

Szent István Katolikus Technikum és Gimnázium

**Erdei Gábor,**

**Klubert Bálint,**

**Tán Gergő**

MushroomTour

**SZOFTVER FEJLESZTÉS ÉS - TESZTELÉS VIZSGAREMEK**

Sátoraljaújhely, 2023/2024

**Tartalom**

[Bevezetés 3](#_Toc163327989)

[Miért választottuk ezt a témakört? 3](#_Toc163327990)

[Felhasználói dokumentáció 4](#_Toc163327991)

[Rendszer követelmény 4](#_Toc163327992)

[Az alkalmazás / oldal indítása 4](#_Toc163327993)

[Telepítés menete (Early access) 4](#_Toc163327994)

[Funkciók használatának (Early access) 4](#_Toc163327995)

[Fejlesztői dokumentáció 4](#_Toc163327996)

[Használt csoportmunka eszközök bemutatása 5](#_Toc163327997)

[Fejlesztéshez használt technológiák, szoftverek 5](#_Toc163327998)

[Frontend: 5](#_Toc163327999)

[Backend: 5](#_Toc163328000)

[Adatbázis: 7](#_Toc163328001)

[Frontend 7](#_Toc163328002)

[Backend 8](#_Toc163328003)

[app.js 8](#_Toc163328004)

[adatbazisKapcsolat.js 10](#_Toc163328005)

[idozitettFeketeListaTorles.js 10](#_Toc163328006)

[AuthMiddleware.js 12](#_Toc163328007)

[regisztracioMegkotesek.js 13](#_Toc163328008)

[Modellek 15](#_Toc163328009)

[felhasznalo.model.js 15](#_Toc163328010)

[turaJelentkezes.model.js 17](#_Toc163328011)

[Controllerek 17](#_Toc163328012)

[bejelentkezes.controller.js 18](#_Toc163328013)

[forum.controller.js 18](#_Toc163328014)

[gomba.controller.js 18](#_Toc163328015)

[profil.controller.js 18](#_Toc163328016)

[regisztracio.controller.js 18](#_Toc163328017)

[turak.controller.js 18](#_Toc163328018)

[Pár példa a fenti Controllerekből: 19](#_Toc163328019)

[bejelentkezesPOSTController, token létrehozás: 19](#_Toc163328020)

[regisztracioPUTController, jelszó validálás hitelesítési példaként, adatok feltöltése: 20](#_Toc163328021)

[osszesTurakGETController, kapcsolatok általi lekérdezés, mapping: 21](#_Toc163328022)

[turakPOSTController, életkor ellenőrzése: 21](#_Toc163328023)

[Adatbázis 22](#_Toc163328024)

[zemplen\_gombai 22](#_Toc163328025)

[Tervezet\_turak 25](#_Toc163328026)

[Felhasznalo 25](#_Toc163328027)

[Post\_table 26](#_Toc163328028)

[Jelentkezok\_turara 26](#_Toc163328029)

[fekete\_lista 26](#_Toc163328030)

[Adatbázis Képek 28](#_Toc163328031)

[E-k diagram 30](#_Toc163328032)

[Tesztelés 33](#_Toc163328033)

[Tesztelési terv 33](#_Toc163328034)

[1.Bevezetés 33](#_Toc163328035)

[2. Tesztelés 33](#_Toc163328036)

[2.1 Regisztrációs oldal 33](#_Toc163328037)

[2.2 Belépes oldal 34](#_Toc163328038)

[2.3 Túrajelentkezés 34](#_Toc163328039)

[2.4 Túralétrehozás 35](#_Toc163328040)

[2.5 Gomba enciklopédia 35](#_Toc163328041)

[2.6 Profil oldal 35](#_Toc163328042)

[2.7 Térkép oldal 35](#_Toc163328043)

[2.8 Forum oldal 36](#_Toc163328044)

[Tovább fejlesztési lehetőségek 36](#_Toc163328045)

[Irodalomjegyzék 37](#_Toc163328046)

# Bevezetés

Erdei Gábornak, Klubert Bálintnak és Tán Gergőnek hívnak bennünket.

A sátoraljaújhelyi Szent István Katolikus Technikum és Gimnáziumban Kezdtük középiskolai tanulmányainkat 2018-ban ahol 4 év után érettségi vizsgát tettünk. Ezek után 2 éves Szoftverfejlesztő és-Tesztelő nappali tagozatos szakképzésre jelentkeztünk azért, hogy gyarapítsuk ismereteinket, amit az elmúlt 4 és folyamán szereztünk. Mindhárman beadtuk jelentkezésünket egyetemre, hogy még tovább gyarapítsuk ismereteinket ebben a szakágban

## Miért választottuk ezt a témakört?

# Felhasználói dokumentáció

## Rendszer követelmény

|  |  |  |
| --- | --- | --- |
|  | Ajánlott: | Minimális |
| Processzor (CPU) | Intel(R)\_Core(TM)\_i3-9100F\_CPU\_@\_3.60GHz |  |
| Operációs rendszer (OP) | Windows 10 pro | Windows 10 |
| RAM: | 8 GB RAM | 6 GB RAM |
| Tárhely: | GB | GB |
| Internet: | Széles sávú | Széles sávú |
| Monitor: | 1920 x 1080 |  |
| GPU: | NVIDIA GeForce GTX 1050 |  |

## Az alkalmazás / oldal indítása

A felhasználó az oldalt a számítógépen, mobilkészülékén futó, már előzetesen fel/letelepített böngészőjén érheti el. Ajánlott böngészők:

* Google Chrome
* Opera

Leginkább javasolt böngésző a

## Telepítés menete (Early access)

## Funkciók használatának (Early access)

# Fejlesztői dokumentáció

# Használt csoportmunka eszközök bemutatása

A projektünkhöz a manapság elterjedt, és a vizsgakövetelmények által megszabott eszközöket alkalmaztuk.  
Projektszervezéshez a Trello webalapú projektszervező és menedzselő webalkalmazást vettük igénybe. A felületen az előzetesen, még a tervezési fázisban megbeszélt terveket, a személyesen megtörtént megbeszéléseket, a prezentációkat (ezek a Githubon is megtalálhatóak), illetve egyéb részdokumentációkat tároltuk le.  
A projekt készítése közben létrejött kódokat, forrásokat, magát a szoftvert tehát GitHubon, egy felhőalapú tárolószolgáltatást és verziókezelést biztosító weboldalt alkalmaztuk. Ide töltöttük fel a feladatunk során a kódokat, követtük nyomon egymás munkáit, és segítségével bárhonnan elérhettük és fejleszthettük munkánkat.   
Ha épp nem az intézményen belül voltunk, hanem otthonról dolgoztunk feladatunkon, akkor általánosan Discordon vagy Messengeren folyt a közvetlen, illetve közvetett kommunikáció, esetlegesen Gmail-en történt még információátvitel.

Github link: <https://github.com/KBalint2003/GombaTura>

Képek : ../ Gomba\_app\_\_képek\_96db\_

Egyéb dokumentumok:

* A gombagyűjtés szabályai2023.09.28 8:08docx,
* Fontos Gomba szabályok etika.docx
* Fontos gomba\_szabályok\_etika\_Mérgezés \_Tévhitek.docx

## Fejlesztéshez használt technológiák, szoftverek

### Frontend:

Fejlesztői környezet:

### Backend:

Szerver-, futtatási környezet: Node.Js

A szerver futtatási környezete a Node.Js, amellyel JavaScript nyelven skálázható weboldalakat, webalkalmazásokat hozhatunk létre.

Miért ezt használjuk?

A Node.Js mellett azért döntöttünk, ugyanis ismerős nyelvezete, könnyű tanulhatósága, egyszerű, de átlátható felépítése kifejezetten könnyen és effektíven használhatóvá teszi, nem beszélve az erőforrás kímélésről, ugyanis - bár képes több kérést is kezelni egyidejűleg - ha a callback elküldése után nincs teendője, „alszik”, vagyis nem használja feleslegesen az erőforrásokat.

