

Szent István Katolikus Technikum és Gimnázium

**Erdei Gábor,**

**Klubert Bálint,**

**Tán Gergő**

MushroomTour

**SZOFTVER FEJLESZTÉS ÉS - TESZTELÉS VIZSGAREMEK**

Sátoraljaújhely, 2023/2024

**Tartalom**

[Bevezetés 4](#_Toc163512753)

[Miért választottuk ezt a témakört? 4](#_Toc163512754)

[Felhasználói dokumentáció 4](#_Toc163512755)

[Rendszer követelmény 4](#_Toc163512756)

[Intel Core i3-6100T Dual-Core 3.2GHz 4](#_Toc163512757)

[Az alkalmazás / oldal indítása 5](#_Toc163512758)

[Főoldal 5](#_Toc163512759)

[Regisztráció 5](#_Toc163512760)

[Bejelentkezés 6](#_Toc163512761)

[Profil 6](#_Toc163512762)

[Enciklopédia 7](#_Toc163512763)

[Túrák 8](#_Toc163512764)

[Térkép 10](#_Toc163512765)

[Fórum 11](#_Toc163512766)

[Fejlesztői dokumentáció 13](#_Toc163512767)

[Használt csoportmunka eszközök bemutatása 13](#_Toc163512768)

[Fejlesztéshez használt technológiák, szoftverek 14](#_Toc163512769)

[Frontend: 14](#_Toc163512770)

[Frontend: 14](#_Toc163512771)

[Backend: 14](#_Toc163512772)

[Adatbázis: 16](#_Toc163512773)

[Frontend 16](#_Toc163512774)

[Backend 23](#_Toc163512775)

[app.js 23](#_Toc163512776)

[adatbazisKapcsolat.js 24](#_Toc163512777)

[idozitettFeketeListaTorles.js 25](#_Toc163512778)

[AuthMiddleware.js 26](#_Toc163512779)

[regisztracioMegkotesek.js 27](#_Toc163512780)

[Modellek 29](#_Toc163512781)

[felhasznalo.model.js 29](#_Toc163512782)

[turaJelentkezes.model.js 30](#_Toc163512783)

[Controllerek 31](#_Toc163512784)

[bejelentkezes.controller.js 31](#_Toc163512785)

[forum.controller.js 31](#_Toc163512786)

[gomba.controller.js 31](#_Toc163512787)

[profil.controller.js 32](#_Toc163512788)

[regisztracio.controller.js 32](#_Toc163512789)

[turak.controller.js 32](#_Toc163512790)

[Pár példa a fenti Controllerekből: 33](#_Toc163512791)

[bejelentkezesPOSTController, token létrehozás: 33](#_Toc163512792)

[regisztracioPUTController, jelszó validálás hitelesítési példaként, adatok feltöltése: 33](#_Toc163512793)

[osszesTurakGETController, kapcsolatok általi lekérdezés, mapping: 34](#_Toc163512794)

[turakPOSTController, életkor ellenőrzése: 35](#_Toc163512795)

[Route-ok 35](#_Toc163512796)

[Táblázat a végpontokhoz 36](#_Toc163512797)

[Adatbázis 40](#_Toc163512798)

[feketelista 40](#_Toc163512799)

[felhasznalok 40](#_Toc163512800)

[gomba 41](#_Toc163512801)

[kommentek 42](#_Toc163512802)

[posztok 42](#_Toc163512803)

[turak 43](#_Toc163512804)

[turarajelentkezestabla 43](#_Toc163512805)

[Adatbázis Képek 44](#_Toc163512806)

[E-k diagram 47](#_Toc163512807)

[Tesztelés 49](#_Toc163512808)

[Tesztelési terv 49](#_Toc163512809)

[1.Bevezetés 49](#_Toc163512810)

[2. Tesztelés 49](#_Toc163512811)

[2.1 Regisztrációs oldal 49](#_Toc163512812)

[2.2 Belépés oldal 50](#_Toc163512813)

[2.3 Túrajelentkezés 50](#_Toc163512814)

[2.4 Térkép 50](#_Toc163512815)

