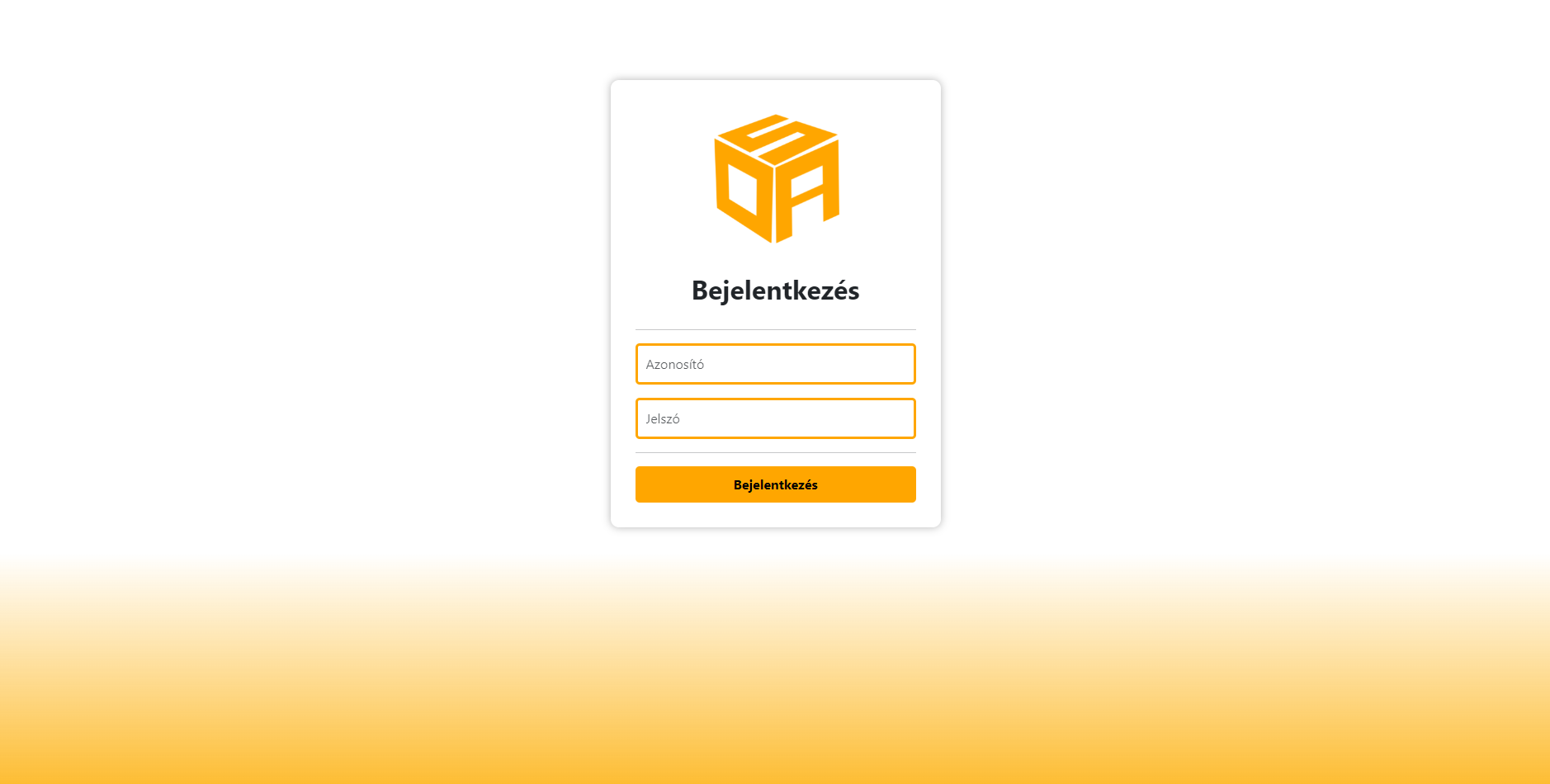
# Funkciói

* „jelenlegi óra”
  + Egy értesítő hogy a felhasználónak milyen órája van jelenleg és lesz a következő feltéve ha van neki
* Osztályok nyilvántartása
  + Vizuálisan felsorakoztatja a diákokat ahol a szaktanár és az osztályfőnök a szerepétől függően tud interakciót folytatni a diákkal (adatok lekérése, üzenet küldése) ki jelzi a hiányzó és a hetes diákokat is egyaránt
* Mulasztások kezelése
  + A mulasztások feljegyzése ehhez ha van akkor indok vagy igazolás hozzáfűzése (pl.: iskolai indokú hiányzás, betegség, családi indokból való távollét vagy ismeretlen). Plusz funkció hogy a diák küldhet az OSA felületén keresztül egy igazolást képpel, szöveggel.
* Tanóra dokumentálása
  + Tanóra megtartásáról egy dokumentálás ami tartalmazza a tantárgyat, az óra témáját, hiányzókat. Ha az órát nem dokumentálja a tanár akkor jelzést küld ha ennek ellenére sem dokumentálja le egy alapértelmezett dokumentálást kap az óra.
* Értékelések
  + jegyek beírása, azok nyilvántartása, átlag és súlyozott átlag számítása
* Órarend
  + tanáronként és diákokként megnézhető órarend

# Felhasználói dokumentáció

## Bejelentkezés:

Az OSA sikeres indításakor a bejelentkező felület jelenik meg a monitoron.

A bejelentkezéshez szükséges adatokat a narancssárga kerettel megjelölt mezőkbe kell begépelni. A felső mezőbe - amelyben alapértelmezett az „Azonosító” szöveg szerepel - kell beírni az azonosítóját, azaz az OM azonosítót. Az ez alatt található mezőbe - amelyben a „Jelszó” felirat szerepel – kell beírni a jelszót ami első belépéskor a település irányító száma, ahol lakik. Ezt később a belépés után a profiladatokon belül erősen ajánlott megváltoztatni!

Bejelentkezés gomb

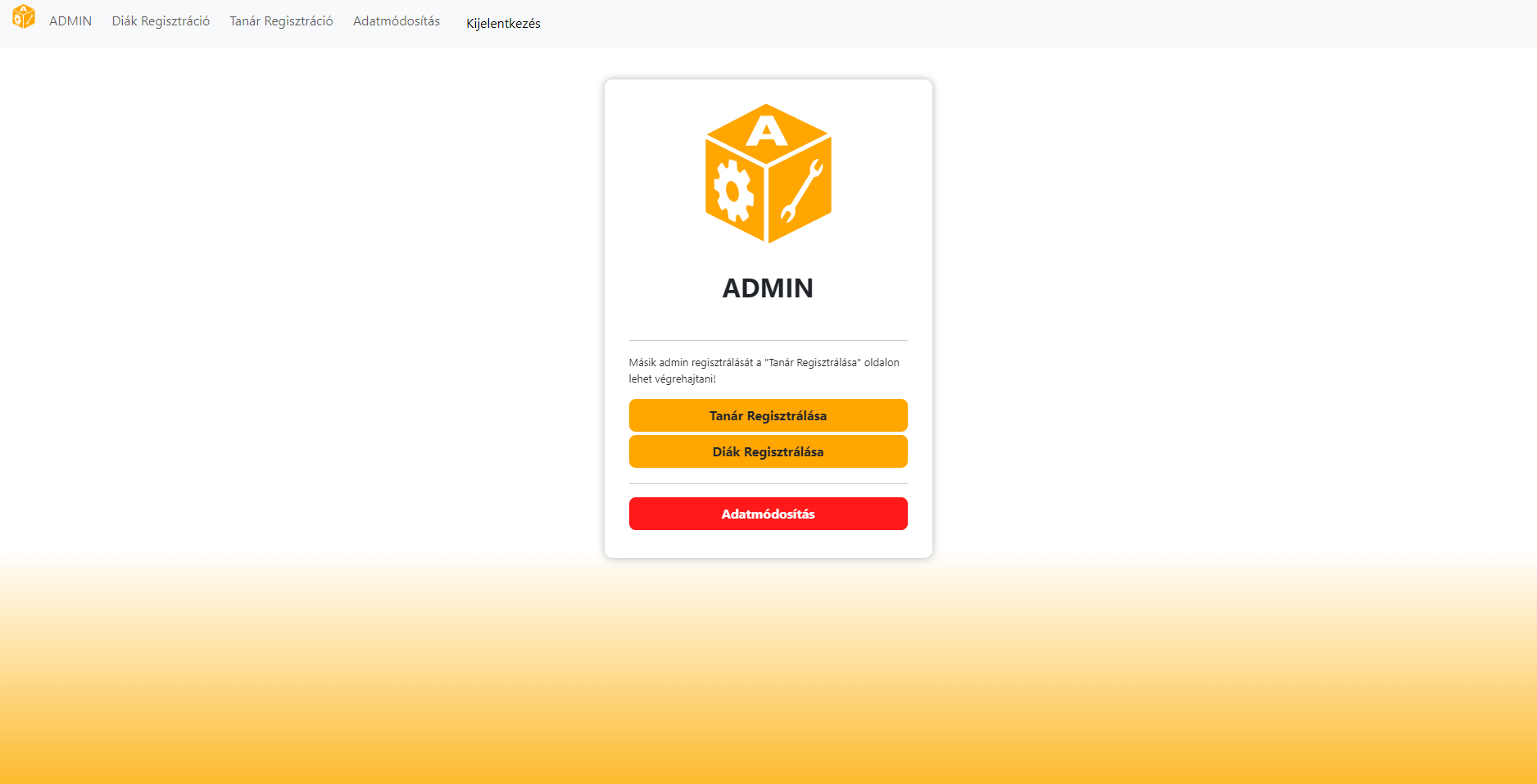
Jelszó

Azonosító

Az adatok megfelelő és sikeres bevitele után a bejelentkezés gombra kattintva a megfelelő profil jelenik meg számunkra. A bejelentkezés gomb a legalsó, teljesen narancssárga színű téglalap a „bejelentkezés” felirattal ellátva.