Ezen előnyei matt a Node.Js-t használjuk a Szerveroldali feladatok ellátására, mint például az Adatbázissal való kapcsolat létrehozására vagy a http kérések kezelésére.

Használt package-ek:  
A Node.Js rengeteg, más felhasználók által létrehozott, karbantartott és fejlesztett package-eket kínál, melyekkel egyrészt mrgkönnyíthetjük munkáknak, másrészt rengeteg új funkciót vagyunk képesek beépíteni projektünkbe. A fejlesztés során a következő package-eket implementáltuk és alkalmaztuk:

* **nodemon**: A nodemonnal a kódban történő változtatásokkor, mentés után automatikusan újraindul a szerver, így elkerülhető a folytonos szerver le- és elindítás.
* **bcyrpt**: A bcrypt egy jelszó hashelő package, amivel a jelszavakat tudtuk titkosítani, és a szükséges funkciókkor azokat a bcrypt beépített function-jeivel ”dekódolni”, ezzel leellenőrizve, hogy pl a bejelentkezést végző felhasználó megfelelő jelszót adott meg.
* **cors**: Mivel a Frontend és a Backend szerver nem ugyanazon a porton fut (mivel ugye ez kivitelezhetetlen esetünkben), így a cors csomag használatával tudjuk elvégezni a biztonsági korlátozások kezelését adattovábbításkor.
* **express**: Az express maga a webszerverünk keretrendszere, amivel a kliensoldali kéréseket és válaszokat tudjuk például kezelni.
* **jsonwebtoken**: json web tokeneket alkalmazunk a biztonságos felhasználói adatok továbbítására, ezekkel végezzük el az autentikációt és a jogosultságokat adott oldalakhoz illetve a felhasználók hitelesítését.
* **mariadb**: Az adatbázis nyelvezetét ezzel s package-el tudjuk meghatározni a sequelize-nak.
* **node-cron**: A node-cron lehetőséget nyújt arra, hogy időzített feladatokat tudjunk beállítani JavaScript kódokhoz. Ezt a feketelista törlésére alkalmazzuk.
* **sequelize**: A sequelize teszi lehetővé az adatbázis kapcsolatot a backenddel és a sequelize funkcióival alkalmazzuk és módosítjuk az adatbázisban tárolt adatokat.
* **uuid**: A uuid segítségével egyedi azonosítókat tudunk generálni, ezzel minden egyes felhasználónak és túrának teljesen egyedi az azonosítója.
* **validate**.**js**: A validate-el könnyedén tudtunk feltételeket, validációs szabályokat meghatározni a regisztrációhoz, és hosszabb kódsorok írása nélkül képes ellenőrizni a benne megadott feltételket, ezzel megkönnyítve az beérkező adatok ellenőrzését, hibakeresését.

Fejlesztői környezet: Visual Studio Code

Fejlesztői környezetek közül a Visual Studio Code-ra esett a választás, mivel elmúlt éveinkben ezt használva már igazán otthonosan mozgunk benne, emellett minden, általunk a Frontendhez illetve, Backendhez használt nyelvet és környezetet támogat, megkönnyítve azok használatát, ráadásul rengeteg hasznos kiegészítője van ezzel is segítve munkánkat.

### Adatbázis:

Adatbázis program és verzió: Xampp, 10.4.28-MariaDB

Az adatbázis kezelő programot a választásra esett mivel az elmúlt tanulmányaink alatt jártasságot szereztünk benne és a projekthez szükséges adatok taroláséra alkalmas elemek találhatóak benne.

Ek diagrammot a online.visual-paradigm ingyenes verziójában történt

# **Frontend**

***Felelős: Tán Gergő***

Frontend: Backend adatbázis kapcsolata (Early access)

Frontend: Regisztráció, bejelentkezés, kijelentkezés (Early access)

# Backend

***Felelős: Klubert Bálint***

A Backend szerverünk a Backend nevű mappában található. Futtatásához szükséges a Node.Js. Projektünk v18.15.0 verzióban készült ezért ajánljuk, hogy legalább ezt, vagy ennél magasabb verziót használjon.

Fejleszés esetén terminálban a következő parancssal indítható: npm run dev

Általános indítás esetén pedig a következővel: npm start

## app.js

Az app.js fájl tartalmazza a backend szerver beállításait, a route-ok beállításait, a szükséges fájlok importálásait, a táblák közötti kapcsolatok és adatbázis-szerver kapcsolat kiépítését illetve a modellek szinkronizálását és a szerver elindításához szükséges kódot.

A const express = require(’express’);- el importáljuk az express package-t, hasonlóképp a többi, a szerverhez szükséges csomagot is. Hasonlóképp meghívjuk a route-okat és modelleket is, hogy a szerver indításkor használni, és még fontosabban a kéréseket kiszolgálni képes legyen.  
A szerver definiálása után adjuk meg az app-nak, hogy használja a route-okat és sequelize ModelManager-nek használatra hozzáadjuk a modelleket. Végül elindítjuk az adatbázissal a szinkronizálást, és ha az sikeres (ehhez szükséges az is, hogy az adatbázis fusson) akkor a szerver is elindul.

//Package-ek importálása

const express = require('express');

const cors = require('cors');

const { idozitettFLTorles } = require('./idozitettFeketeListaTorles')

//Adatbázis kapcsolat importálása

const sequelize = require('./adatbazisKapcsolat')

//Modellek importálása

const FelhasznaloModel = require('./models/felhasznalo.model');

const TuraModel = require('./models/turak.model');

const TuraraJelentkezes = require('./models/turaJelentkezes.model');

const feketeLista = require('./models/feketeLista.model');

const GombakModel = require('./models/gomba.model');

//Táblák közötti kapcsolatok

FelhasznaloModel.hasMany(TuraModel, { foreignKey: 'Letrehozo', as: 'TuraLetrehozasok'});

TuraModel.belongsTo(FelhasznaloModel, { foreignKey: 'Letrehozo', as: 'LetrehozoNeve'});

TuraModel.belongsToMany(FelhasznaloModel, { through: TuraraJelentkezes, as: 'JelentkezoId'});

FelhasznaloModel.belongsToMany(TuraModel, { through: TuraraJelentkezes, as: 'JelentkezettTuraId'});

//route-ok importálása

const regisztracioRouter = require("./routes/regisztracio.route");

const bejelentkezesRouter = require("./routes/bejelentkezes.route");

const fooldalRouter = require('./routes/fooldal.routes');

const turakRouter = require('./routes/turak.route');

const profilRouter = require('./routes/profil.route');

const gombaRouter = require('./routes/gomba.route');

//Az express Szerver konfigurálása

const app = express();

app.use(express.json());

app.use(cors({ origin: 'http://localhost:4200', credentials: true }));

const PORT = 3000;

//route-ok definiálása

app.use("/", fooldalRouter);

app.use("/",regisztracioRouter);

app.use("/",bejelentkezesRouter);

app.use("/", turakRouter);

app.use("/", profilRouter);

app.use("/", gombaRouter);

//Szerver indítása és az adatbázissal való kapcsolatfelvétel. Ha sikertelen a kapcsolat létrehozása az adatbázissal, a szerver nem indul el. Ha sikeres a kapcsolat, az időzített Feketelista törlés is elindul.

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

  console.log('Sikeres kapcsolat az adatbázissal!');

  sequelize.modelManager.addModel(FelhasznaloModel);

  sequelize.modelManager.addModel(TuraModel);

  sequelize.modelManager.addModel(TuraraJelentkezes);

  sequelize.modelManager.addModel(feketeLista);

  sequelize.modelManager.addModel(GombakModel);

  sequelize.sync({}).then(() =>{

    app.listen(PORT, () => {

      console.log(`A szerver elindult és elérhető a http://localhost:${PORT} URL-en!`)

      idozitettFLTorles.start()

    });

  })

}).catch((error) => {

  console.log("Az adatbázissszerverrel való kapcsolat sikertelen")

  console.log(error);})

## adatbazisKapcsolat.js

Az adatbazisKapcsolat fájlban definiáljuk az adatbáziskapcsolatot a sequelize-nak. Megadjuk az adatbázis nevét, a felhasználónevet és a jelszót, az időzónát. Ez alapján tudja majd a sequelize, hogy mely adatbázist használja, melyekbe szinkronizálja a táblákat.