[2.5 Enciklopédia 51](#_Toc163512816)

[2.6 Profil oldal 51](#_Toc163512817)

[2.7 Túrák oldal 51](#_Toc163512818)

[2.8 Fórum oldal 51](#_Toc163512819)

[Tovább fejlesztési lehetőségek 52](#_Toc163512820)

[Irodalomjegyzék 52](#_Toc163512821)

# 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 év folyamán szereztünk, és, hogy a meglévő tudásunkat tovább mélyítsük. Mindhárman beadtuk jelentkezésünket egyetemre, hogy még magasabb végzettséget és több tudást szerezve minél könnyebben elhelyezkedhessünk ebben a szakmában.

## Miért választottuk ezt a témakört?

Témakörünk megválasztásához több dolog is közrejátszott, ám a legnagyobb talán csapatunk tagja, Erdei Gábor tapasztalatai és ötletei, ezen felül a gombászok között szerzett információk voltak. Mint tudjuk, családok számára kellemes kikapcsolódást jelenthet az erdőkben való túrázás, amit meg is koronázhatunk az erdő kincseinek gyűjtésével, amivel új ismerősök, barátok mellett, egy ízletes étkezést is nyerhetünk a kirándulások alkalmával. Sajnos előfordulnak esetek, amikor szakellenőrök egy egész kosár tartalmát nem engedik fogyasztásra, mivel mérgező gombát tartalmaz, ahogy néha bonyolódások is lehetnek a túrák szervezésekor. A célunk az volt, hogy ezeket a problémákat megoldjuk, könnyebbé téve a gombákról való információszerzést a laikusok számára, egyszerűbbé, átláthatóbbá tegyük és központosítsuk a túrák szervezését, és egy olyan közösségi felületet hozzunk létre a téma iránt érdeklődők számára, amelyen kezdők és haladók, kicsik és nagyok együttesen, egymástól is tanulva tudjanak információkat szerezni és egy közösséget kovácsolni.

# Felhasználói dokumentáció

## Rendszer követelmény

|  |  |  |
| --- | --- | --- |
|  | Ajánlott: | Minimális |
| Processzor (CPU) | Intel(R)\_Core(TM)\_i3-9100F\_CPU\_@\_3.60GHz | Intel Core i3-6100T Dual-Core 3.2GHz |
| Operációs rendszer (OP) | Windows 10 pro | Windows 10 |
| RAM: | 8 GB RAM | 6 GB RAM |
| Tárhely: | 6 GB | 6 GB |
| Internet: | Széles sávú | Széles sávú |
| GPU: | NVIDIA GeForce GTX 1050 | Intel(R) UHD Graphics 620 |

## Az alkalmazás / oldal indítása

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

* Google Chrome
* Opera

Az oldal a <http://localhost:4200/> oldalon érhető el.