## Admin:

Ha admin ként regisztrálták, bejelentkezés után a következő felületet kell látnia:



Menü sor

Adatmódosítás

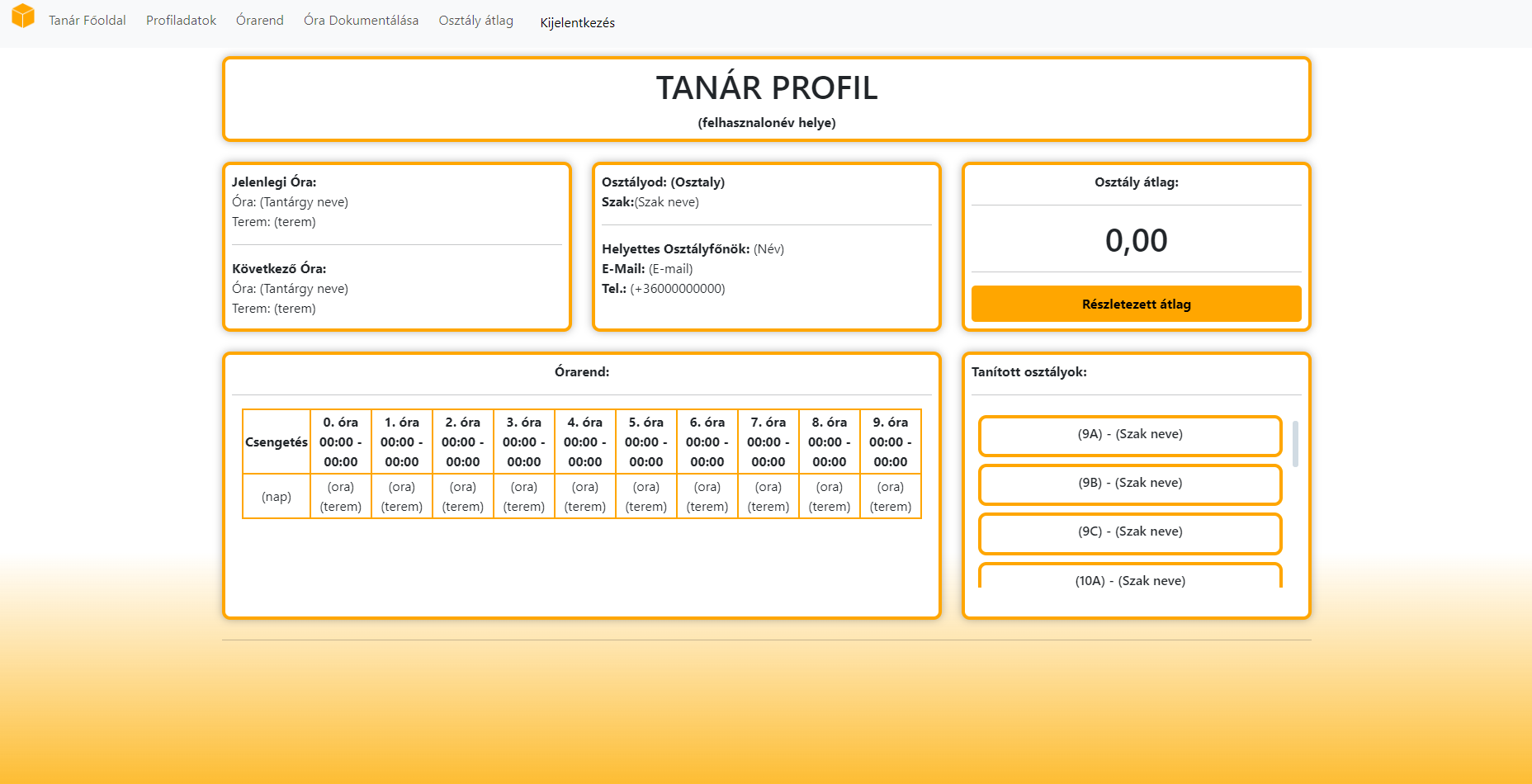
Diák regisztrálása

Tanár regisztrálása

Itt lehetséges a tanárok, diákok és újjabb adminok regisztrálása a rendszerbe, ugyanakkor már meglévő felhasználók adatai is módosíthatók. A felső narancssárga téglalap a

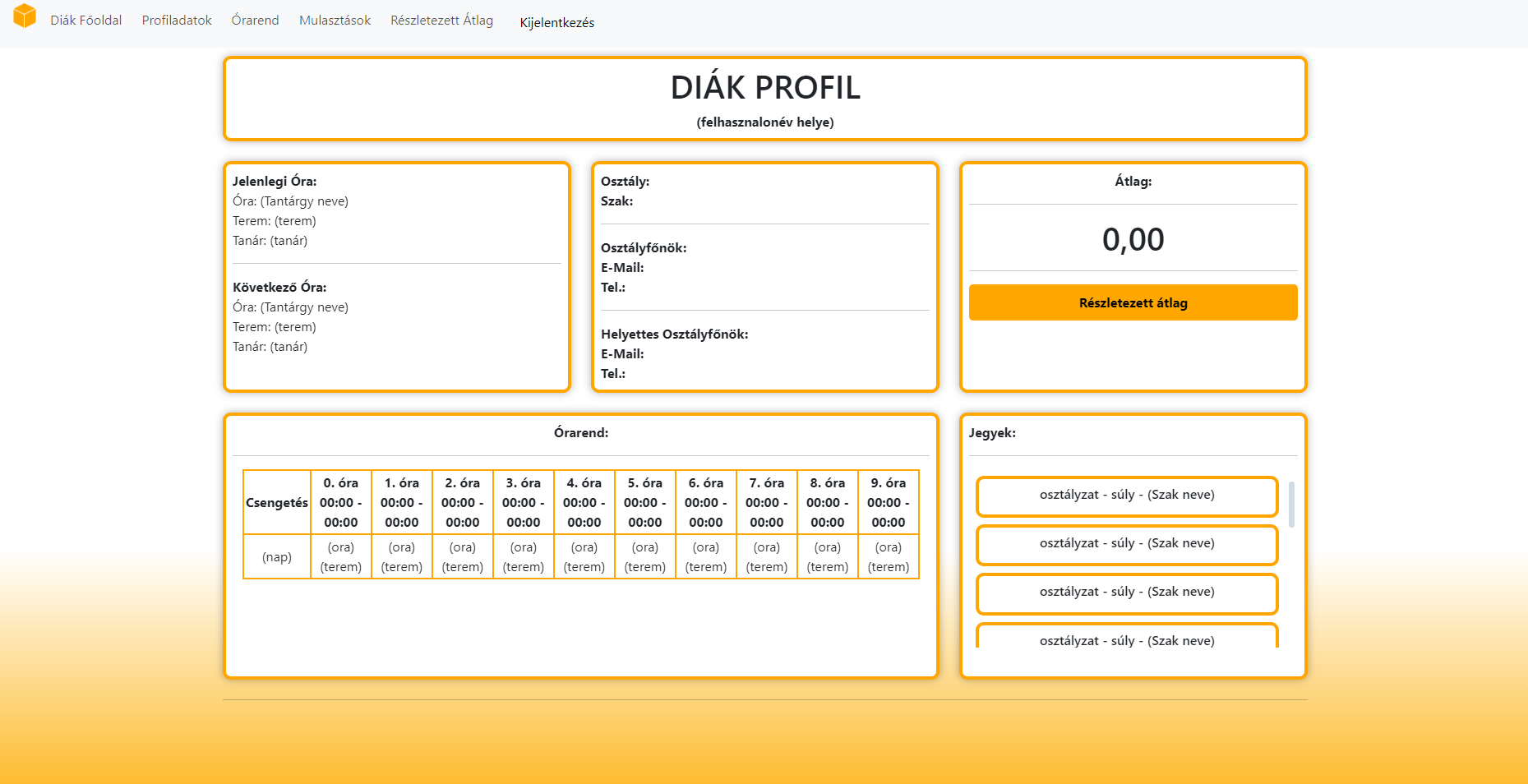
## Tanár:

Ha tanár ként regisztrálták, bejelentkezés után a következő felületet kell látnia:



## Diák:

Ha diák ként regisztrálták, bejelentkezés után a következő felületet kell látnia:

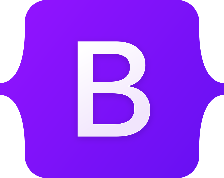


# Használt technikák

## Frontenden használtak:

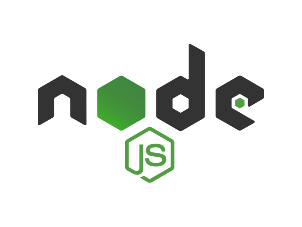
**HTML 5** : A egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki. A weboldalak létrehozásához használtuk.

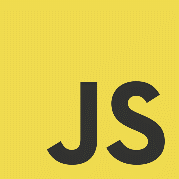
**CSS**: A számítástechnikában egy stílusleíró nyelv, mely a HTML vagy XHTML típusú strukturált dokumentumok megjelenését írja le. a weboldalaink stíluslapjait ebben írtuk meg.

**Bootstrap**: A Bootstrap egy nyílt forráskódú keretrendszer, mely HTML, CSS, JavaScript technológiákat használ. Ezt a technológiát a reszponzivitás megoldására használtuk

**Angular**: Az Angular egy olyan JavaScript alapú rendszer, amellyel dinamikus webes alkalmazásokat készíthetünk ezt a frondend kiszolgálásához használtuk.

## Backenden használtak

**NodeJS**: A Node. js egy szoftverrendszer, melyet skálázható internetes alkalmazások, mégpedig webszerverek készítésére hoztak létre. A backend szerverünk ebben a nyelvben íródott.

**JavaScript:** A JavaScript (röviden JS) egy programozási nyelv, amelyet kifejezetten az internetre fejlesztettek ki. A weboldalon található bizonyos funkciókhoz használtuk.

## Adatbázishoz használtak

**MySQL:** A MySQL, ahogy a neve is mutatja, egy SQL-alapú adatbázis-kezelő rendszer vagy DBMS. Az adatbázisunk első sorban ezt használja.

# Adatbázis

*felelős : Sike István*

# **Adatbázis**

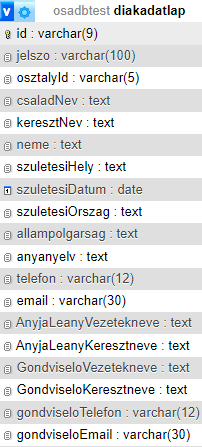
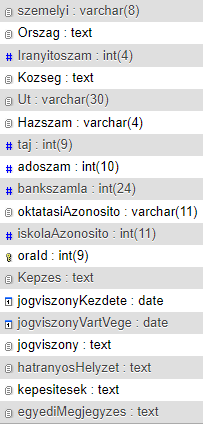
*felelős : Sike István*

Osaadatbazis célja: Elsődlegesen az volt hogy a tanárok és diákok személyes illetve tanitáshoz és tanuláshoz szükséges adataikat eltároljuk. Ezeket az adatokat a későbbiekben feltudjuk használni csoportosításhoz például milyen szakos egy tanár és neki órarendet tárolni.

Kódolás: utf8\_hungarian\_ci

diakadatlap

Ebben a táblában tároljuk el a diák személyes illetve az általa használt legtöbb adatot. Ezek mellé a tábla tartalmazza az iskolához szükséges egyéb adatokat.

id - varchar(9) - Ez az elsődleges kulcs és a diák saját azonosítója

jelszo - varchar(100) - Diák által megadott bejelentkezéshez használandó jelszó

osztalyId - varchar(5) - A diák osztály azonosítója

csaladNev - text - A diák családneve

keresztNev - text - A diák keresztneve

neme - text - A diák neme

szuletesiHely - text - A diák születési helye

szuletesiDatum - date - A diák születésének dátuma

szuletesiOrszag - text - A diák születési országa

allampolgarsag - text - A diák milyen állampolgárú

anyanyelv - text - A diák által használt elsődleges nyelv

telefon - varchar(12) - A diák telefonszáma

email - varchar(30) - A diák email címe

AnyjaleanyVezetekneve - text - A diák anyjának leánykori vezetékneve

AnyjaleanyKeresztneve - text - A diák anyjának leánykori keresztneve

GondviseloVezetekneve - text - A diák gondviselőjének vezetekneve

GondviseloKeresztneve - text - A diák gondviselőjének keresztneve

gondviseloTelefon - varchar(12) - A diák gondviselőjének telefonszáma

gondviseloEmail - varchar(30) - A diák gondviselőjének email címe

szemelyi - varchar(8) - A diák személyi igazolvány száma

Orszag - text - A diák jelenleg melyik országba lakik

Iranyitoszam - int(4) - A diák városának irányítószáma

Kozseg - text - Annak a községnek a neve amelyben a diák lakik

Út - varchar(30) - Az út/utca neve amelyben a diák lakik

Hazszam - varchar(4) - A diák házszáma

taj - int(9) - A diák tajszáma

adoszam - int(10) - A diák adó azonosító száma

bankszamla - int(24) - A diák vagy gondviselő bankszámla száma

oktatasiAzonosito - varchar(11) - Egy egyedi azonosító a diáknak

iskolaAzonosito - int(11) - Az iskola egyéni azonosítója

Kepzes - text - A diák milyen képzésen vesz részt

jogviszonyKezdete - date - A diák mikor kezdete meg tanulmányait

jogviszonyVartVege - date - A diák meglevő tanulmányainak leghamarabb várt vége

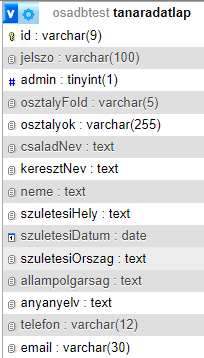
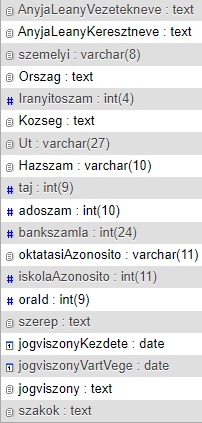
jogviszony - text - Milyen a jogviszonya az iskolában

hatranyosHelyzet - text - Igazolás valamilyen helyzetről

kepesitesek - text - A diáknak van-e valamilyen plusz képesítése

egyediMegjegyzes - text - A diák szorul-e bármilyen speciális igénybe

Tanaradatlap

Ebben a táblában mentjük le a tanár összes olyan adatát amely a tanításhoz és a munkájához elengedhetetlen lenne.