//sequelize importálása

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

//Kapcsolat definiálása a sequelize-nak

const sequelize = new Sequelize('gombaproba','root','',{

    host: 'localhost',

    dialect: 'mariadb',

    define: {

        freezeTableName: true,

    },

    dialectOptions: {

        useUTC: false

    },

    timezone: "+01:00",

    typeCast: true

})

module.exports = sequelize;

## idozitettFeketeListaTorles.js

Ez a fájl tartalmazza a node-cron által beállított, óránként automatikus törlést a feketelista táblából. A feketelistában a kijelentkezett felhasználók tokenjeit tároljuk. Erre azért van szükség, hogy elkerülhessük azt az eshetőséget, hogy valaki valahogy megszerezve egy már kijelentkezett felhasználó még le nem járt tokenjét, és azt felhasználva hozzáférjen az adott felhasználó profiljához és bizonyos adataihoz. Viszont mivel a tokenek a lejáratuk után nem használhatóak – invalidak – ezért olyankor már felesleges tárolni őket. A node-cron így tehát remek megoldást biztosít arra, hogy feleslegesen letárolva tartsuk ezeket a tokeneket.

Egy for ciklussal kiszedjük a táblából lekért tokeneket, majd azokat kicsomagolva és így megkapva a lejárati idejüket ellenőrizzük, hogy a lejárati idejük kisebb-e, mint az aktuális idő. Ha igen, akkor az azt jelenti, hogy lejárt, és töröljük a táblából.

//Package-k importálása

const cron = require('node-cron');

const feketeLista = require('./models/feketeLista.model');

const jwt = require('jsonwebtoken');

//node-cron beállítása. Óránként ellenőrzi a feketelistát

var idozitettFLTorles= cron.schedule('0 \* \* \* \*', async() => {

    try {

        const osszesFeketeListasToken = await feketeLista.findAll();

        if (!osszesFeketeListasToken) {

            return;

        }

        //minden egyes letárolt token ellenőrzése. Ha a lejárati ideje kisebb, mint a jelenlegi dátum, vagyis a token lejárt így pedig használhatatlan, törli a táblából.

        for(egyDbListasToken of osszesFeketeListasToken) {

            const token = egyDbListasToken.token;

            try {

                const decodedToken = jwt.decode(token)

                if(decodedToken.exp \*1000 < Date.now()) {

                   await egyDbListasToken.destroy();

                   console.log("Lejárt JWT törölve a Feketelistából!");

                }

            }

            catch (error) {

                console.log(error);

                console.log("Hiba a token dekódolásakor!");

            }

        }

    }

     catch (error) {

        console.log(error);

        console.log("Hiba történt a token törlése közben!");

    }

    console.log("Időzített törlés a backlistből beállítva!");

},

{

    scheduled: true,

    timezone: "Europe/Budapest"

}

);

module.exports = { idozitettFLTorles };

## AuthMiddleware.js

Az AuthMiddleware a felhasználók bejelentkezésének meglétét ellenőrzi azokon a route-okon, amelyek megtekintéséhez vagy adott funkciók használatához szükséges a felhasználói fiók használata. Ilyen például a túrák létrehozása.

A Middleware először megkapja a tokent a headers-ből, ezután (amennyiben kapott tokent) szétbontja, ezzel eltávolítva a Bearer-t, majd leellenőrzi, hogy a Token nincs-e benne a feketelistában. Ha nincs, dekódoljuk, és további használatra belerakja a req.user-be a felhasználó tokentben letárolt információit, majd továbbengedi a folyamatot az adott Controllernek.

//Package-ek importálása

const jwt = require('jsonwebtoken')

const feketeLista = require('../models/feketeLista.model');

//middleware elkészítése

const tokenErvenyesites =async (req, res, next) => {

    var token = req.headers.authorization; //A header authorization részéből a token kiszedése

  //Eshetőség: Nincs token

    if (!token) {

      res.status(401).json({

        error: true,

        status: 401,

        message: "Hozzáférés megtagadva: bejelentkezés szükséges!"

      });

      return;

    };

    //Ellenőrzés, hogy a token nem szerepel-e a feketelistában

    token = token.split(' ')[1]

    const listaEleme = await feketeLista.findOne({ where: { token: token } } )

    if (listaEleme) {

      res.status(401).json({

        error: true,

        code: 401,

        message: "Hozzáférés megtagadva: Ez a token már blacklisten van!"

      });

      return;

    }

      try {

        const decodedToken = jwt.verify(token, 'titkositokulcs');

        req.user = decodedToken;

        if (decodedToken) {

            next();

        }

      } catch (error) {

        console.log(error);

        res.status(401).json({

            error: true,

            status: 401,

            message: "nem megfelelő token"

        });

        return;

      }

    };

module.exports = tokenErvenyesites;

## regisztracioMegkotesek.js

A regisztrációhoz különböző és konkrét feltételeknek kell megfelelni ahhoz, hogy a regisztráláskor megadott adatok letárolásra kerüljenek. Ahhoz, hogy a kód átláthatóságát növeljük, ezek a követelmények egy külön fájlba lettek helyezve melyet a validate.js package kezel.   
A fájl hasonló sorrendben kezeli a megkötéseket, mint amilyenekben a regisztációnál kezeljük az adatokat, az első tehát itt is a felhasználónév. Mindenképp meg kell, hogy adva legyen egy felhasználónév (ez megtalálható az emailnél és a jelszónál is ugyanezen célból), a hossza 10 és 32 karakter közöttinek kell lenni és a hozzá tartozó regexnek kell megfelelnie (tartalmazhat kisbetűt, nagybetűt és számot).  
A következő az email cím, melynek email formátumúnak kell lennie. Az email: true segítségével nincs ezzel további teendőnk, a validate-ben meghatározott séma alapján képes ellenőrizni az email cím helyes formátumát.  
A végére marad a jelszó, ami minimum 10 karakter hosszúnak kell lennie, és meg kell felelnie a hozzá tartozó regexnek (legalább egy kisbetűt, nagybetűt és számot kell, hogy tartalmazzon). Abban az esetben, ha valami nem teljesül, akkor a megfelelő message küldődik vissza.

let regisztracioMegkotesek = {};

const regexFelhasznalo = "[\-\'A-Za-z0-9áéíóöőúüűÁÉÍÓÖŐÚÜŰ]+";

const regexJelszo = /^(?=.\*[a-z+áéíóöőúüű])(?=.\*[A-Z+ÁÉÍÓÖŐÚÜŰ])(?=.\*\d).+$/;

regisztracioMegkotesek.felhasznalonev = () => {

    const megkotes = {

        'presence': {

            allowEmpty: false,

            message: "Kötelező megadni felhasználónevet!"

        },

        'type': 'string',

        'length':{

            'minimum': 10,

            'maximum':32,

            'message': 'A felhasználónév túl rövid vagy túl hosszú! A felhasználónévnek 10 és 32 karakterhossz közöttinek kell lennie!'

        },

        'format': {

            'pattern' : regexFelhasznalo,

            'flags' : 'i',

            'message':'A felhasználónévnek a következő sémát kell követnie: Nagybetű A-Z, kisbetű a-z, számok 0-9'

        }

    }

    return megkotes;

}

regisztracioMegkotesek.email = () => {

    const megkotes = {

        'presence': {

            allowEmpty: false,

            message: "Kötelező megadni jelszót!"

        },

        'type': 'string',

        'email':true

    }

    return megkotes;

}

regisztracioMegkotesek.jelszo = () => {

    const megkotes = {

        'presence': {

            allowEmpty: false,

            message: "Kötelező megadni jelszót!"

        },

        'type': 'string',

        'length':{

            'minimum': 10,

            message: "A jelszónak legalább 10 karakter hosszúnak kell lennie!"

        },

        'format': {

            'pattern' : regexJelszo,

            'flags' : 'i',

            'message': "A jelszónak a következő karaktereket kell tartalmaznia: Kisbetű, nagybetű, szám."