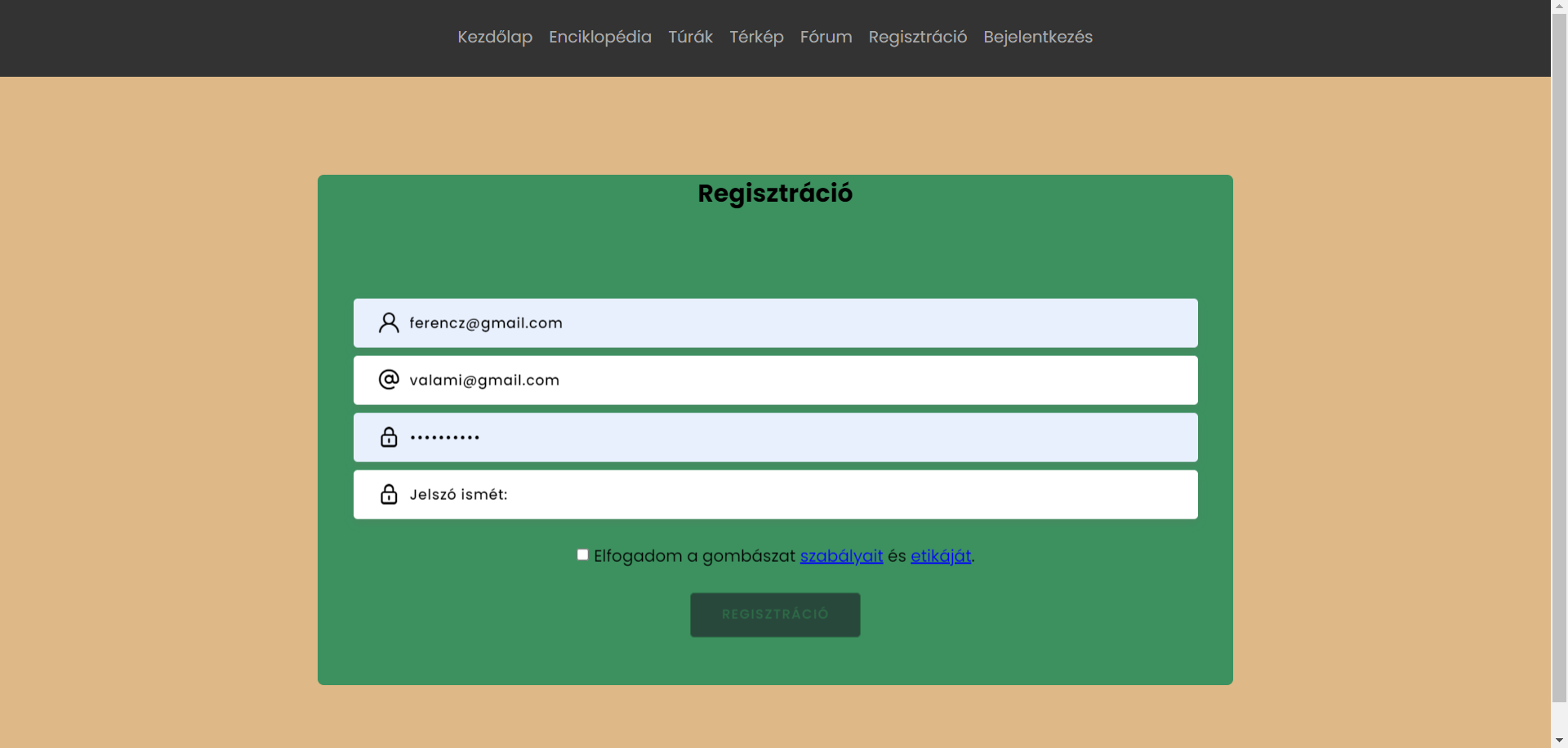
### Főoldal

Az URL meglátogatása után a főoldalon találjuk magunkat, innen érhetőek el a további funkciók a navigációs sávon. A főoldalon egy forgótár látható, mely az oldalon található funkciókat mutatja be képekben, ezen kívül, ha a képre kattintunk, a hozzá tartozó aloldalra visz át. Az oldal alján egy b probléma vagy észrevétel esetén kereshető email címet találunk.