id - varchar(9) - Ez egy elsődleges kulcs és a tanár saját azonosítója

jelszo - varchar(100) - Tanár által megadott bejelentkezéshez használandó jelszó

admin - tinyint(1) - A tanárnak van-e admin joga vagy nem

osztalyFoId - varchar(5) - A tanárosztályfőnök azonosítója

osztalyok - varchar(255) - A tanár milyen osztályokat tanít

csaladNev - text - A tanár családneve

keresztNev - text - A tanár keresztneve

neme - text - A tanár neme

szuletesiHely - text - A tanár születési helye

szuletesiDatum - date - A tanár születésének dátuma

szuletesiOrszag - text - A tanár születési országa

allampolgarsag - text - A tanár állampolgársága

anyanyelv - text - A tanár által használt elsődleges nyelv

telefon - varchar(12) - A tanár telefonszáma

email - varchar(30) - A tanár email címe

AnyjaleanyVezetekneve - text - A tanár anyjának leánykori vezetékneve

AnyjaleanyKeresztneve - text - A tanár anyjának leánykori keresztneve

szemelyi - varchar(8) - A tanár személyi igazolvány száma

Orszag - text - A tanár jelenleg melyik országba lakik

Iranyitoszam - int(4) - A tanár városának irányítószáma

Kozseg - text - Annak a községnek a neve amelyben a tanár lakik

Út - varchar(30) - Az út/utca neve amelyben a tanár lakik

Hazszam - varchar(4) - A tanár házszáma

taj - int(9) - A tanár tajszáma

adoszam - int(10) - A tanár adó azonosító száma

bankszamla - int(24) - A tanár vagy gondviselő bankszámla száma

oktatasiAzonosito - varchar(11) - Egy egyedi azonosító a tanárnak

iskolaAzonosito - int(11) - Az iskola egyéni azonosítója

szerep - text - A tanár milyen szerepe van az intézményben

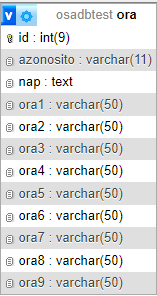
jogviszonyKezdete - date - A tanár mikortól kezdete meg munkáját

jogviszonyVartVege - date - A tanár meglévő munkájának leghamarabb várt vége

jogviszony - text - Milyen a jogviszonya az iskolában

szakok - text - A tanár milyen szakot taníthat

Ora

A diáknak és tanárnak a heti órarendjét táróljuk le ebben a táblában

id - int(9) - Ora egyedi azonosítója

azonosito - varchar(11) - Vagy a tanár azonosítója vagy a diák osztályazonosítója

nap - text - A hét melyik napja

ora1 - text - Az első óra neve

ora2 - text - A második óra neve

ora3 - text - A harmadik óra neve

ora4 - text - A negyedik óra neve

ora5 - text - Az ötödik óra neve

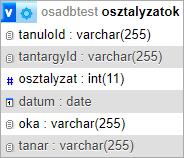
ora6 - text - A hatodik óra neve

ora7 - text - A hetedik óra neve

ora8 - text - A nyolcadik óra neve

ora9 - text - A kilencedik óra neve

Osztalyzatok

Itt a tanuló által kapott érdemjegyeket mentjük el. Azok alapján hogy milyen tantárgyból milyen jegyet, mikor, miért és milyen tanártól kapott.

tanuloId - varchar(255) - A tanuló egyedi azonosítója

tantargyId - varchar(255) - A tantárgy neve

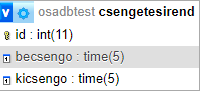
osztalyzat - int(11) - 1-től 5-ig milyen jegyet kapott a tanuló

datum - date - Mikor kapta a tanuló az osztályzatot

oka - varchar(255) - Milyen okból kapta a tanuló az osztályzatot

tanar - varchar(255) - Melyik tanártól kapta a tanuló az osztályzatot

csengetesirend

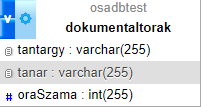
Az órák kezdetett és végét jelző csengetési időpontok vannak a táblázatba lementve.

id - int(11) - Az adott órát jelzi

becsengo - time(5) - Az adott órára való becsengő időpontját jelzi

kicsengo - time(5) - Az adott órára való kicsengő időpontját jelzi

dokumentaltorak

A megtartott óra tantárgyát, tanárát és óraszámát lehet ide lementi

tantargy - varchar(255) - A megtartott óra neve

tanar - varchar(255) - A órát tartó tanár neve

oraSzama - int(255) - Az évben eddig megtartott tantárgy összes óraszáma

hianyzasok

Ebben a táblában a hiányzások lehet vezetni. Illetve az Órákat igazolni

id - int(11) - A diák azonosítója

datuma - date - A hiányzások dátuma

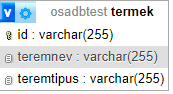
orakSzama - int(11) - Hiányzott órák száma

igazoltOrak - int(10) - Leigazolt órák száma

azonosito - varchar(25) - A hiányzás azonosítója

igazolva - varchar(20) - Az adott hiányzás igazolt állapota

termek

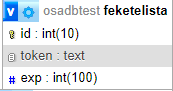
A termek specifikus felépítését például informatikaterem vagy kémiaterem itt tároljuk le

id - varchar(255) - A terem azonosítója

teremnev - varchar(255) - A terem neve

teremtipus - varchar(255) - A terem típusa

feketelista

Ebben a táblában tároljuk le tokenneket.