        }

    }

    return megkotes;

}

module.exports = regisztracioMegkotesek;

## Modellek

A modellek a sequelize használata miatt lényegesek. Segítségükkel képes a sequelize kezelni az adatbázis tábláit, azok egyedeit, oszlopait, így ismeri fel, milyen a felépítsük a tábláknak, és így tudja belőlük a megfelelő információt is elérni.

**Modelljeink**:

* feketeLista.model.js
* felhasznalo.model.js
* gomba.model.js
* komment.model.js
* poszt.model.js
* turaJelentkezes.model.js
* turak.model.js

Mivel a modelleknek a felépítése szinte ugyanolyan, így csak kettő lesz felhozva példának: A felhasznalo.model.js, és a turaJelentkezes.model.js – utóbbi azért, mert kapcsolótáblaként kissé eltér a többitől.

### felhasznalo.model.js

A modell elején importálásra kerül a DataTypes a sequelize package-ből, és példányosítjuk az adatbáziskapcsolatot.

Ezután jön az tábla sémájának megadása. A táblának az összes oszlopát definiálni kell, hogy minden értékét láthassa a sequelize. meg kell adni legalább az adott egyed letárolt értékének a típusát (például: type: DataTypes.STRING), illetve érdemes azt is megadni, hogy az adott rekord beérkezésekor kihagyható-e valamelyik érték megadása. A tábla elsődleges kulcsa esetén meg kell adni a primaryKey: true-t, ezzel jelezve ezt a sequelize-nak.  
A tableName-mel az adható meg, hogy a táblát milyen névvel építse fel/milyen néven van létrehozva az adatbázisban.

//Package és db importálása

const { DataTypes } = require('sequelize');

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

//Model definiálása

const Felhasznalo = sequelize.define('Felhasznalok', {

    User\_id: {

        type:DataTypes.STRING,

        allowNull: false,

        primaryKey: true

    },

    Felhasznalonev: {

        type: DataTypes.STRING,

        allowNull: false,

    },

    Jelszo: {

        type: DataTypes.STRING(60),

        allowNull: false,

    },

    Email: {

        type: DataTypes.STRING,

        allowNull: false,

    },

    Szuletesi\_ido: {

        type: DataTypes.DATEONLY,

        allowNull: true

    },

    Telefon\_szam: {

        type: DataTypes.STRING(12),

        allowNull: true,

        defaultValue: ""

    },

    FelhasznaloProfilKep: {

        type: DataTypes.STRING(50),

        allowNull: true

    }

},

{

    tableName:"Felhasznalok"

});

module.exports = Felhasznalo;

### turaJelentkezes.model.js

A turaJelentkezes egy kapcsolótábla, melyben a túrára jelentkező személyt kapcsoljuk össze azzal a túrával, amelyre jelentkezett. Ez az adott személy és az adott túra egyedi azonosítójának letárolásával történik. Bár be lehet állítani a modellen belül is a rekordokat, mi csak a kapcsolatnál tettük ezt meg, így a kapcsolat kiépítésekor automatikusan létrehozta őket. Mivel a kapcsolótáblához felesleges, így a removeAttribute-al eltávolításra kerül az id.

//db importálása importálása

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

//Model (kapcsolótábla) definiálása

const TuraraJelentkezes=sequelize.define("TuraraJelentkezesTabla",

{

},

{

    timestamps: false,

});

    TuraraJelentkezes.removeAttribute('id');

module.exports=TuraraJelentkezes

A sequelize modeljeinek további felépítéséről és használatáról bővebb információ található a <https://sequelize.org/docs/v6/core-concepts/model-basics/> oldalon.

## Controllerek

A Controllerekben vannak megírva a CRUD műveletekhez tartozó, http kéréseket feldolgozó metódusok. Minden fájl a nevéhez tartozó feladathoz szolgáltatja a szükséges adatokat, információkat. A frontend és a backend JSON formátumú állományokon keresztül kommunikál, kivételek ez alól azok az adatok, amelyek a headers-ben történnek átadásra a backendnek. Az alábbiakban a controllerek tartalma kerül kifejtésre, a végén egy-két példával.

**Controllerjeink:**

* bejelentkezes.controller.js
* fooldal.controller.js
* forum.controller.js
* gomba.controller.js
* profil.controller.js
* regisztracio.controller.js
* turak.controller.js

### bejelentkezes.controller.js

A bejelentkezes.controller.js tartalmazza a bejelentkezésnek a POST metódusához, és a kijelentkezés POST metódusához a controllereket. A bejlelentkezesPOSTController először megkeresi a felhasználót az email cím alapján, majd ellenőrzi, hogy a megfelelő jelszó lett e megadva, akkor létrehoz egy json web tokent a felhasználónak, és visszaküldi a frontendnek az adatokat.  
A kijelentkezesPOSTController a kapott tokent spliteli, majd feltölti a feketelistába a már splitelt tokent.

### forum.controller.js

A forum.controller.js tartalmazza a fórum aloldalon történő posztok és kommentek megtekintéséhez, létrehozásához, módosításához és törléséhez szükséges controllereket. A poszthoz kettő GET metódus társul: az első az összes posztot és annak létrehozóját adja vissza (ez fut le elsőként és alapértelmezetten a fórum aloldalon), a másik pedig a felhasználó által megtekintett poszt létrehozóját, tartalmát és a hozzá tartozó kommenteket és azok létrehozóainak felhasználónevét adja vissza. A PATCH hasonlóan működik, mint a felhasználói vagy a túra adatok módosításánál. A törlés esetén nem tároljuk a posztokat adminisztrációs célból, töröljük az adatbázisból. Hasonlóképp történik a kommenteknél is.

### gomba.controller.js

A gomba.controller.js a legkisebb controllerünk. A feladata annyi,, hogy az enciklopédiához biztosítsa a gombákat, amiket a továbbiakban a frontend kezel.

### profil.controller.js

A profil.controller.js-ben lévő controllerek feladatai, hogy visszaadja a bejelentkezett felhasználó adatait, amik megjelennek a saját profilon. Ezenkívül a felhasznaloPATCHController futásakor tudja a felhasználó megadni a telefonszámát és életkorát, előbbit bármennyiszer, utóbbit viszont egyszer, ezt a frontend kezeli. Végezetül DELETE törlés esetén a felhasználói profil törlődik.

### regisztracio.controller.js

A regisztracio.controller.js a regisztráció elvégzéséhez szükséges PUT metódus, és egy, a jelszó titkosításához használt function. A beérkező adatok először ellenőrzésre kerülnek, egyrészt, hogy meg lett-e adva bármi, másrészt, hogy megfelelnek-e a regisztracioMegkotesek.js-ben megírt feltételeknek. Ezenfelül ellenőrizzük, hogy a jelszó és a jelszó ismétlése megegyezik-e, vagy, hogy létezik-e már ilyen felhasználónévvel vagy email címmel felhasználó. Ha minden megfelel, akkor létrejön a felhasználó, a jelszavát pedig megkapva a titkosításra írt metódus hash-eli, és így kerül letárolásra az adatbázisban biztonsági okokból.