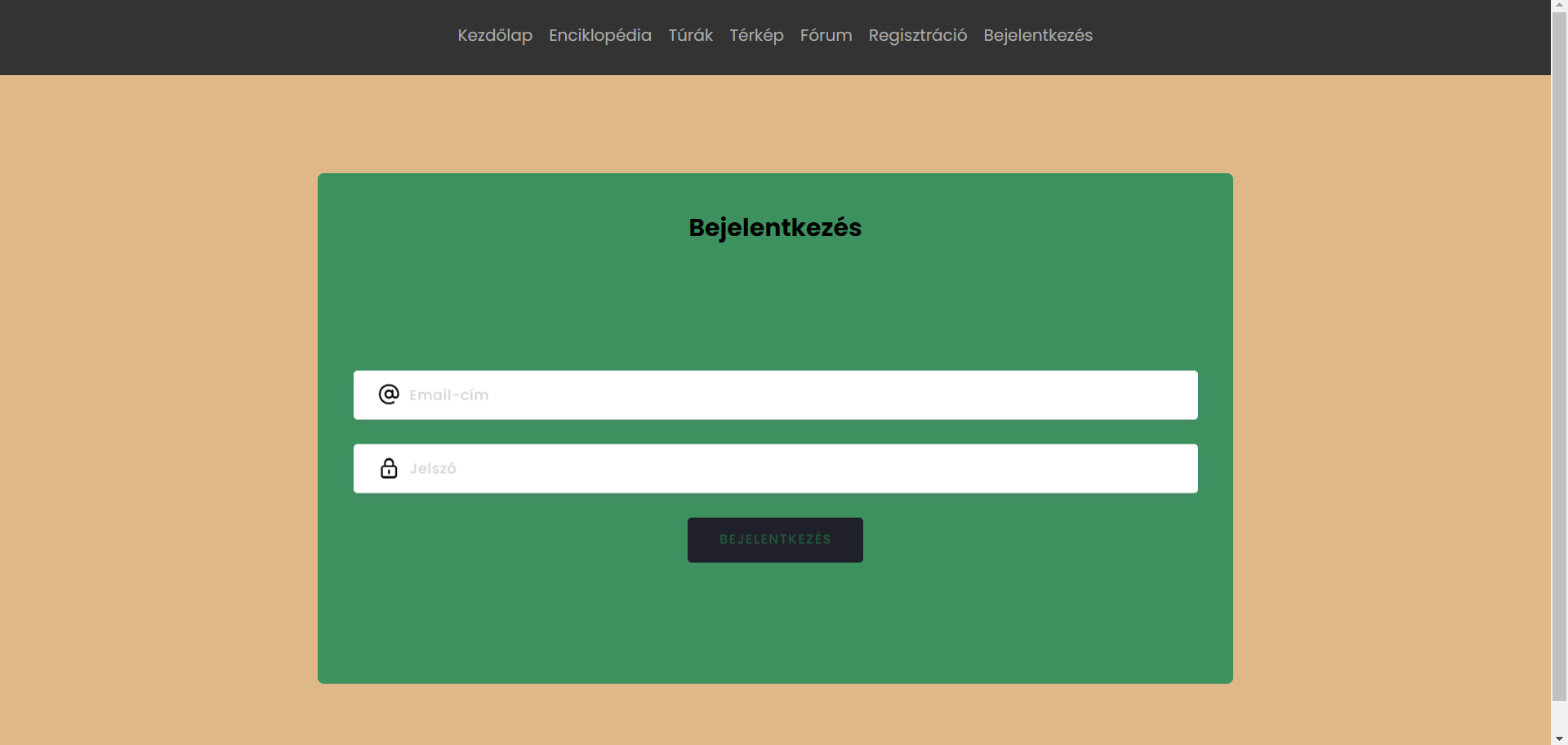
### Regisztráció

A regisztrációs aloldalon tudunk új fiókot létrehozni. Nincs más dolgunk, mint a megfelelő információkat megadni. A felhasználónév és a jelszó mezőre húzva az egeret megtekinthetjük, milyen feltételeknek kell megfelelnie a készülő profilunk adatainak. Ezek után el kell fogadni a gombászat szabályait és etikáját, melyeket a kék szövegre kattintva megtekinthetünk, átolvashatunk és letölthetünk. Ha valamely információt rosszul adtuk meg, arról értesít az oldal.