id - int(10) - token azonosítója

token - text - token neve

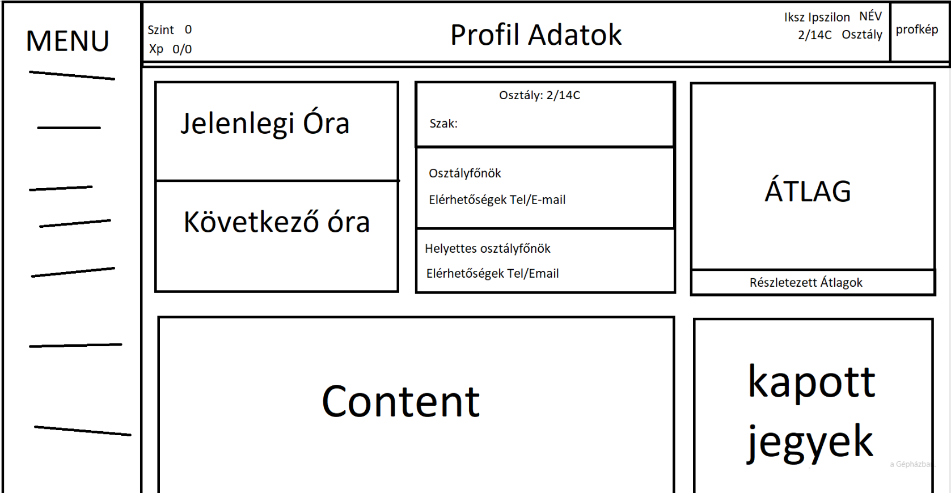
exp - int(100) -

# Frontend szerver

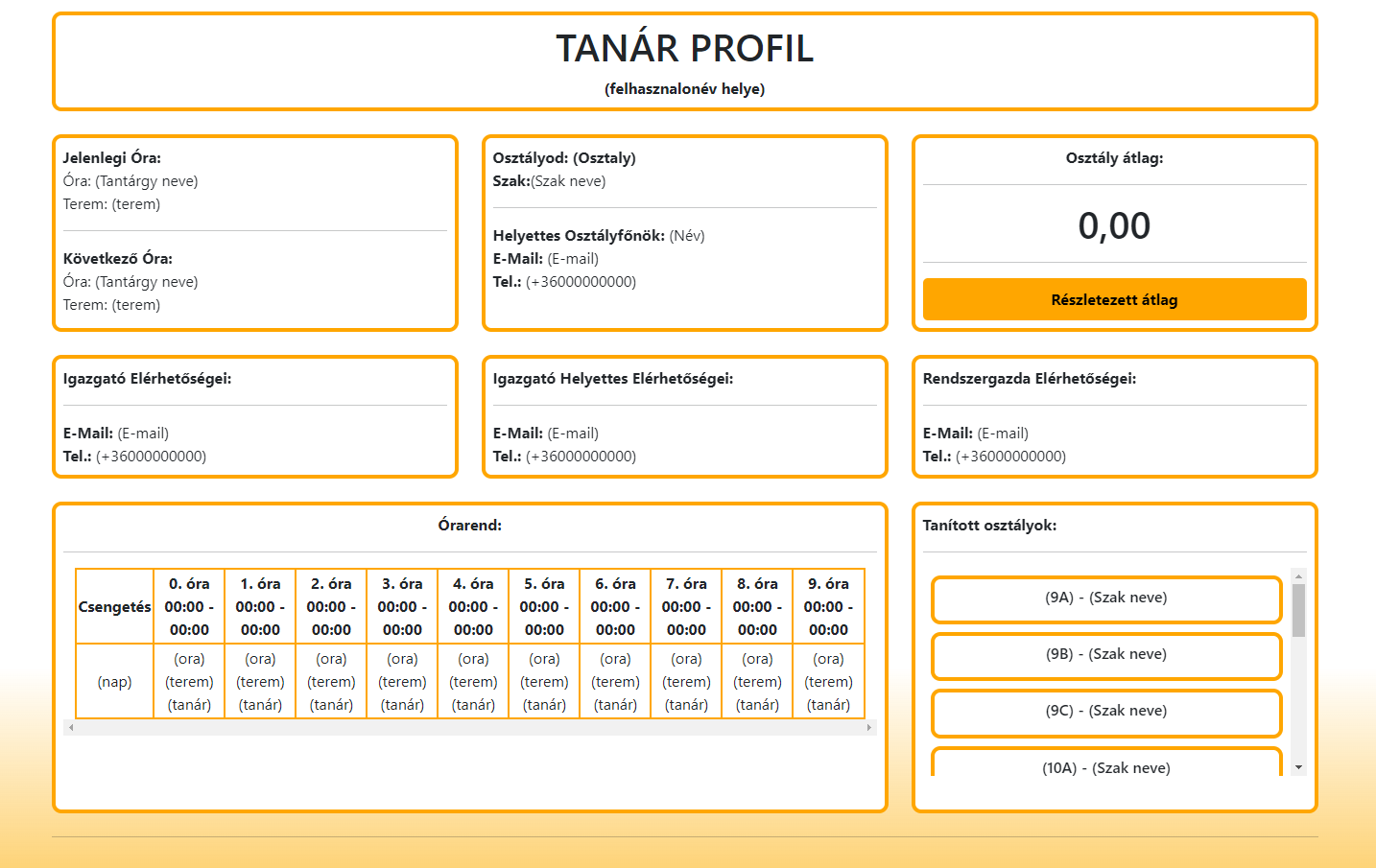
*felelős: Szabadka Konrád*

## Felhasználói felület

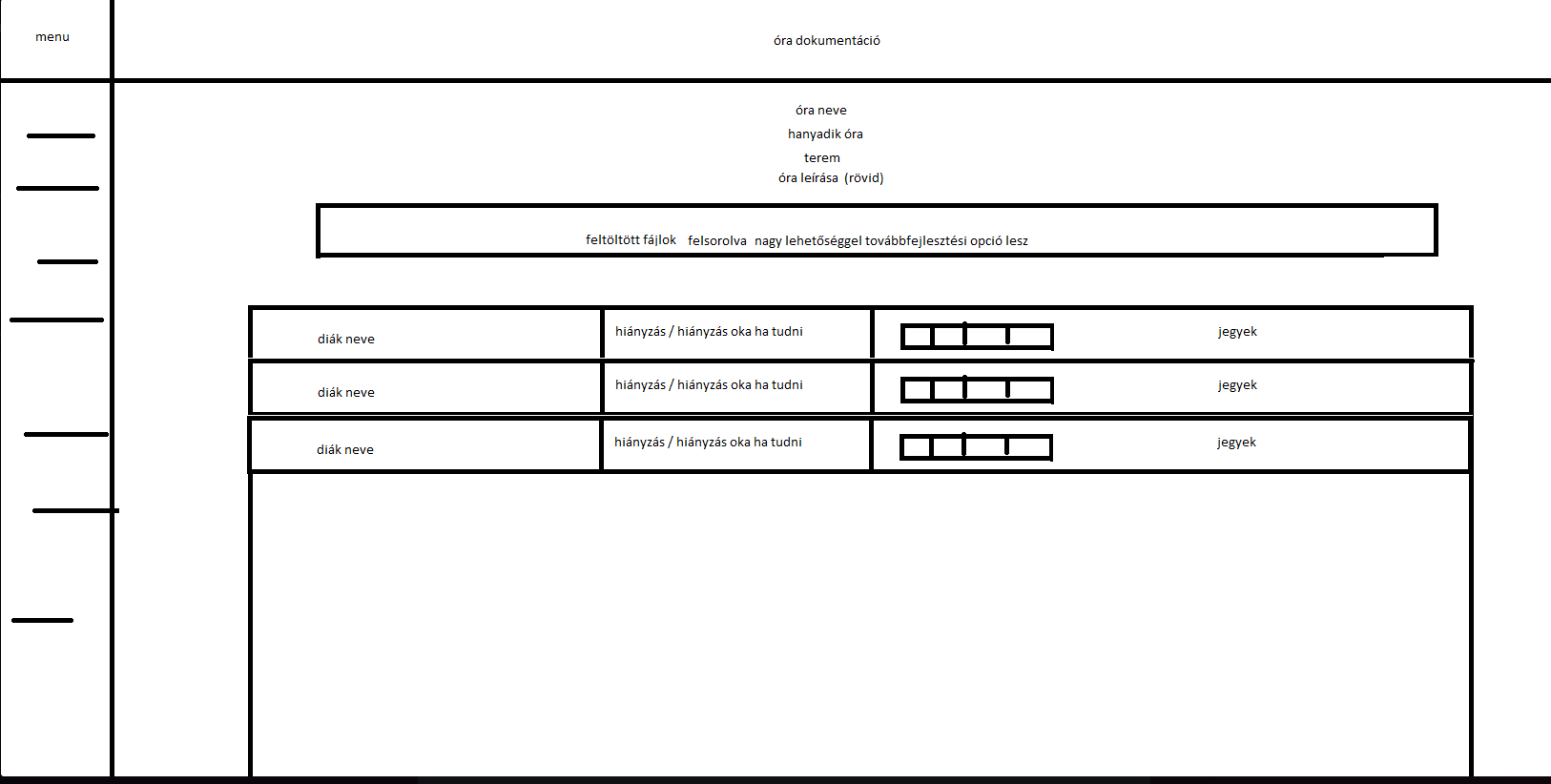
A felhasználói felület során próbáltuk a legjobbat kihozni a hatalmas információ mennyiség minél egyszerűbb és könnyen navigálhatóvá tenni ezeket jelenleg (2023.11.28) csak vázlatrajzokkal ábrázolnánk. A végleges oldaldizájn a fejlesztés közben folyamatos módosításokon ment keresztül, de hű marat az eredeti vázlatok lényeges és fő részeihez. A profilképeket teljesen eltüntettük és a legfelső, „Profil Adatok” sávban található osztály megjelenítését utólag feleslegesnek gondoltuk, mivel az már megjelenítésre kerül a középső blokk legfelső sorában, a szak megnevezése felett.

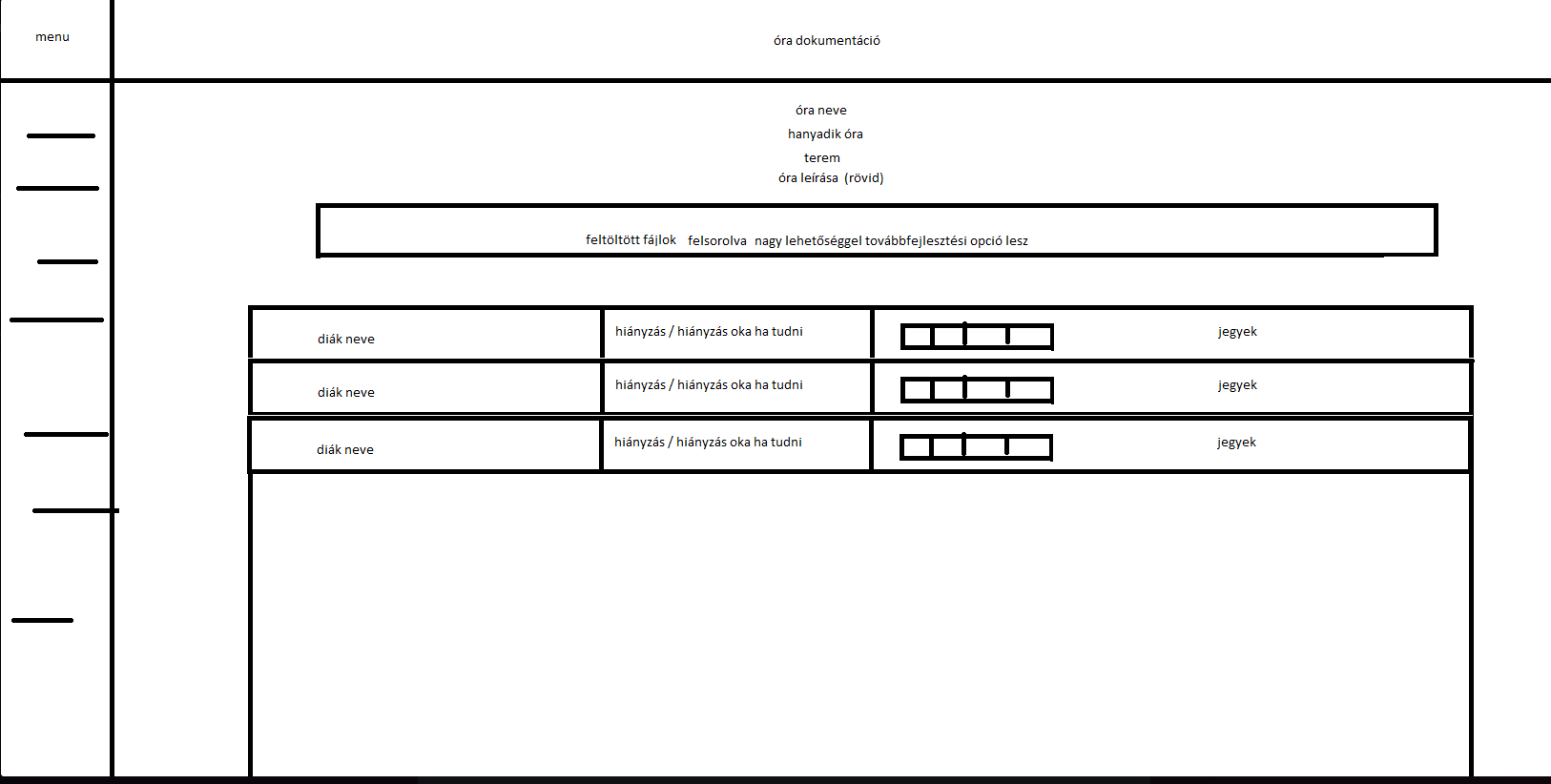
Főoldal: (vázlat)

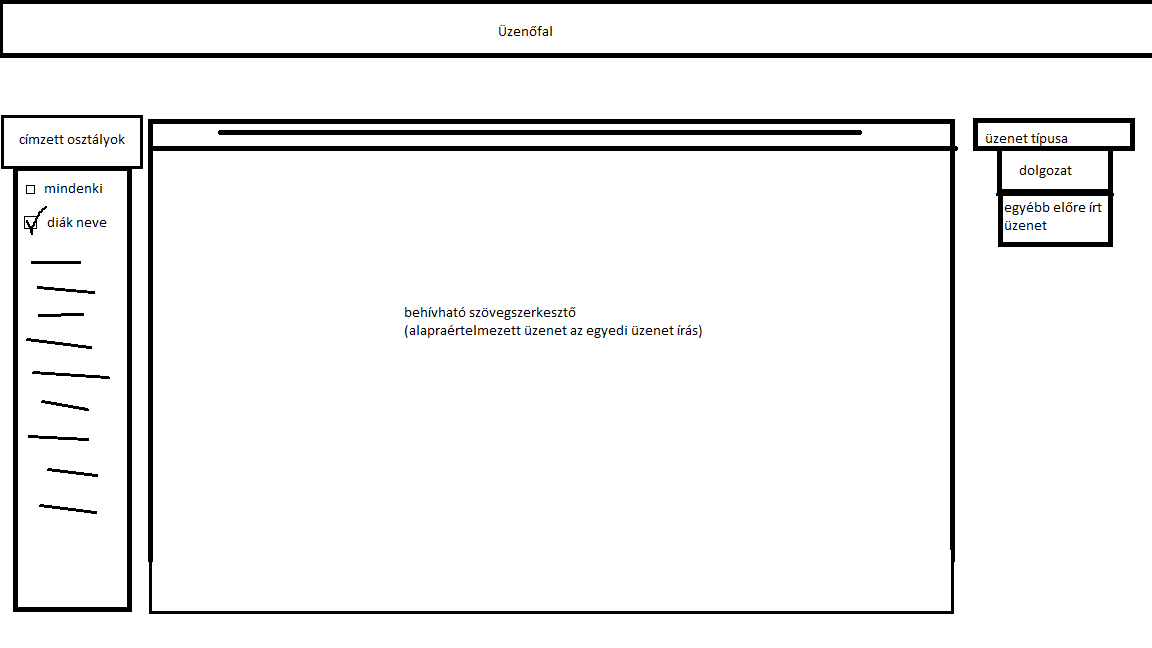
(jelenlegi verzió)