### turak.controller.js

A turak.controller.js-ben található a túrákhoz szükséges három GET, egy POST, egy PUT, egy PATCH, és a túrákra való jelentkezés törléséhez egy DELETE controller. A három GET kisebb eltérések vannak. Az osszesTurakGETController az összes túrát adja vissza, azok adatait, a létrehozó nevét, a jelentkező számát, és küldött token esetén azoknak a túráknak az egyedi azonosítóját, amelyekre a bejelentkezett felhasználó jelentkezett. A turakGETController azokat a túrákat küldi vissza a frontendnek, amiket az adott felhasználó hozott létre. A harmadik, turakJelentkezettGETController pedig azokat adja vissza, amelyekre a bejelentkezett felhasználó jelentkezett.  
A turakPOSTControllerben van megírva a túrára való jelentkezés folyamata. Először is megérkezis a bejelentkezett felhasználó azonosítója és a túra azonosítója. Ezután leelenőrzi, hogy a jelentkező betöltötte-e a 18. életévét. Ha mindent rendben talált, feltölti az érkékeket a kapcsolótáblába.  
A turakPUTController hajtja végre a túra létrehozását és letárolását. A controller az értékek megkapása után ellenőrzi, hogy a bejelentkezett felhasználó betöltötte-e a 18. életévét, vagy sem. Ezután végigmegy az értékeken, hogy hiányzik-e valami. Mivel a frontend az üresen hagyott érték esetén egy üres Stringet küld, így nem az undefined, hanem ”” érték van megadva a feltételben. Ellenőrzi azt is, hogy a megadott indulási idő nagyobb-e, mint a jelenlegi idő, és, hogy az érkezési nagyobb-e, mint az indulási. Az esetben, ha a szervező nem ad meg elérhetőséget, a controller automatikusan hozzárendeli a létrehozó email címét. Ha minden megfelel, akkor létrejön a túra.   
A turakPATCHController felel a túra adatainak módosításáért. Mivel itt is az az eset áll fent az adat küldésekor, mint a PUT-nál, így először lekéri a módosítani kívánt túra adatait az adatbázisból, majd ahol üres String található a küldött body-ban, ott a body értékét kicseréli a letároltra. ezután fut le a sequelize update metódusa. Ha a rowsAffected első indexének az értékének a hossza 0, az azt jelenti, hogy nincs ilyen túra az adatbázisban. Ha nem nulla, és a nulladik indexen lévő érték 1, az azt jelenti, hogy van különbség a letárolt és kapott értékek között és a frissítés végbe megy. Ha nem 1 az érték, akkor a letárolt és módosítani kívánt értékek között nincs különbség, ilyenkor egy 200-as státuszkóddal válaszol, ám a message tartalmazza, hogy nincs különbség, a módosítás így nem is történik meg.  
**MEGJEGYZÉS:** Mivel adminisztrációs célból a túrákat megtartjuk, így azok nem törlődnek az adatbázisból, hanem lefuttatva a turakPATCHControllert az Elmarad\_a\_tura értéke true lesz, és nem jelenik meg a felhasználóknak

Az utolsó controller a jelentkezesDELETEController. Ez megkapva a túra és a felhasználó egyedi azonosítóját ellenőrzi, hogy van-e ilyen kapcsolat, és ha van, törli a jelentkezést a táblából.

### Pár példa a fenti Controllerekből:

#### bejelentkezesPOSTController, token létrehozás:

//JWT Token létrehozása

    let token;

    try {

        token = jwt.sign(

            {

                userId: letezoFelhasznalo.User\_id,

                email: letezoFelhasznalo.Email,

                felhasznalonev: letezoFelhasznalo.Felhasznalonev

            },

            "titkositokulcs",

            { expiresIn: "1h" }

        );

    }

#### regisztracioPUTController, jelszó validálás hitelesítési példaként, adatok feltöltése:

    //jelszó lekezelése

    if (jelszo === '') {

        res.status(400).json({

            code:400,

            type: 'Nincsjelszo',

            message:"Nem lett megadva jelszó!"

        })

        return;

    }

    else if(jelszo !== jelszoUjra) {

        res.status(400).json({

            code:400,

            type: 'Nemegyezojelszavak',

            message:"A két jelszó nem egyezik!"

        })

        return;

    }

    else{

        let msg = validate.single(jelszo, regisztracioMegkotesek.jelszo);

        if (msg) {

            res.status(400).json({

                code: 400,

                type: "Nemjojelszo",

                message: msg

            })

            return;

        }

    }

    felhasznalo = await felhasznaloModell.build({

        User\_id: uuidv4(),

        Felhasznalonev: felhasznalonev,

        Email: email,

        Jelszo: await jelszoHash(jelszo),

        Szuletesi\_ido: szuletesiIdo,

        Telefon\_szam: telefonSzam

    })

    await felhasznalo.save();

    res.status(200).json({felhasznalo, success: true});

};

#### osszesTurakGETController, kapcsolatok általi lekérdezés, mapping:

        const turak = await Turak.findAll({

            attributes: {

                include: [[Sequelize.fn("COUNT", Sequelize.col("JelentkezoId.User\_id")), "Jelentkezok"]],

            },

            include: [

                {

                    model: Felhasznalo,

                    as: "JelentkezoId",

                    attributes: []

                },

                {

                    model: Felhasznalo,

                    attributes: ["Felhasznalonev"],

                    as: "LetrehozoNeve",

                },

            ],

            group: ['Tura\_id'],

        // Manuális dátumformázás és csak a szükséges mezők hozzáadása

        const formazottTurak = turak.map(tura => ({

            Tura\_id: tura.Tura\_id,

            Tura\_neve: tura.Tura\_neve,

            Indulas\_ido: new Date(tura.Indulas\_ido).toLocaleString(),

            Indulas\_hely: tura.Indulas\_hely,

            Erkezesi\_ido: new Date(tura.Varhato\_erkezesi\_ido).toLocaleString(),

            Erkezesi\_hely: tura.Erkezesi\_hely,

            Utvonal\_nehezsege: tura.Utvonal\_nehezsege,

            Szervezo\_elerhetosege: tura.Szervezo\_elerhetosege,

            Tura\_dija: tura.Tura\_dija,

            Elmarad\_a\_tura: tura.Elmarad\_a\_tura,

            Leiras: tura.Leiras,

            Jelentkezok: tura.Jelentkezok,

            Felhasznalonev: tura["LetrehozoNeve.Felhasznalonev"],

        }));

        res.status(200).json({

            turak: formazottTurak

        });

    }

#### turakPOSTController, életkor ellenőrzése:

        const felhasznalo = await Felhasznalo.findByPk(felhasznaloId);

        if (felhasznalo.Szuletesi\_ido === null) {

            res.status(400).json({

                error: true,

                type: "nincsSzulEv",

                status: 400,

                message: "Nem lett életkor megadva! Túrára csak olyan jelentkezhet, aki betöltötte a 18. életévét!"

            });

            return;

        }

        const tizennyolcE = ((new Date())-(new Date(felhasznalo.Szuletesi\_ido)))/1000/60/60/24/365.25

        if (tizennyolcE < 18) {

            res.status(403).json({

                error: true,

                status: 403,

                message: "Legalább 18 évesnek kell lenni, hogy túrára tudjon jelentkezni!"

            })

            return;

        }

## Route-ok

A routes mappa tartalmazza az elérési utak definiálását. A Controllerek ide kerülnek meghívásra, majd összekapcsoljuk a megfelelő Controllert a megfelelő route-hoz, és vele együtt az érkező HTTP kérés típusához is.

**Route-jaink:**

* bejelentkezes.route.js
* fooldal.route.js
* forum.route.js
* gomba.route.js
* profil.route.js
* regisztracio.route.js
* turak.route.js

A http beállításhoz az Express csomag Router function-jét alkalmazzuk.  
Azokhoz a route-okhoz, amelyeket csak bejelentkezés megléte után lehet elérni, hozzácsatoljuk még az AuthMiddleware-t is, ezzel biztosítva azt, hogy illetéktelenek és/vagy jogosultsággal nem rendelkezők ne férjenek hozzá az adott funkciókhoz. A végén exportálásra kerül a beállított Router, hogy a szervert futtató fájl (app.js) hozzáférhessen.