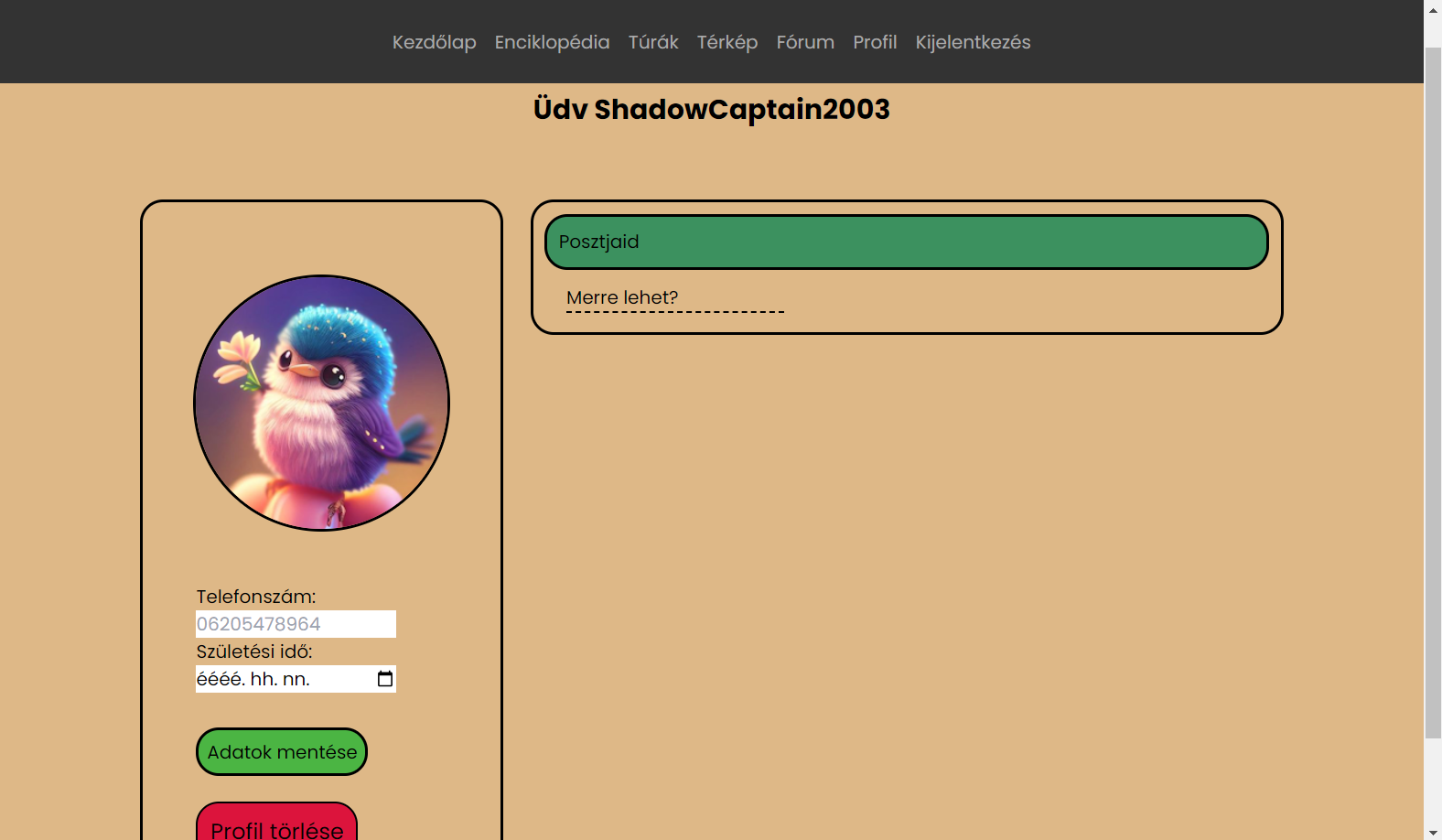
### Bejelentkezés

A bejelentkezés fülre kattintva jelenik meg az aloldal. Bejelentkezni az email címünk és jelszavunk helyes megadásával tudunk. Ha valamely információt rosszul adtuk meg, arról értesít az oldal.