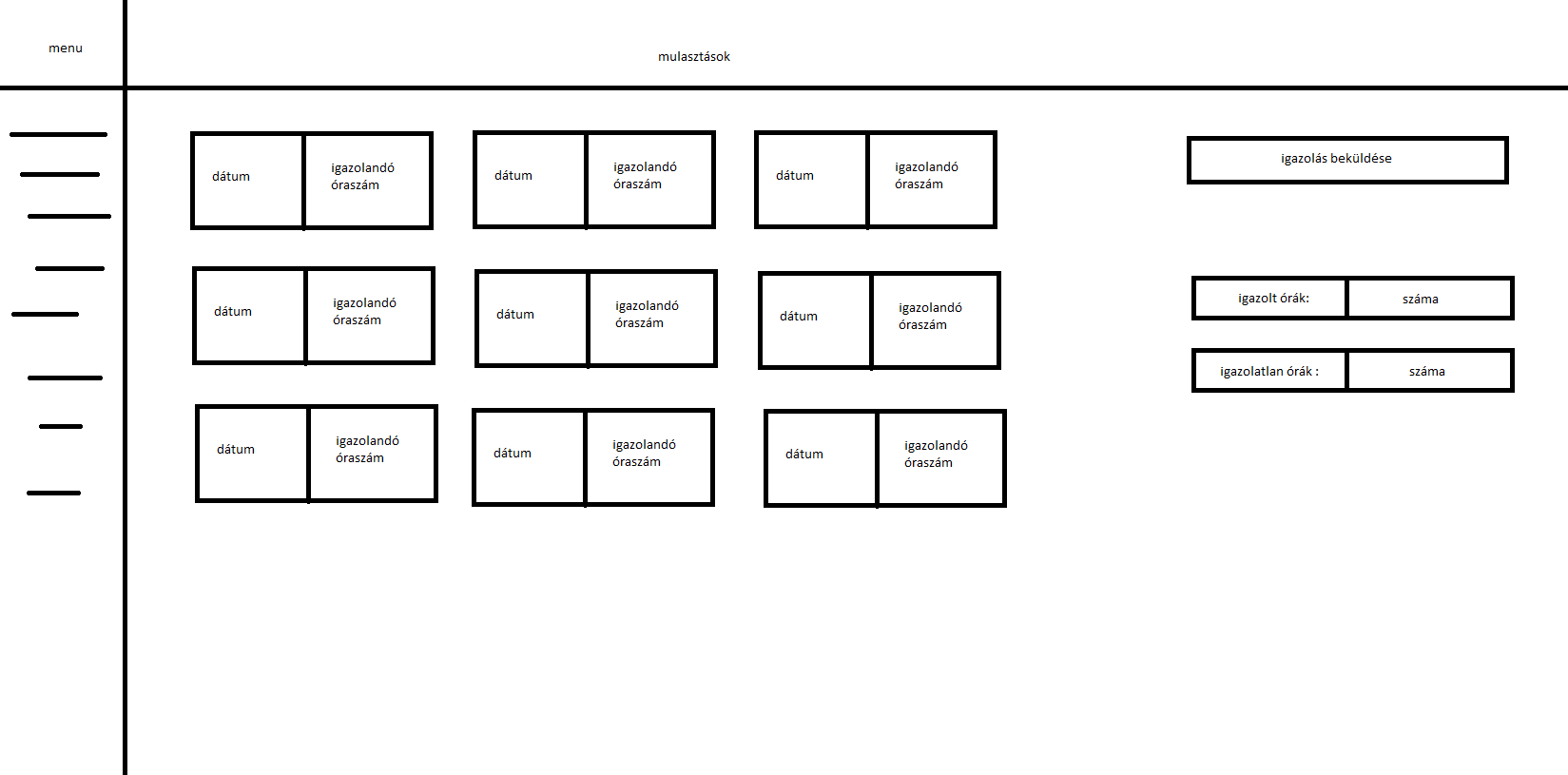
Óra dokumentációs felület



üzenőfal:

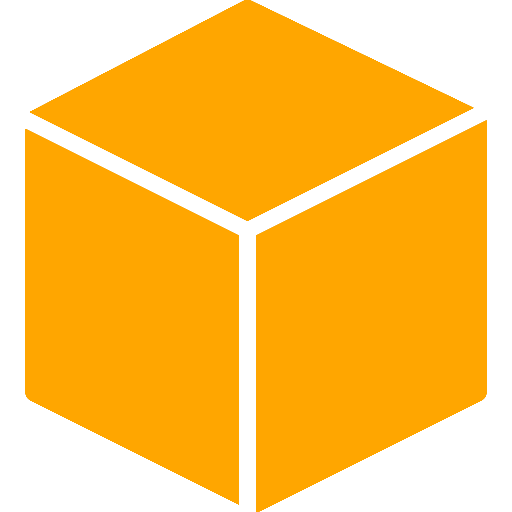


mulasztások:

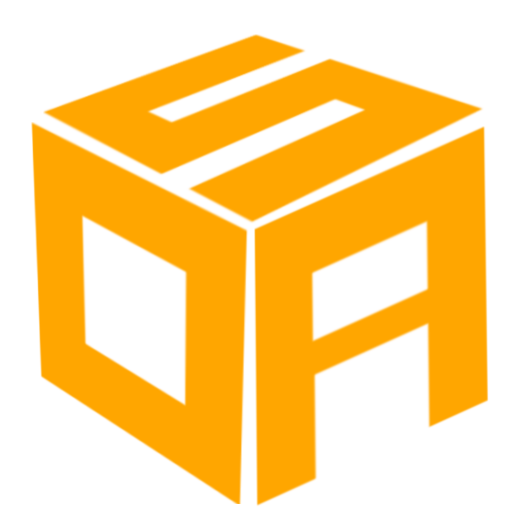


## Logó

Az OSA logójának az alapja:

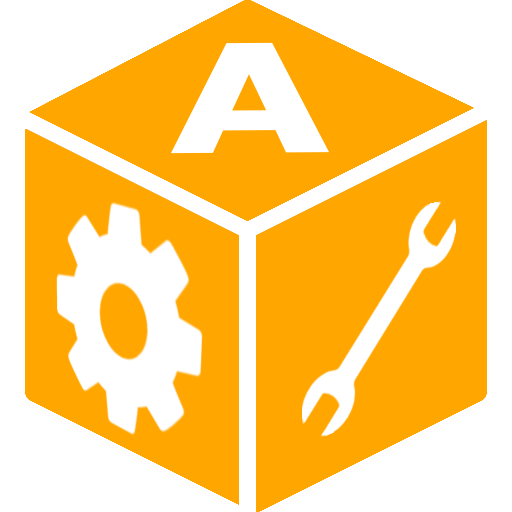


Az OSA logó:



Egy O, S és A betű

Az admin OSA logó:



Az OSA Admin logó, az osa logó alapja egy fogaskerékkel, villáskulcsal és az admin szó „A” betűjével

## Színek

Az oldal fő színpalettájának a következő színek kerültek kiválasztásra:

|  |  |
| --- | --- |
| **Szín** | **Világos** |
| **#FD9F2E** | **#FFA600** |
| **#D13117** | **#E63C1E** |
| **#064065** | **#095585** |
| **#00BDFF** | **#7CDDFF** |
| **#00E0FF** | **#70EEFF** |

A „Világos” színek abban az esetben kerülnek használatra, ha mondjuk egy gombra kattint a felhasználó vagy más interakciókat hajt végre (Pl.: :hover)

li.warning{

    background-color: #D13117;

}

li.warning:hover{

    background-color: #E63C1E;

*}*

*( Használt szín még a fekete és a fehér [#000000], [#FFFFFF] )*

Az oldal alapszíne azért lett a narancssárga, mert ez a szín komfortos érzetet kelt és segíti a neurális funkciókat (mozgás, érzékelés, észlelés és megismerés).

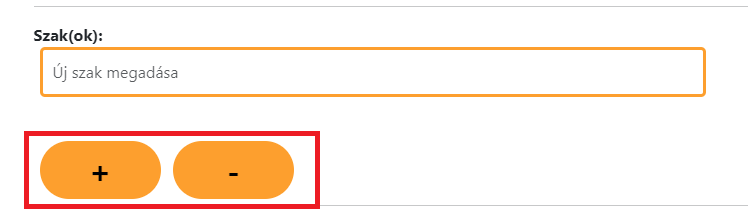
A kék pedig alkalmas a magas szintű tanulás és gondolkodás elősegítésére, de ezt csak egy másodlagos szín ként lett alkalmazva. Ha túl sok lenne a kék az egy hideg hatást keltene amely ellenezné a narancssárga szín használatával szánt, tanulásra és figyelemre ösztönző érzés megteremtését az oldal használata közben.

A piros színt lényeges részek kiemelésére és figyelemfelhívásra használjuk.

## Script

**A script.js file tartalmazza:**

A **Tanár regisztrálása** felületen lévő + és – gombok kódját, amely a tanár szakjaihoz ad hozzá és távolít el text beviteli mezőt.



(részlet a script.js file-ból)

// plusz gomb funkciója

// (hozzáad egy [új szak megadása] beviteli mezőt)

function plusButton(){

    const inputs = szakContainer.querySelectorAll('input');

    if (inputs.length < 10) {

        const ujSzak = document.createElement('input');

        // [ujSzak] input mező paraméterei:

        ujSzak.type= "text";

        ujSzak.classList = "form-control ex-long";

        ujSzak.placeholder = "Új szak megadása";

        ujSzak.title = "Új szak megadása";

        ujSzak.id = "szak-text";

        ujSzak.required = true;

        ujSzak.maxLength = "128";

        ujSzak.style = "width: 55%; padding: 10px; margin-bottom: 20px; margin-left: 0.5%; margin-right: 0.5%; border: 3px solid #ffa600; border-radius: 5px;"

        szakContainer.appendChild(ujSzak);

        console.log("PLUS");

    }

}

// minusz gomb funkciója

// (elvesz egy [új szak megadása] beviteli mezőt )

function minusButton(){

    const inputs = szakContainer.querySelectorAll('input');

    if (inputs.length > 1) {

        szakContainer.removeChild(inputs[inputs.length - 1]);

    }

    console.log("MINUS");

}

A script file fő része a **plusButton()-** és a **minusButton()-function.**

A plusButton funkcióban megvizsgáljuk, hogy a beviteli mezők tárolására szolgált div tartalmaz 10-től kevesebb mezőt. Abban az esetben ha a vizsgálat értéke kisebb mint 10 a kódban létrejön egy új elem „ujszak” néven ami megkapja a szükséges paramétereket (type, classList, placeholder, title, id, require, maxLength, style). A paraméterek megadása után a szakContainer div-hez hozzáadódik a bevitelimező és a konzolra kiíródik a „PLUS” felirat sikeres létrehozás esetén. Abban az esetben ha a beviteli mezők száma eléri a 10db-ot akkor nem jön létre új beviteli mező

A minusButton funkcióban megvizsgáljuk, hogy a beviteli mezők darabszáma nagyobb mint 1. Ha a beviteli mezők száma meghaladja az egyet, akkor a beviteli mezők tárolására szolgált div-ből eltávolítódig egy mező és a konzolra kiíródik a „MINUS” felirat, igazolva a gomb megfelelő működését. Abban az esetben, ha a beviteli mezők száma eléri az egyet, akkor több mező nem törölhető.

# CSS

A diák és tanár komponenseken belül megtalálható „.row” class-nál dizájn szempontból lényeges hogy tartalmazza a „margin-bottom: 24px” tulajdonságot, hogy a blokkok a megfelelő távolságra legyenek egymástól minden irányban!