//Express és az AuthMiddleware importálása

const express = require('express');

const AuthMiddleware = require('../middlewares/AuthMiddleware');

//A controller és a Router importálása, inicializálása

const turakController = require('../controllers/turak.controller');

const turakRouter = express.Router();

//Route-ok létrehozása, ahol szükséges, a route elérési útja és a megfelelő controller közé az AuthMiddleware elhelyezése

turakRouter.get("/turak/osszes", turakController.osszesTurakGETController);

turakRouter.get("/turak/jelentkezett",AuthMiddleware, turakController.turakJelentkezettGETController);

turakRouter.delete("/turak/jelentkezett",AuthMiddleware, turakController.jelentkezesDELETEController);

turakRouter.get("/turak",AuthMiddleware , turakController.turakGETController);

turakRouter.put("/turak",AuthMiddleware , turakController.turakPUTController);

turakRouter.post("/turak",AuthMiddleware , turakController.turakPOSTController);

turakRouter.patch("/turak", AuthMiddleware, turakController.turakPATCHController);

turakRouter.delete("/turak", AuthMiddleware, turakController.jelentkezesDELETEController);

module.exports = turakRouter

## Táblázat a végpontokhoz

Az alábbi táblázat a végpontok, az adott végponton használt metódust, a feladatot és a visszaküldött http státuszkódokat tartalmazza azok egyszerűbb megtalálása és megértése érdekében.

|  |  |  |  |
| --- | --- | --- | --- |
| **VÉGPONT** | **HTTP METÓDUS** | **FUNKCIÓ** | **STÁTUSZKÓDOK** |
| *Regisztráció* | | | |
| /regisztracio | put | A beérkező regisztrációs adatokat ellenőrzi, majd letárolja. | 201 – Sikeres regisztráció!  400 – Valamilyen hiba adódott, konkrét információt a regisztracioMegkotesek biztosít.  500 –Szerver hiba |
| *Bejelentkezés* | | | |
| /login | post | Megtekinti a bejelentkezési adatokat, összehasonlítja a letároltakkal és a felhasználónak egy tokent készít, amit visszaküldd a Frontendnek |  |
| /logout | post |  |  |

# **Adatbázis**

***Felelős: Erdei Gábor***

MushroomTour adatbázis célja: tervezés során a gomba olyan elemeit, amit nem szakmai emberek számára könnyen meghatározható az erdőben talált gombákat semmilyen plusz segédeszköz nélkül.

A könnyű összekeverhetőség miatt több mérges gombát is bele helyezünk, két galócát egy tinóru és egy papsapka gombát a figyelmeztetéssel, amivel felhívjuk a figyelmet más fajokra és a tervek alapjén további gombákat is feltöltjük.

**Felhasznált programok:** Xampp , 10.4.28-MariaDB

**Kódolás**: utf8\_hungarian\_ci

## zemplen\_gombai

Az adatbázis segíti a gombásznak könnyen meghatározni a gombák egyedi jellemzőit. Amit semmilyen segédeszközt nem igényel. De ezek az adatok időjárásnak, külső behatásnak megfelelően változhatnak. A következő években lévő kutatások rávilágíthatnak, hogy egyes fajok több alfajból is álhatnak. Ezért mindig ellenőrizze gombaszakellenőrrel a termőtesteket.

***Id kivételével engedélyezett a null***

**Id(auto\_increment)**  int(10) elsődleges kulcs az adatok pontos darabszámának meghatározása érdekében

**Latin\_nev** varchar (35) a külföldi vagy másképpen nevezett gomba könnyebb azonosítás érdekében

**Magyar\_nev** varchar (35) Zemplénben magyar nyelvű gomba neve

**Nemzetseg** varchar (20) a gomba nemzetségének meghatározása okán

**Ethezes\_erteke** tinyint (1) 0-ehető 1-mégező 2-nem ehető 3-védett

**Feltetel** varchar (50) van-e az étkezéshez valami fontos információ **20 perc főzést igényel , nem iható mellé alkohol**

**Leiras** varchar (150) egy rövid leirés amit, a növényzetet és az évszakot, amikor terem

**Bocskora** tinyint (1) van ezen a termőteste bocskor 0 vagy 1

**Galler** tinyint (1) van ezen a termőteste gallér 0 vagy 1, **ami lehet, hogy az időjárás miatt eltűnhetett**

**Szezon eleje** tinyint (2) a termőtest első növekedésének hónapja, amit számokkal adtunk meg 1- 12 januártól -decemberig

**Szezon vege** tinyint (2) a termőtest utolsó növekedésének hónapja, amit számokkal adtunk meg 1- 12 januártól -decemberig

**Novenyzet** varchar (100) a termőtest milyen talajon terem és kedvel, hol keresse azt a felhasználó

**Termotest\_tipus** varchar (30) a felhasználó milyen gombát hol keressen, illetve milyen alakú lehet

**Termoretegtarto\_tipusok** varchar (25) a természetben található termőrétegtartó

**Kalapforma** varchar (30) a természetben található kalapforma meghatározása

**Kalap\_felszin** varchar (35) a természetben található kalap felszíne meghatározása

**Kalap perem** varchar (30) a természetben található kalap pereme meghatározása

**Lemezallas** varchar (40) a természetben található lemezállás meghatározása

**Lemez\_el** varchar (25) a természetben található lemez él meghatározása

**Burok\_maradvany** varchar (30) a gallérnak több formája is lehet **ennek pontosabb meghatározás céljából, de elsősorban az alkalmazás tovább fejlesztésével lesz jelentősége**

**Tonk\_alak** varchar (35) a természetben található tönk alak meghatározása

**Tonk\_felulet** varchar (35) a természetben található tönk felülete meghatározása

**Hus** varchar (45) a természetben található hús meghatározása

**Hus elszinezodés** varchar (45) a természetben található hús elszíneződés meghatározása, **van rá lehetőség, hogy a hőmérséklet miatt a szín nem rögtön fog megjelenni**

**Ize** varchar (20) a gomba íze mire hasonlít

**Illat** varchar (20) a gomba illata mire hasonlít

**Ara\_HUF** int (10) a védett gombák leszedésének ára

**Kep\_keszito\_neve** varchar (30) a készült kép tulajdonosának a neve

**Kep\_neve** varchar (35) Kép neve

## Tervezet\_turak

A tábla a tervezett túrákat tartalmazza, ahova részt vehetnek Kezdés időpontja és helyét, és a befejezés idejét és helyét tartalmazza egy leírást a túra nehézségéről illetve az árát Forintban. Rossz idő esetén vagy egyéb okok miatt pedig meg egy változót hogy valóban meg lesz e tartva. Illetve tárolva van az első létrehozás és módosítás.

**Tura\_id(Primary key Foreign key,)** int(11) **AUTO\_INCREMENT**

**Tura neve** varchar(50) A túra meghirdetett neve

**Letrehozas** varchar (50) a szervező és felelős

**Indulas**\_**ido** datetime túra kezdete: év-hónap-nap

óra:perc:másodperc:2023-10-01 08:00:0

**Indulas**\_**hely** varchar (50) a túra indulásának helye

**Varható\_erkezesi\_ido** datetime túra várható vége: év-hónap-nap

óra:perc:másodperc:2023-10-01 12:00:0

**Erkezesi**\_**hely** varchar (50) A túra érkezésének helye

**Utvonal**\_**nehezseg** varchar (50) A túra nehézsége könnyű/közepes/nehéz

**Szervezo**\_**elerhetosege** varchar (50) email/telefon elérhetőség

**Tura**\_**dija** int (11) ha a túra díja kerül (alapértelmezetten 0)

**Elmarad\_a\_tura** tinyint (1) ha időjárás miatt elmaradt értesítés (alapvetően 0)

**createdAt** date A felhasználónak első bejelentkezésének

az ideje óra:perc:másodperc

**updatedAt** date A felhasználónak utolsó

bejelentkezésének az ideje óra:perc:másodperc

## Felhasznalo

A tábla a felhasználók adatait és elérhetőségét tartalmazza, a jelszót kódolva tároljuk így az adatbázis számára ismeretlen az eredeti jelszó, a profilkép szabadon változtatható, de egy alapértelmezett profilkép be van állítva. Illetve tárolva van az első létrehozás és módosítás.

**User\_id(Primary key Foreign key)** int(11) **AUTO\_INCREMENT**

**Felhasznalonev (Foreign key)** varchar (32) a bejelentkező felhasználó neve

**Jelszo** varchar (60) a felhasználó jelszava kódolva, tárolva

**Email** varchar (50) a felhasználó email címe

**Születesi\_ido** date (engedélyezett a null)a felhasználó

születési éve

**Telefon\_szam**  varchar (12) (engedélyezett a null)a felhasználó

Telefon száma

**FelhasznaloProflKep** varchar (50) a felhasználó profil képe

**createdAt** date A felhasználónak első bejelentkezésének

az ideje óra:perc:másodperc

**updatedAt** date A felhasználónak utolsó

bejelentkezésének az ideje óra:perc:másodperc

## Post\_table

A tábla a poszt adatait tartalmazza posztoló tulajdonosának és a kép nevét. illetve a kommenteket Illetve tárolva van az első létrehozás és módosítás.