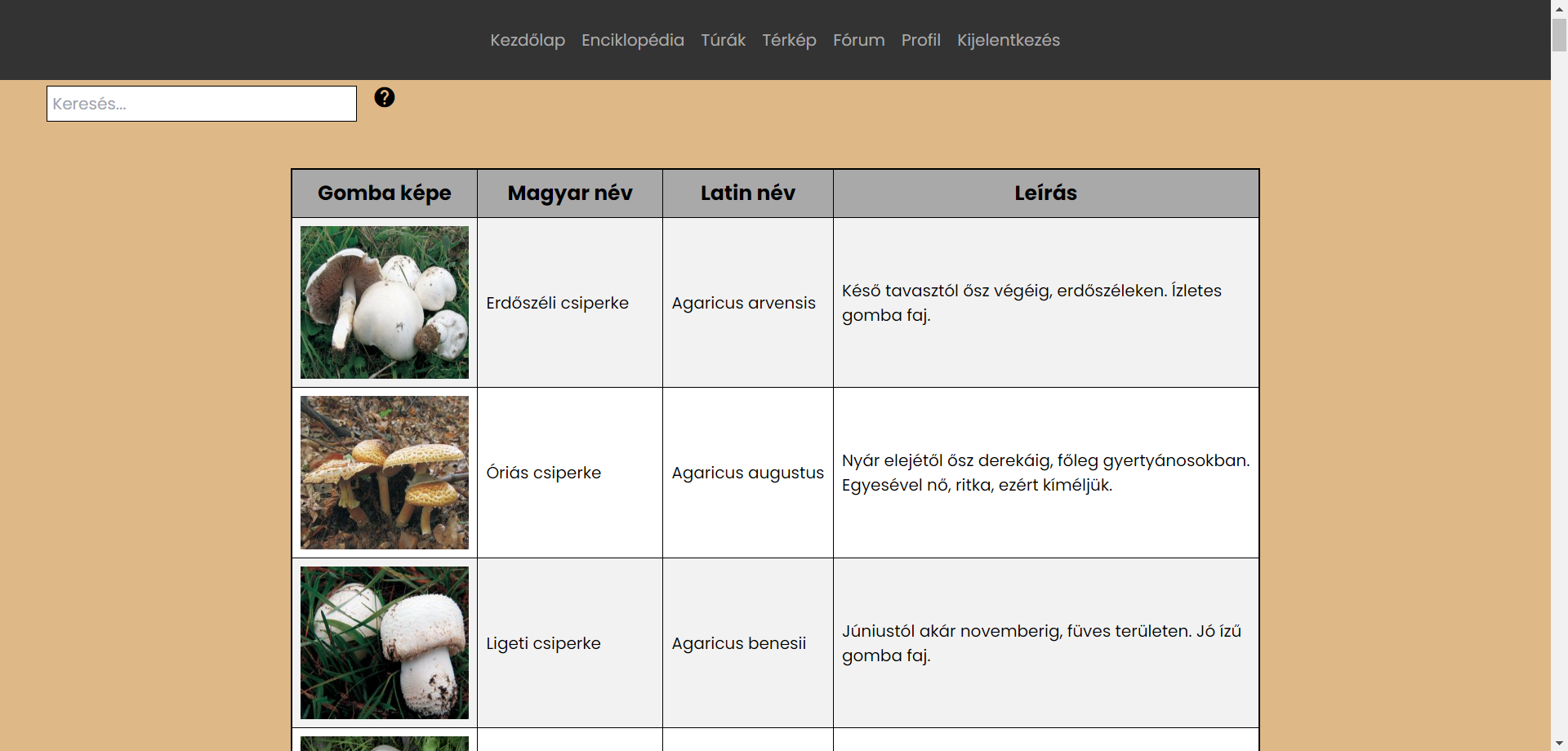
### Profil

A profilon találhatjuk egy-két megadható adatot, illetve itt találjuk az általunk létrehozott posztok címeit felsorolva. A születési évünket csak **egyszer** adhatjuk meg. Rossz megadás esetén kérjük, vegye fel a kapcsolatot a főoldalon található email címen keresztül velünk. Telefonszámunkat bármennyiszer módosíthatjuk, az aktuálisan beállított a beállítás mezőjében látható. A legalsó gombbal törölhetjük a profilunkat.