/\* a row nak alsó margót állítja be. FONTOS a 24px! \*/

.row{

    margin-bottom: 24px;

}

A regisztrációs felületek (regist-t, regist-d komponensek) beviteli mezőinek különböző hosszúságaiért felelősek a következő paraméterek:

.login-form input[type="text"], .login-form input[type="date"] {

    width: 25%;

    padding: 10px;

    margin-bottom: 20px;

    margin-left: 0.5%;

    margin-right: 0.5%;

    border: 3px solid #ffa600;

    border-radius: 5px;

}

/\* TEXT/DATE INPUT SZÉLESSÉGEK \*/

.login-form input[type="text"].short {

    width: 10%;

}

.login-form input[type="date"].date-short{

    width: 12%;

}

.login-form input[type="text"].mid {

    width: 20%;

}

.login-form input[type="text"].long {

    width: 30%;

}

.login-form input[type="text"].ex-long {

    width: 55%;

}

Első sorban a felhasznált beviteli mezők típusainak (text, date) alap tulajdonságait határozzuk meg: (A következő sorokban a lényegesebb tulajdonságok kerülnek felsorolásra a példa kedvéért.) Egy alap 25%-os szélesség beállítása, egy 20px-es alsó bélés és egy-egy 0,5%-os baloldali és jobboldali bélés, hogy az egy sorban, egymás után következő beviteli mezők megfelelő távolságban legyenek egymástól és egy 3px vastag keret.

Ezt követően a „TEXT/DATE INPUT SZÉLESSÉGEK” cím alatt megtalálhatóak a beviteli mező típusok és a hozzájuk rendelt osztályok amelyek segítségével végezhetőek el a beviteli mezők szélességének beállításai.

Elsőként a rövid beviteli mezők beállításai szerepelnek:

* Rövid, szöveges beviteli mezőhöz [ 10% ] short (short -> rövid) osztály haszálata.
* Rövid dátum beviteli mezőhöz [ 12% ] date-short (date-short –> rövid dátum) osztály használata ( A dátum beviteli mezőhöz csak egy, 12%-os szélesség lett létrehozva, mivel az megfelelő méretnek bizonyult a jelenlegi kontextusban ).

Közepes méretű beviteli mező:

* Közepes, szöveges beviteli mezőhöz [ 20% ] mid (middle -> középső (közepes méretű)) osztály használata.

Végül a nagy méretű beviteli mezők:

* Hosszú szöveges beviteli mező esetén [ 30% ] long (long - hosszú) osztáy használata.
* Extre hosszú szöveges beviteli mező esetén [ 55% ] ex-long (extra long – extra hosszú) osztály használata.

(ezek az arányok nem az egész oldal szélességével arányosak, hanem a „container” osztállyal ellátott div szélességéhez viszonyulnak.)

# Backend server

*felelős: Hadari Dávid*

## Backend ismertetése

Szerver fálj: /src/app.js

Dependency:

* bcrypt: bcrypt egy olyan könyvtár amely célja hogy egy karakterláncot hasheljünk. Mi a jelszavainkat hasheljük ennek használatával
* body-parser: body-parser egy olyan middleware amely lekezeli a kérések tartalmát
* express: expres.js egy olyan keretrendszer aminek segítségével web aplikációkat lehet készíteni. Express.js segítségével hozzuk létre a backend szervert.,
* jsonwebtoken: JWT egy biztonságos módszert nyújt kliensek közötti komunikációra. JWT-t felhasználók azonosítására és hitelesítésére használjuk,
* mariadb: ^3.2.3,
* sequelize: sequelize.js lehetőséget ad arra hogy adatbázissal kapcsolatot létesíthessünk az adatbázis és backend szerver között.
* cors: 2.8.5

Dev-dependency:

* nodemon: ^3.0.2

Dependency-k letöltéséhez írjuk a terminálba : *npm i*

Szerver inditásához : *npm run start*

nodemon futtatásához: *npm run dev*

## Backend Fájljai

Az src nevű mappa tartalmazza az app.js és db.js nevű fájlokat illetve a controllers,middleware,routes és dbModels nevű mappákat.

### app.js

Az app.js fájl tartalmazza a backend szerver futásához szükséges forráskódot.

//könyvtárak

const express = require("express");

const bodyParser = require("body-parser");

const sequelize = require("./db");

const jwt = require("jsonwebtoken");

const nodeCron = require("node-cron");

const cors = require("cors");

const { Op } = require("sequelize");

const bcrypt = require("bcrypt");

////////////////////////////////////////////////////////////////

//modellek

const TeremModel = require("./dbModels/termek.model");

const TanaradatlapModel = require("./dbModels/tanaradatlap.model");

const TantargyModel = require("./dbModels/tantargyak.model")

const OsztalyzatModel = require("./dbModels/osztalyzat.model");

const DiakadatlapModel = require("./dbModels/diakadatlap.model");

const FeketeListaModel = require("./dbModels/feketelista.model");

const OraModel = require("./dbModels/ora.model");

const HianyzasokModel = require("./dbModels/hianyzasok.model");

const dokumentaltOrakModel =require("./dbModels/dokumentaltOrak.model");

//////////////////////////////////////

//route-k

const logRegRouter = require("./routes/loginRegisterRoute");

const adminRouter = require("./routes/adminRoute");

const diakRouter = require("./routes/diakRoute");

const tanarRouter = require("./routes/tanarRoute");

//////////////////////////////////////

A fájl elején találhatjuk a használt könyvtárak.

Utána következik az adatbázishoz szükséges adatmodellekre történő hivatkozások az elérési útvonalukkal.

Következő részleg pedig a route-k eléréseiért felelős itt szintén az elérési útvonalakat használjuk.

//////////////////////////////adatbázis csatlakozás////////////////////////

sequelize.authenticate().then(() => {

    sequelize.modelManager.addModel(TeremModel);

    sequelize.modelManager.addModel(TanaradatlapModel);

    sequelize.modelManager.addModel(TantargyModel);

    sequelize.modelManager.addModel(OsztalyzatModel);

    sequelize.modelManager.addModel(DiakadatlapModel);

    sequelize.modelManager.addModel(FeketeListaModel);

    sequelize.modelManager.addModel(OraModel);

    sequelize.modelManager.addModel(HianyzasokModel);

    sequelize.modelManager.addModel(dokumentaltOrakModel);

    try{

    sequelize.sync();

    console.log("kapcsolat sikeresen létesült");

    }catch(err){

        return console.log(err);

    }

}).catch((error) => {

    console.log("Az adatbázissal nem sikerült a kapcsolat");

    console.log(error);

})

//////////////////////////////////////////////////////////////////////////////

A fájl következő fontosabb részlege az adatbázishoz a csatlakozás itt ellenőrizzük hogy a kapcsolat létrehozható-e utána a modelManagert használva hozzáadjuk a meghívott adat modelleinket majd szinkronizáljuk. Ha a kapcsolat nem létesül egy hibát adunk vissza és a console.log segítségével a konzolon tudatjuk hogy nem létesült a kapcsolat ebben az esetben a szerver nem indul el.

app.use("/", logRegRouter);

app.use("/", adminRouter);

app.use("/",diakRouter);

app.use("/", tanarRouter);

app.listen(PORT, () => {

    console.log(`A szerver elindult a http://localhost:${PORT} -es porton!`);

});

Majd végül megmondjuk a szervernek hogy használja a megadott routekat és végül ha az adatbázis kapcsolat létrejön akkor a szerver elindul.

### db.js

db.js nevű fájl tartalmazza az adatbázis kapcsolathoz szükséges konfigurációt

//könyvtárak

const {Sequelize} = require("sequelize");

///////////////////////////

const sequelize = new Sequelize("osadbtest", "root", "", {

    host:"localhost",

    dialect:"mariadb"

});

module.exports = sequelize;

A fájl elején hivatkozunk a Sequelize könyvtárunkra majd az adatbázis konfigurációja következik ahol beálítjuk a host-ot, a dialect-et.

### dbModels mappa

Itt találjuk az adatbázishoz szükséges modellek fájljait

A kód felépítése a következő : Meghívjuk a sequelize DataTypes részét és a db.js fájlunkra majd definiálunk a modellünk amit az adattábla „oszlopainak” leírásával és specifikálásával teszünk .

### Routes mappa

Ebben a mappában találjuk a routekat amiket használunk tartalmazza az admin, a diák, a tanár, és egyébb routek fájlait.

//könyvtárak

const express = require("express");

/////////////////////////////////////////////////////

//middleware

const {tokenHitelesites} = require ("../middleware/hitelesitesMiddle");

const {adminHitelesites} = require("../middleware/hitelesitesMiddle");

////////////////////////////

//Router

const adminRouter = express.Router();

//Controller

const modositasController = require("../controllers/adatmodositasController");

const logRegController = require("../controllers/loginRegisterController");

////////////////////////////

//metódusok

adminRouter.get("/admin/adatmodositas",adminHitelesites, modositasController.getUser);

adminRouter.patch("/admin/adatmodositas",adminHitelesites,modositasController.patchUser);

adminRouter.delete("/admin/adatmodositas",adminHitelesites,modositasController.deleteUser)

adminRouter.put("/admin/regist-d",adminHitelesites, logRegController.registStudent);

adminRouter.put("/admin/regist-t",adminHitelesites, logRegController.registTeacher);

module.exports = adminRouter;

adminRoute.js fájl a példa képen

A route fájlokban hivatkozunk az express-re

Hivatkuzunk a hitelesítésért felelős middleware-re (későbiekben kifejtésre kerül a működése)

utána hivatkozunk a használandó controller fájlokra.

Végül a routereket megírjuk:

router.metódus(„url”,middleware, Controller.function);

### Controllers mappa

A Controllers mappa tartalmazza a szerver Controller fájljait amik felelősek a http kérések feldolgozásáért.

### logRegisterController.js

Ez a controller felelős a felhasználók bejelentkezéséért, kijelentkezéséért és regisztrációjáért

regisztráció

for(let i = 0; i< tanarKötAdatok.length; i++){

    if(!Object.keys(req.body).includes(tanarKötAdatok[i])){

        console.log(tanarKötAdatok[i]);

        res.status(400).json({

            "error": "nem megfelelő adat került elküldésre"

            });

            return;

    }

  };

Object.values(tanarKötAdatok).forEach((element)=>{

    if(element == "" || element === undefined){

        res.status(400).json({

            "error": "Egyik adat hiányzik kérem ellenőrizze"

            });

            return;

    }

});

Itt ellenőrizzük hogy az elküldött kérés tartalmazza-e a kötelező adatokat amiket a tanarKötAdatok vagy diakKötAdatok (felhasználótól függően) nevű listában tároljuk le emellett ellenőrizzük hogy tartalmazza-e az adatokat.

const tanar = TanaradatlapModel.build({

            id:req.body.oktatasiAzonosito,

            osztalyId:"2/14c",

            csaladNev:req.body.csaladNev,

            keresztNev: req.body.keresztNev,

            neme:req.body.neme,

            szuletesiHely: req.body.szuletesiHely,

            szuletesiDatum:req.body.szuletesiDatum,

            szuletesiOrszag:req.body.szuletesiOrszag,

            allampolgarsag:req.body.allampolgarsag,

            anyanyelv:req.body.anyanyelv,

            telefon:req.body.telefon,

            email:req.body.email,

            AnyjaLeanyVezetekneve:req.body.AnyjaLeanyVezetekneve,

            AnyjaLeanyKeresztneve:req.body.AnyjaLeanyKeresztneve,

            Orszag:req.body.Orszag,

            Iranyitoszam:req.body.Iranyitoszam,

            Kozseg:req.body.Kozseg,

            Ut:req.body.Ut,

            Hazszam:req.body.Hazszam,

            taj:req.body.taj,

            adoSzam:req.body.adoSzam,

            bankszámlaszám:req.body.bankszámlaszám,

            oktatasiAzonosito:req.body.oktatasiAzonosito,

            iskolaAzonosito:12345678911,

            oraId:123456789,

            szerep:req.body.szerep,

            jogviszonyKezdete:req.body.jogviszonyKezdete,

            jogviszonyVartVege:req.body.jogviszonyVartVege,

            jogviszony:req.body.jogviszony,

            szakok: req.body.szakok,

            jelszo: await hashData(req.body.jelszo),

            admin: req.body.admin

            });

            console.log(tanar.toJSON());

            tanar.save().then(() => {

                res.status(200).json({

                    "response":"sikeres regisztráció"