**Id(auto\_increment)** int(10) Post Id

**Postolo\_neve (Foreign key)** varchar (20)Felhasználó neve

**Kep\_id** varchar (20) Kép egyedi azonosítója

**Comment** varchar (255) A commentek

**createdAt** date A felhasználónak első bejelentkezésének

az ideje óra:perc:másodperc

**updatedAt** date A felhasználónak utolsó

bejelentkezésének az ideje óra:perc:másodperc

## Jelentkezok\_turara

A tábla tartalmazza ki melyik túrára jelentkezett

**User\_id (Foreign key)** int (11) A túrára jelentkezők azonosítója

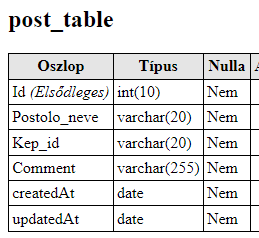
**Tura\_id (Foreign key)** int (11) A túra azonosítója

## fekete\_lista

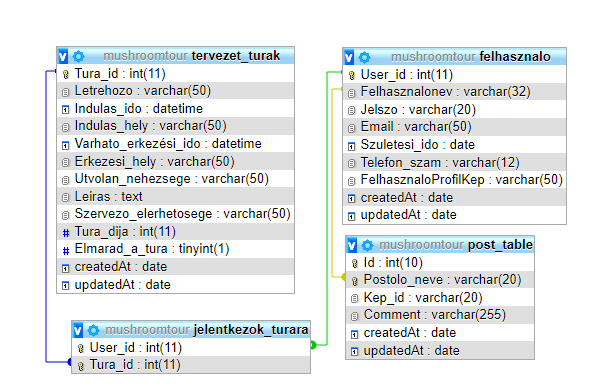
Kijelentkezett felhasználok tokenjeit tároljuk amik még nem jártak le

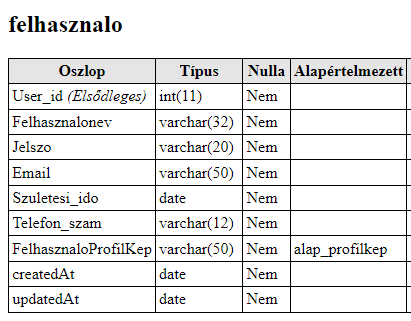
**token** varchar(300) A tokenek listája

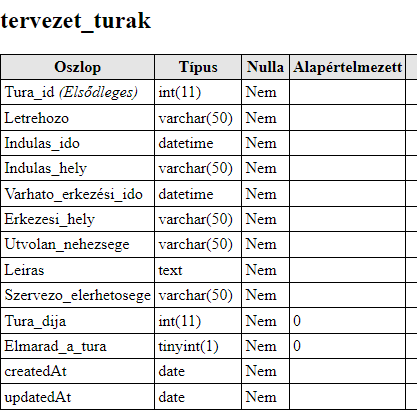
# Adatbázis Képek

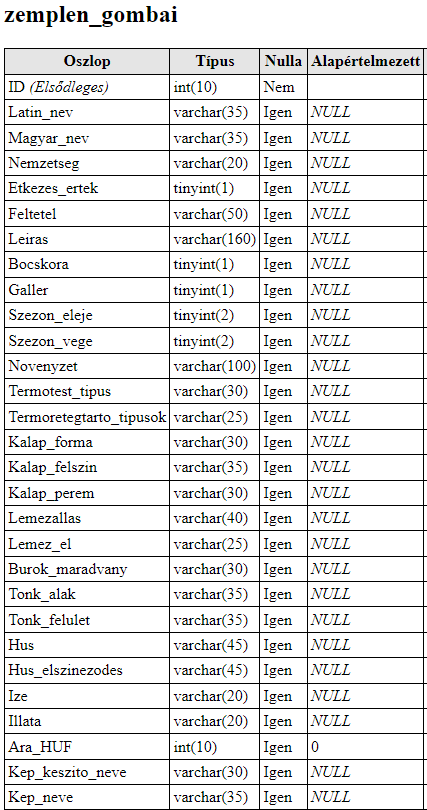
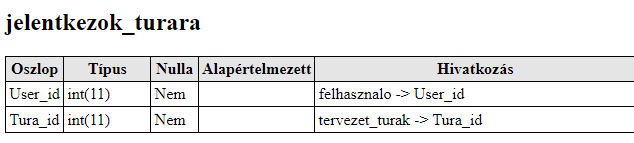