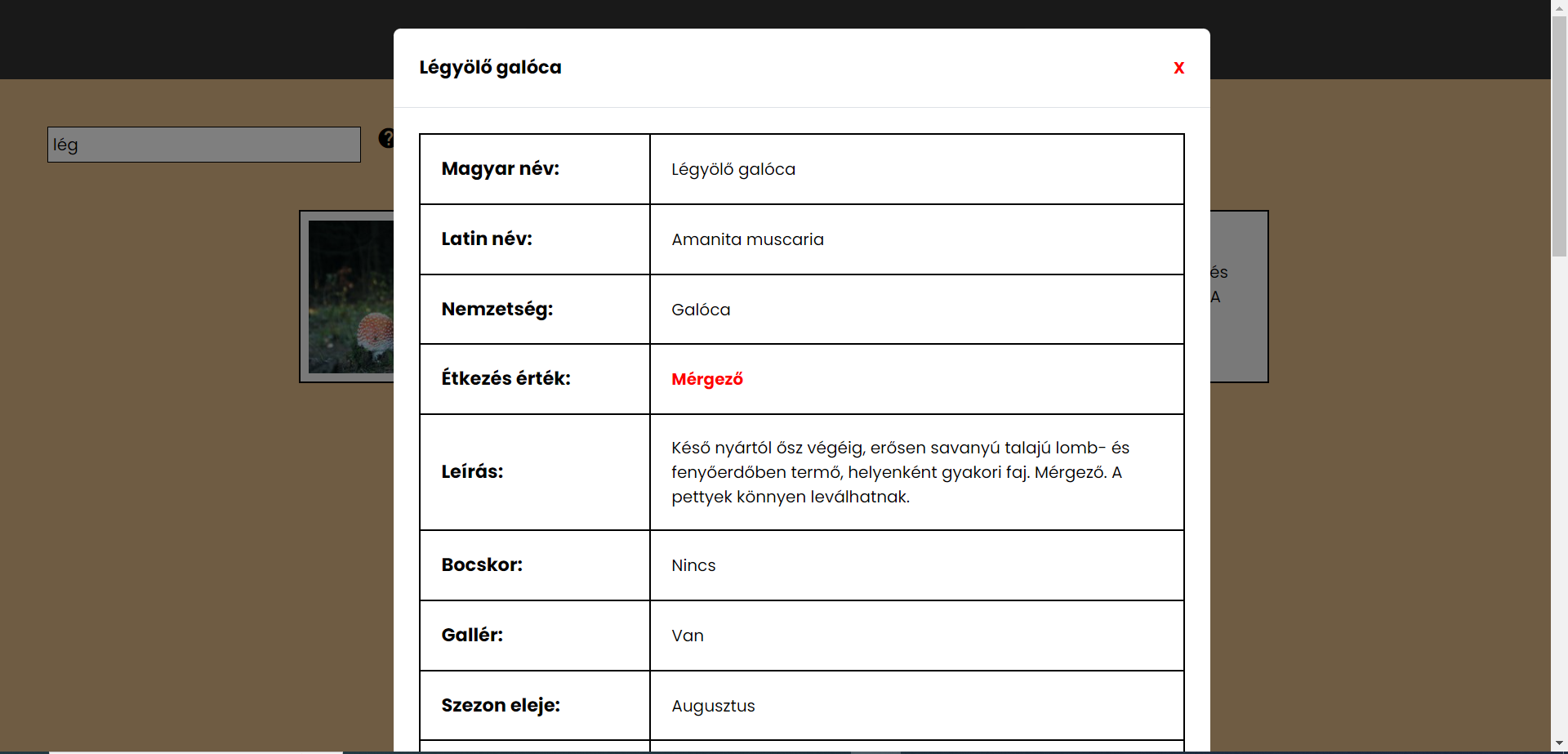


### Enciklopédia

Az enciklopédia aloldalon azokat a gombákat találjuk, amelyekről jelenleg információt tudunk szolgáltatni. Megtudhatjuk például, hogy mi a gomba neve, ehető-e, védett faj-e, stb. (Ezek az információk változhatnak adott környezeti hatásra, időjárásra, akár a helyszínre is.)  
Az aloldalon található keresővel a gombák magyar és latin neve alapján szűrhetünk. Csupán el kell kezdenünk a keresett gomba nevét beírní, és az enciklopédia automatikusan szűri a találatokat.



A gomba képére kattintva részletes információkat tudhatunk meg a kiválasztott gombáról. Külön színnel ki van emelve, ha a gomba mérgező-e, ha csak feltétellel ehető, ha ingyenesen szedhető, ha védett, és ha van eszmei értéke.