                });

            });

Majd kérés adatait feldolgozzuk majd feltöltjük az adatbázisba

bejelentkezés

//ha a felhasználóhoz elküldött jelszó is egyezik generálja le a tokent

        if (await bcrypt.compare(jelszo, existingUser.jelszo) === true) {

            //generate JWToken

        let payload;

            try{

                //ellenőrizzük hogy a felhasználónak van-e admin joga és azalapján állítjuk a token tartalmát

                if(existingUser.admin === undefined){

                    payload = {

                        nev : existingUser.vezeteknev +" "+existingUser.keresztNev,

                        id: existingUser.id,

                        admin:0

                    }

                }else{

                     payload = {

                        nev : existingUser.vezeteknev +" "+existingUser.keresztNev,

                        id: existingUser.id,

                        admin: existingUser.admin

                    }

                }

            //token egy óráig tart

            var token = jwt.sign(payload, jwtSecretKey, {expiresIn:"50m"});

            }catch(error){

                console.log(error);

                return res.status(520).json({

                    response:"váratlan hiba történt"

                });

            }

        }else{

            return res.status(400).json({

                response:"nincs ilyen felhasználó"

            });

        }

Bejelentkezésnél az elküldött kérés adatai alapján lekérjük a felhasználót ellenőrizzük hogy a felhasználó jelszava megegyezik-e a megadott jelszóval és ha minden megfelelő kreálunk egy payload nevű objektumot ez tartalmazza a token adatait és később ezt hozzá is rendeljük a JWTokenhez. Legvégül amikor a választ küldjük a tokent is elküldjük.

kijelentkezés

 const token = req.headers.authorization.split(' ')[1];

    const {exp} = jwt.decode(token);

    if(token === undefined){

        return res.status(400).json({

            status:"error",

            message:"nincs bejelentkezve felhasználó"

        })

    }

const blacktoken =FeketeListaModel.build({id:zsidé, token: token, exp: exp});

zsidé = zsidé+1;

blacktoken.save().

Kijelentkezés a felhasználó JWToken-jét lekérjük. Ha ez nincs meg akkor nincs felhasználónk szóval visszadobunk egy hibát. Ellenkezőleg a kijelentkezés után megmaradó tokent feltöltjuk a feketelistára.

### adatmodositasController.js

Ez a fájl felelős azért hogy a a felhasználó adatait tudjuk módosítani

A getUser() metódussal elküldük egy felhasználó adatait a kérésben küldött azonosító alapján

adatmódosítás

Adatmódosításunk úgy működik hogy az elküldött azonosítóhoz tartozó felhasználó adatait megváltoztatjuk a kérésben megfelelő adatokra és végül feltöltjük.

TanaradatlapModel.findOne({

            where:{

                id: req.body.id

            }

        }).then(async (tanar) => {

            for(const elem of Object.keys(req.body)){

                tanar[elem] = req.body[elem];

            }

                    tanar.save().then(()=>{

                            return res.status(200).json({

                               succes:true,

                                message: "sikeres frissités"

                            });

                        })

                })

            if ( tanar === null){

                DiakadatlapModel.findOne({

                        where:{

                            id: req.body.id

                        }

                    }).then((diak) => {

                        for(const elem of Object.keys(req.body)){

                            diak[elem] = req.body[elem];

                        }

                            diak.save().then(()=>{

                                return res.status(200).json({

                                   succes:true,

                                    message: "sikeres frissités"

                                });

                            })

                    });

            } else{

                return res.status(404).json({

                   succes:false,

                    message:"Nincs ilyen felhasználó"

                })

            }

Felhasználó törlés

Kérésben elküldött azonosítóhoz tartozó felhasználót töröljük az adatbázisból

### mulasztasokController.js

Ez a controller fájl felelős a mulasztások kezeléséért

mulasztás írása

 if(req.body.igazolva == null || req.body.igazolva == undefined){

        igazolasTipus = "igazolatlan";

    }else{

        igazolasTipus = req.body.igazolva;

    }

    HianyzasokModel.findOne({

        where:{

        [Op.or]:[

            {azonosito:req.body.azonosito},

            {datuma: date}

        ]}

    }).then(async (existingHianyzas)=>{

        if(existingHianyzas == null){

           const hianyzas = HianyzasokModel.build({

            datuma:date,

            orakSzama:1,

            azonosito: req.body.azonosito,

            igazolva: igazolasTipus

            });

                hianyzas.save().then(()=>{

                    return res.status(200).json({

                        succes:true,

                        message:"hiányzás rögzítve"

                    })

                }).catch((err)=>{

                    console.log(err)

                    return res.status(520).json({

                        succes:false,

                        message:"adatbázis hiba"

                    })

                });

        }else{

            existingHianyzas.set({

                orakSzama: existingHianyzas.orakSzama + 1})

Mulasztások feljegyzéséhez előszőr is ellenőrizzük hogy az adott dátummal létre van-e hozva mulasztás ha igen akkor az orakSzama nevű adatot növeljük eggyel. Ellenkezőleg hozza létre a mulasztást.

Igazolás

 HianyzasokModel.findOne({

        where:{

            azonosito: req.body.azonosito,

datuma: req.body.datum

        }

    }).then((igazolando)=>{

        if(igazolando == null){

            return res.status(404).json({

                succes:false,

                message:"nincs ilyen hianyzas"

            })

        }

        igazolando.set({

            igazoltOrak: req.body.orakIgazolva,

            igazolva:req.body.igazolas

        });

            igazolando.save()

Diák azonosítója alapján lekérjük a hiányzásait. Utána a visszakapott hiányzás adatait átállítjuk igazoltOrak adatot megnöveljük annyira amennyi órát igazolunk és beállítjuk hogy milyen igazolással lett leigazolva.

Igazolás lekérése

Igazolás lekérése azonosító és dátum alapján működik

### osztalyzatokController.js

Osztályzatok rögzítéséért, lekérdezéséért és átlag számításért felelős controller.

Jegy rögzítése

if((0 >= req.body.osztalyzat) || (req.body.osztalyzat >= 6)){

        return res.status(400).json({

            succes:false,

            message:"nem érvényes érdemjegy"

        })

    }

    const osztalyzat = OsztalyzatModel.build({

        tanuloID:req.body.tanuloId,

        tantargyID:req.body.tantargyId,

        osztalyzat:req.body.osztalyzat,

        datum: date,

        oka:req.body.oka,

        tanar: id

    })

    osztalyzat.save().then(()=>{

        return res.status(200).json({

            succes:true,

            message:"sikeres osztályozás"

        })

    }).catch((err)=>{

Jegy rögzítéséhez felhasználjuk egy tanuló azonosítóját, tantárgy megnevezését, osztályzatot, dátumot, tanár azonosítót és indokot majd feltöltjük ezt az adatbázisba

jegy lekérés

Tantárgy vagy diák azonosító alapján visszaadja a jegyeket

átlag számítás

Tantárgy vagy diák azonosító alapján visszaadja az átlagot ha a tantárgy azonosító nincs akkor a diák összes jegyéből számolja az átlagot

### profilController.js

A profilhoz tartozó metódusokat tartalmazza ez a controller fájl mind a tanár mind a diák részére.

profil adatok lekérése

const date = new Date(Date.now());

var maiOrarend;

const {id} = jwt.decode(req.headers.authorization.split(' ')[1]);

    DiakadatlapModel.findOne({where:{ id: id}}).then(async (diak)=> {

        if(diak == null){

            return res.status(404).json({

               succes:false,

                message:"nincs ilyen felhasználó"

            });

        }else{

            //megvárjuk a szükséges adatok lekérdezését az adatbázisból

            Promise.all([

            tanar = await TanaradatlapModel.findOne({where:{osztalyFoId: diak.osztalyId}}),

            jegyek = await OsztalyzatModel.findAll({where:{tanuloId: id}}),

            orarend = await oraModel.findAll({where:{azonosito:diak.osztalyId}}),

            orak = await CsengetesiRendModel.findAll(),

            ]).then(() => {

        ////////////////////////felhasználó órarendjének lekérése

                    if(date.getDay() == 0 || date.getDay() == 6){

                        maiOrarend = "mai nap nincs tanítási óra";

                    }else{

                        maiOrarend = orarend[date.toLocaleString('hu-HU', {  weekday: 'long' })];

                    }

                    ///////////////jelenlegi és következő óra

                    //console.log(date.getHours() + " : " + date.getMinutes());

                    var oraJ;

                    var oraK;

                    for(let i = 0; i < orak.length; i++){

 let beÓ = parseInt(orak[(i)].becsengo.split(":")[0]); //becsengetés órája

 let beP = parseInt(orak[(i)].becsengo.split(":")[1]);   //becsengetés perce

let kiÓ = parseInt(orak[(i)].kicsengo.split(":")[0]);   //kicsengetés órája

let kiP = parseInt(orak[(i)].kicsengo.split(":")[1]);

    if( (beÓ <= date.getHours()) && (beP <= date.getMinutes())){

                                   oraX = `ora${(i+1)}`;

                                   oraJ = maiOrarend[oraX];

                                   oraX = `ora${(i+2)}`;

                                    oraK = maiOrarend[oraX];

                    }

                    }

                    /////////////////////////////////////////////////////////

                    ////////////////////átlag kiszámítása

                        let atlag = 0;

                        jegyek.forEach(jegy => {

                            atlag += jegy.osztalyzat;

                        });

                        atlag = atlag/jegyek.length;

                    ///////////////////////////////////////

                    return res.status(200).json({

                       succes:true,

                        message:"sikeres adatlekérés",

                        data:{

                            nev: diak.csaladNev + " " + diak.keresztNev,

                            szak: diak.Kepzes,

                            osztaly: diak.osztalyId,

                        osztalyFo: tanar.csaladNev + " " + tanar.keresztNev,

                            osztalyFoTel: tanar.telefon,

                            osztalyFoEmail: tanar.email,

                            jegyek:jegyek, //array

                            atlag: atlag,

                            napiOrarend: maiOrarend, //object

                            aktualisOra:{

                                jelenOra: oraJ,

                                kovOra: oraK

                            }

                        }

                    })

            }).catch((err) => {

                console.log(err);

                return res.status(502).json({

                   succes:false,

                    message:"adatbázis hiba"

                })

            })

A főoldalon lévő profilhoz szükséges adatokat id alapján lekérjük az adatbázisból. Ezek az adatok: név, képzés, osztály azonosító, osztályfőnök telefonszáma és e-mail, legutóbbi jegyek, átlag, mai napi órarend, az aktuális óra és következő óra. Tanár oldalon ezek változnak.

Ennek a folyamatnak a menete a következő

Promise.all() metódussal megvárjuk míg a benne lévő összes promise alapú metódus kész és miután ez végez a parancsok folyamata folytatódik. Promise.all()-ban található metódusok vissza adják a: osztályfőnököt, orarendet, és csengetési rendet, jegyeket

Mai dátumot lekérjük ez alapján pedig megszerezzük hogy jelenleg „hanyadik óra” van és ezzel az információval lekérdezzük melyik óra van jelenleg és milyen óra lesz következőnek.

Tanár oldalon

Változás lekérjük a tanár összes tanított osztályát és végül lekérjük hogy mely diákok tartoznak az adott osztályokba és ezeket visszaadjuk.

Felhasználó összes adata

Ez egy egyszerű adatbázis lekérdezés alapján működik azaz az adatbázisban lévő profil adatokat elküldjük.