Adatbázis kapcsolatok

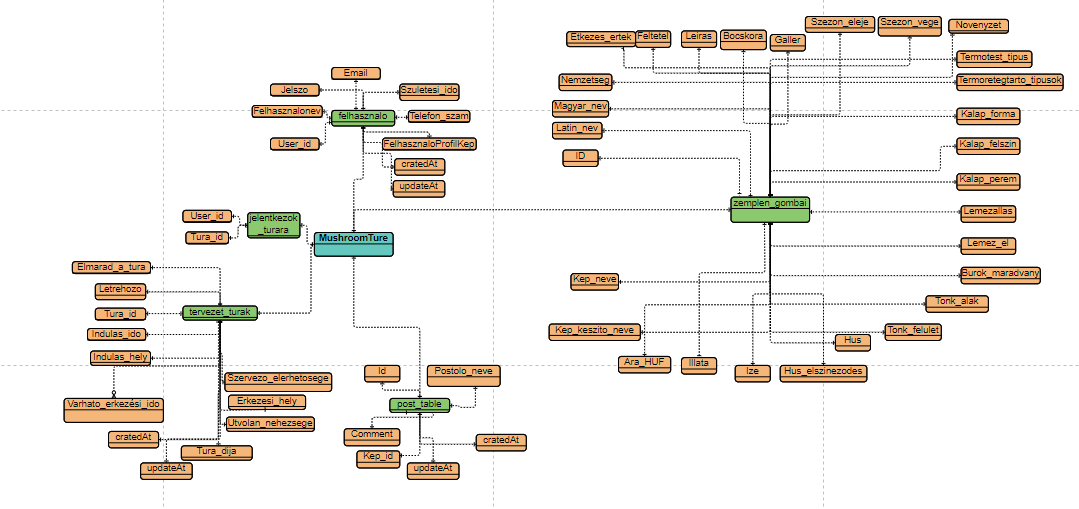


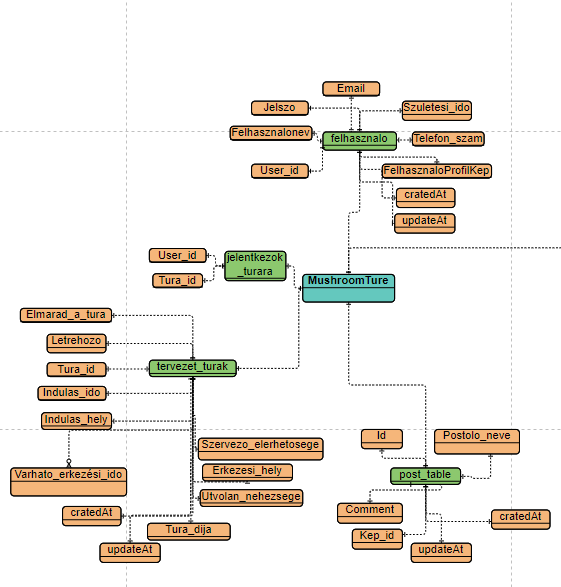


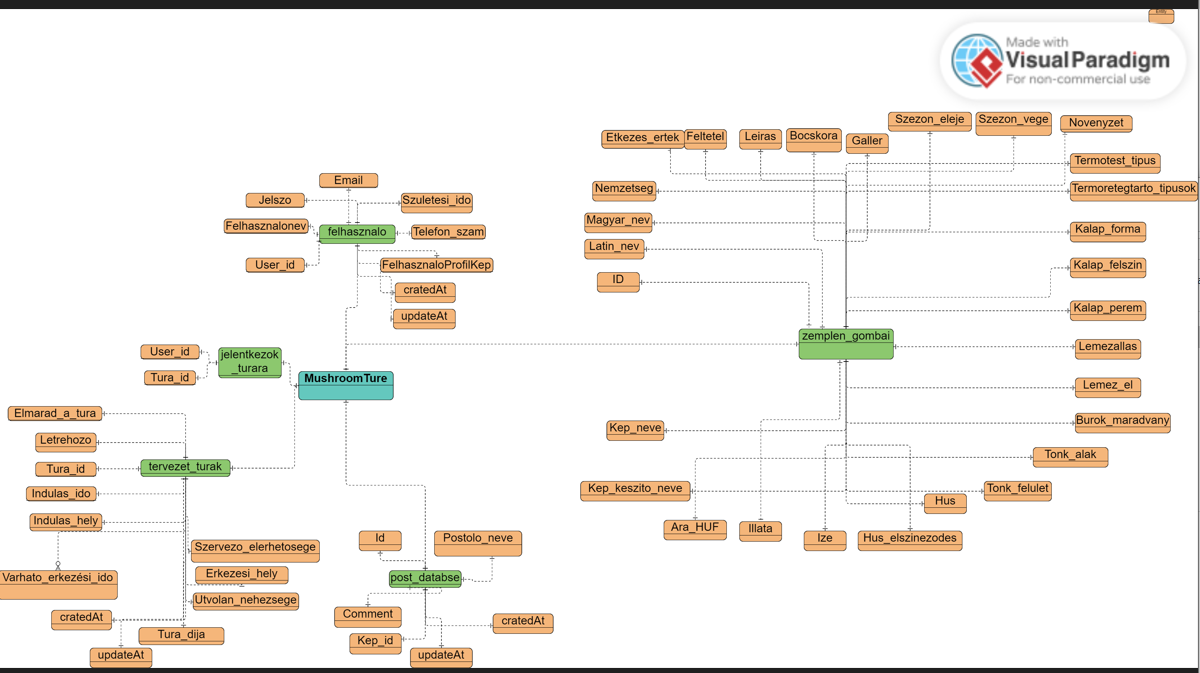




# E-k diagram







# Tesztelés

## Tesztelési terv

Név: Erdei Gábor

Projekt neve: MushroomTour

Verzió száma:

Utolsó mentés dátuma:

## 1.Bevezetés

* 1. Testelési terv hatóköré célja

A tesztelés széles körben való biztosítása előre meghatározott eljárások es megoldások által

A teszt végre hajtásáért Erdei Gábor felel, és a tesztcsoport hajtja végre a 2 pontban meghatározottak alapján

* 1. Elvárások

Az olvasó ismeri a rendszert

A Erdei Gábor Felelős a tesztadatok előállításáért

A teszt program az ebben a dokumentumban meghatározott teszttervek alapján fut.

## 2. Tesztelés

Tesztesetek alábbi adatok alapján

## 2.1 Regisztrációs oldal

**Feladat** A bejelentkezés sikeres legyen felvegy a az adatokat a z adatbázisba a jelszőt kódólt formában

**Hiba lehetőség**: Felhasználó név: Több, mint 32 vagy kevesebb, mint 8 karakter ad meg

Email: @ és pont domain végződés

Jelszó: Nagybetű kisbetű és szám minimum 10 karakter vagy a két jelszó nem azonos

**Előfeltétel**:

**Utófeltétel**:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@gmail.com  Jelszó: Laskagomba12  Jelszó ismét: Laskagomba12 |  |  |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@.com  Jelszó: Laskagomba12  Jelszó ismét: Laskagomba12 |  |  |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@gmail.com  Jelszó: Laskagomba  Jelszó ismét: Laskagomba12 |  |  |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@gmail.com  Jelszó: Laskagomba12  Jelszó ismét: Laskagomba12 |  |  |

## 2.2 Belépes oldal

**Feladat:** Az oldal a megfelelő felhasználó név és jelszó párossal beengedje a felhasználót és engedi túrát létrehozni

**Hiba** **lehetőség**: A felhasználó név és jelszó páros nem megfelelő vagy nincs a regisztrált felhasználok között

**Előfeltétel**:

**Utófeltétel**:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Felhasználó név: Első tesztelő  Jelszó: Laskagomba12 | A felhasználót beengedi, és ha meg van adva, az életkora tud túrát szervezni |  |
| Felhasználó név: Első gombász  Jelszó: Laskagomba12 | Nem létező felhasználó |  |
| Felhasználó név: Első gombász vizsga  Jelszó: Kucsmagomba12 | Ilyen felhasználó nincs regisztrálva |  |
| Felhasználó név: Első tesztelő  Jelszó: Laskagomba23 | Nem megfelelő a jelszó |  |

## 2.3 Túrajelentkezés

Feladat

Hiba lehetőség:

Előfeltétel:

Utó feltétel:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
|  |  |  |
|  |  |  |

## 2.4 Túralétrehozás

Feladat

Hiba lehetőség:

Előfeltétel:

Utó feltétel:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
|  |  |  |
|  |  |  |

## 2.5 Gomba enciklopédia

Feladat

Hiba lehetőség:

Előfeltétel:

Utó feltétel:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
|  |  |  |
|  |  |  |

## 2.6 Profil oldal

Feladat

Hiba lehetőség:

Előfeltétel:

Utó feltétel:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
|  |  |  |
|  |  |  |

## 2.7 Térkép oldal

Feladat

Hiba lehetőség:

Előfeltétel:

Utó feltétel:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
|  |  |  |
|  |  |  |

## 2.8 Forum oldal

Feladat

Hiba lehetőség:

Előfeltétel:

Utó feltétel:

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
|  |  |  |
|  |  |  |

# Tovább fejlesztési lehetőségek

* A zemplen\_gomba tábla feltöltése védett, mérgező, nem ehető és több ehető gombákkal:

A könyv közel 400 gombafajt tartalmaz ehető nem ehető védett és mérgező fogyaszthatósági kategóriában, feltölteni az adatbázisba illetve az új fajokkal való bővítésé

* Vármegye és országszintű kiterjesztés:

Vármegye és az Alföld gomba világának megjelenítésé. Illetve a magashelységek gombavilága feltüntetése.

* Tanulási lehetőségek Gomba kvíz, tanfolyamszervezés:

Tájékoztatás az esetleges tanfolyamokról miket az egyesület szervez.

Különböző tanulást segítő játékos feladatot PL:

* + Egy gomba képe alatta két vagy több gombanév és a helyeset kell kiválasztani.
  + Egy gombának a neve és alatta két vagy több kép és a helyeset kell kiválasztani
  + Az együtt élő gombafajok különböző növényekkel élnek együtt ezeknek a növényeknek bemutatása ,Illetve a védett növényeken lévő hang súly, hogy azokat ne gyűjtsék.
* Google naptárral való összehangolás:

Ha jelentkeztem egy túrára, akkor az email cím segítségével automatikusan beírható legyen a Google naptárba

* Közös tárhely a képek megosztására:

A túrán készült képek csak a csoport tagjai számára való megosztás egy külön tárhelyen. Lehetőség szerint még versenyeket kiírni a legszebb képekből feltüntetve a kép szerzőjét, majd a képeket könyvek vagy projektumkákhoz könnyen fel lehessen használni

* Év gombájának megszavaztatása:

Minden évben lehet szavazni az év gombájáról ennek lebonyolítása érdekében egy szavazást biztositani majd kihirdetni a nyertest.

# Irodalomjegyzék

Az adatbázishoz adatokat és képeket szolgáló könyv

* Gécziné Nagy Mária, Dr. Hegyessy Gábor: Gombák Zemplénből nem csak zemplénieknek (kiadó: Zempléni Gombász Egyesület) (2021) ISBN: 9786150119267
* Zempléni Gombász Egyesület fajlistája 2023