Órarend lekérés

const {id} = jwt.decode(req.headers.authorization.split(' ')[1]);

    DiakadatlapModel.findOne({where:{id:id}}).then((diak)=>{

        if(diak == null){

                TanaradatlapModel.findOne({where:{id:id}}).then((tanar)=> {

          oraModel.findAll({where:{azonosito:tanar.id}}).then((orarend) =>{

                    return res.status(200).json({

                        succes:true,

                        message:"sikeres lekérés",

                        data: orarend

                    });

                    }).catch((err)=>{

                        console.log(err);

                        return res.status(502).json({

                            succes:false,

                            message:"adatbázis hiba"

                        })

                    })

                })

        }else{

     oraModel.findAll({where:{azonosito:diak.osztalyId}}).then((orarend) =>{

                return res.status(200).json({

                    succes:true,

                    message:"sikeres lekérés",

                    data: orarend

                });

            }).catch((err)=>{

                console.log(err);

                return res.status(502).json({

                    succes:false,

                    message:"adatbázis hiba"

                })

            })

        }

    }).catch((err)=>{

            console.log(err);

            return res.status(502).json({

                succes:false,

                message:"adatbázis hiba"})))

Órarend lekérése a jwtokenben található id-vel lekeresi hogy az órarendben hol található a felhasználó azonosítója és az ahoz tartozó napi órarendet visszaadjuk.

### oraDokumentalas.js

óra dokumentálás

 if(tanar == null){

            return res.status(401).json({

                succes:false,

                message:"Nem tanítja a megadott osztályt vagy tantárgyat"

            })

        }

        //ora alap dokumentálása

        dokumentaltOrakModel.max("oraSzama", { where: { tantargy: req.body.tantargy}}).then((maxOra)=>{

            if((maxOra== null)){

                max = 1;

            }else{

                console.log(maxOra);

            max = maxOra + 1;

            }

            const dokOra = dokumentaltOrakModel.build({

                osztaly: req.body.osztaly,

                tantargy:req.body.tantargy,

                tanar: tanar.id,

                oraSzama: max,

                datum: date

            })

            dokOra.save().then(()=>{

                oraBool = true;

                console.log("óra sikeresen dokumentálva");

            }).catch((err)=>{

                console.log(err)

                return res.status(500).json({

                    succes:false,

                    message:"szerver hiba"

                });

            });

        }).catch((err)=>{

            console.log(err)

            return res.status(500).json({

                succes:false,

                message:"szerver hiba"

            })

        })

Lekérdezzük az adott tantárgy maximum óraszámát hogy tudjuk mihez hasonlítani a jelenlegi óra óraszámát majd feltöltjük

Opcionálisan meg van a lehetőség hogy a kéréshez hozzá adjanak egy egy listát ami tartalmazza a hiányzókat és a kiosztott jegyeket.

Ezek a metódusok a már bemutatott jegy adás és mulasztás rögzítés metódusokat használják.

## Felhasználó hitelesítés

Hitelesítésre JWT (Json Web Token) használunk ezekben adatokat tudunk biztonságosan továbbítani kliensek között.

diakRouter.get("/diak", tokenHitelesites, profilController.getDiak);

adminRouter.get(

"/admin/adatmodositas",adminHitelesites,modositasController.getUser

);

A route-ban található tokenHitelesites vagy adminHitelesites nevű rész az ami ellenőrzi a felhasználót.

### middleware/hitelesitesMiddle.js

 const token = req.headers.authorization.split(' ')[1];

        //check for token

        if (!token) {

            return res.status(401)

                .json(

                    {

                        success: false,

                        message: "Error!Token was not provided."

                    }

                );

        }

Itt ellenőrizzük hogy van-e JWT küldve.

 if(await FeketeListaModel.findOne({where:{token:token}}) !== null){

           return res.status(401).json({

                error:true,

                messagüzenet:"hozzáférés megtagadva (blacklisted)"

            })

        }

Ellenőrizzűk hogy a token szerepel-e a feketelistában ha igen akkor a tokent nem engedjük tovább.

const decodedToken = jwt.verify(token, jwtSecretKey);

            if(decodedToken){

            next();

            }else{

                return res.status(520).json({

                    error:true,

                    message:"hitelesítés közben hiba történt",

                    type:1 //token nem hitelesíthető});}

Majd a jwt beépített ellenőrző metódusával ellenőrizzük hogy a felhasználó által használt token hiteles-e ha igen tovább engedjük.

const decodedToken = jwt.verify(token, jwtSecretKey);

        const {admin} = jwt.decode(token);

            if(decodedToken && admin !== undefined){

                if(admin == 1 ){

                    next();

                }else{

                    return res.status(401).json({

                        error:true,

                        messagüzenet:"hozzáférés megtagadva nem admin"

                    })

                }

            }else{

                return res.status(520).json({

                    error:true,

                    message:"hitelesítés közben hiba történt",

                });

            }

Admin felhasználó hitelesítése esetén a változás itt található ahol a tokent ellenőrizzük hogy hiteles-e utána pedig a tokenben található-e admin adat és annak az értéke alapján (ha 1 akkor admin ha nem akkor 0) vagy tovább engedjük vagy pedig megtagadjuk a hozzáférését.

### Fekete lista

A fekete lista 3 részre lehet bontani 1 feketelista nevű adat táblából a kijelentkezésből ahol a kijelentkeztetett de még le nem járt tokenek kerülnek feltöltésre és egy feketelista tisztító parancssor.

Az adattábla id, token és exp (expiration date) adatokat tartalmazza.

const token = req.headers.authorization.split(' ')[1];

    const {exp} = jwt.decode(token);

    if(token === undefined){

        return res.status(400).json({

            status:"error",

            message:"nincs bejelentkezve felhasználó"

        })

    }

const blacktoken =FeketeListaModel.build({id:zsidé, token: token, exp: exp});

Kijelentkezésnél feltöltjük a felhasználó tokenjét az adatbázisba

//blacklist ellenörzés

        nodeCron.schedule('\*/55 \* \* \* \*', () => {

            FeketeListaModel.destroy({

                where: {

                    exp: {[Op.lt]:Date.now() /1000}

                }

            })

        },{

            scheduled:true,

            timezone:"Europe/Budapest"

        });

//////////////////////////////////////////////////////////////////////

Az app.js fájlunkban találhatjuk azt a forráskódot ami megadott időközönként töröl minden olyan tokent aminek a lejárati ideje már lejárt.

A sequelize segítségével töröljük azokat a tokeneket ahol az exp vagyis lejárati idő kisebb mint a mostani dátum pontosabban az exp már elmúlt ezt a nodeCron használatával ismételtetetten tudjuk futtatni megadott időközönként.

Erre a megoldásra azért van szükség mivel a JWT-t nem lehet csak úgy törölni. Ha egy illetéktelen felhasználó hozzájuk ennek használatával be tudna jutni a rendszerbe

## Backend végpont táblázat

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /bejelentkezes | POST | id (OM azonosító) és jelszó párossal a felhasználót bejelentkeztetjük a felhasználót | Státuszkód:200  sikeres belépés  Státusz:400  Nincs ilyen felhasználó  Státusz 520  Váratlan hiba történt |
| /kijelentkezes | POST | A felhasználót kijelentkeztetjük és feltöltjük a felhasznált jwt-t egy feketelistába | Státusz: 200  sikeres kijelentkezés  státusz: 400  nincs felhasználó bejelentkezve |

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /admin/regist-d | PUT | Bejövő adatok alapján regisztráljuk a diák felhasználót | Státusz 200  sikeres regisztráció  Státusz: 400  nem megfelelő adat került elküldésre  Státusz: 400  Egyik adat hiányzik kérem ellenőrizze |
| /admin/regist-t | PUT | Bejövő adatok alapján regisztráljuk a diák felhasználót | Státusz 200  sikeres regisztráció  Státusz: 400  nem megfelelő adat került elküldésre  Státusz: 400  Egyik adat hiányzik kérem ellenőrizze |

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /admin/adat-modositas | GET | id alapján megjelenÍtjük egy fehasználó adatait | Státusz 200  lekérés sikeres  felhasználó adatai  Státusz 404  nincs ilyen felhasználó |
| /admin/adat-modositas | PATCH | Változtatásokat feltölti a változtatásokat | Státusz 200  Sikeres frissítés  Státusz 400  problemo |
| /admin/adat-modositas | DELETE | id alapján töröljük a felhasználót | Státusz 200  Sikeres törlés  Státusz 404  Nincs ilyen felhasználó |

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /diak | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó specifikus adatait (profil adatok, mai órarend, átlag) | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| /diak/adatok | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó összes adatát | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| /diak/orarend | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó órarendjét | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| /diak/osztalyzatok | GET | a megadott tantárgy vagy tanuló azonosítóval lekérdezzük a megfelelő osztályzatokat | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| /diak/osztalyzatok/atlagszamitas | GET | a megadott tantárgy vagy tanuló azonosítóval lekérdezzük a megfelelő átlagot  tantárgy azonosító nélkül a tanuló össz átlagát adjuk vissza tantárgy azonosítóval az adott tantárgy átlagát | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /diak/mulasztasok | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó mulasztásait | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /tanar | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó specifikus adatait (profil adatok, mai órarend, átlag) | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| /tanar/adatok | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó összes adatát | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| tanar/osztalyok | GET | A jwt-ben tárolt azonosító alapján  lekérjük a felhasználó tanárhoz tartozó osztályokat | státusz 200  sikeres adatlekérés  státusz 500  adatbázis hiba |
| tanar/osztalyok/diakok | GET | A bejövő json adatok segítségével lekérjük a kíválasztott osztály tanulóit | státusz 200  sikeres adatlekérés  státusz 500  adatbázis hiba |
| tanar/osztalyzatok | GET | a megadott tantárgy vagy tanuló azonosítóval lekérdezzük a megfelelő osztályzatokat | státusz 200  sikeres adatlekérés  Státusz 404  nincs ilyen felhasználó  státusz 502  adatbázis hiba |
| tanar/osztalyzatok | PUT | Bejövő json adatokkal létrehozunk egy értékelést és azt feltöltjük az adatbázisba | státusz 200  sikeres adatlekérés  státusz 502  adatbázis hiba |

|  |  |  |  |
| --- | --- | --- | --- |
| Route | Metódus | Működés | Válaszok |
| /tanar/oradokumentalas | PUT | A bejövő json adatokkal dokumentáljuk a jelenleg tartott órát és az esetleges (opcionális) hiányzásokat vagy értékeléseket | Státusz 200  óra sikeresen dokumentálva  Státusz 400  nem érvényes adat  státusz 502  adatbázis hiba |

# Tesztelés

Tesztelés során a postman nevű programot használtuk Backend részen a tesztelésre. Ezt importálni tudjuk a postmanbe az *osa-back-test.postman\_collection.json* export fájl importálásával.

# Használt tartalom

Mdn web docs oldala: [https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express\_Nodejs/Introduction](%20https:/developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction)

Mdn web docs oldala: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Geegs for geeks oldala: <https://www.geeksforgeeks.org/how-to-implement-jwt-authentication-in-express-js-app/> 2024.03.1. 10:23

NPM oldala: <https://www.npmjs.com/package/node-cron>

Sequelize.js hivatalos oldal: <https://sequelize.org/docs/v6/>

Node.js hivatalos oldal: <https://nodejs.org/docs/latest/api/>

Angular hivatalos oldal: <https://angular.io/docs>

Bootsrap hivatalos oldal: <https://getbootstrap.com/>

HTML Color Codes: <https://htmlcolorcodes.com/>

CSS Tricks: <https://css-tricks.com/resolution-specific-stylesheets/> 2024.04.04. 20:49:46