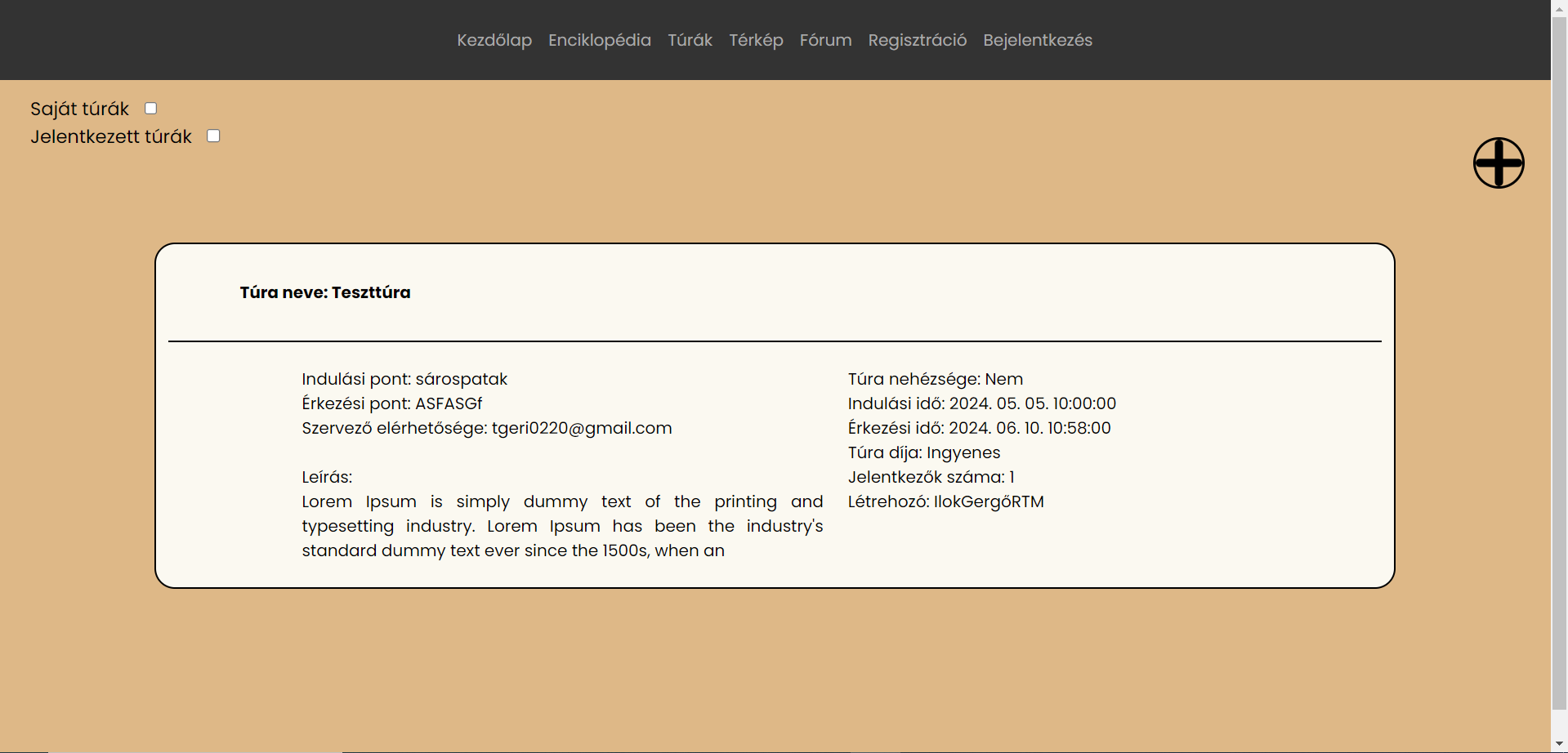


**FONTOS: Bár az eniklopédia adatai segíthetnek a beazonosításban, nem helyettesítik a gombaszakellenőr véleményét. Ha kétes a talált gomba fajában, vagy szeretne biztosra menni, mindenképp keresse fel a legközelebbi gombaszakellenőrt és nézettesse meg vele!**

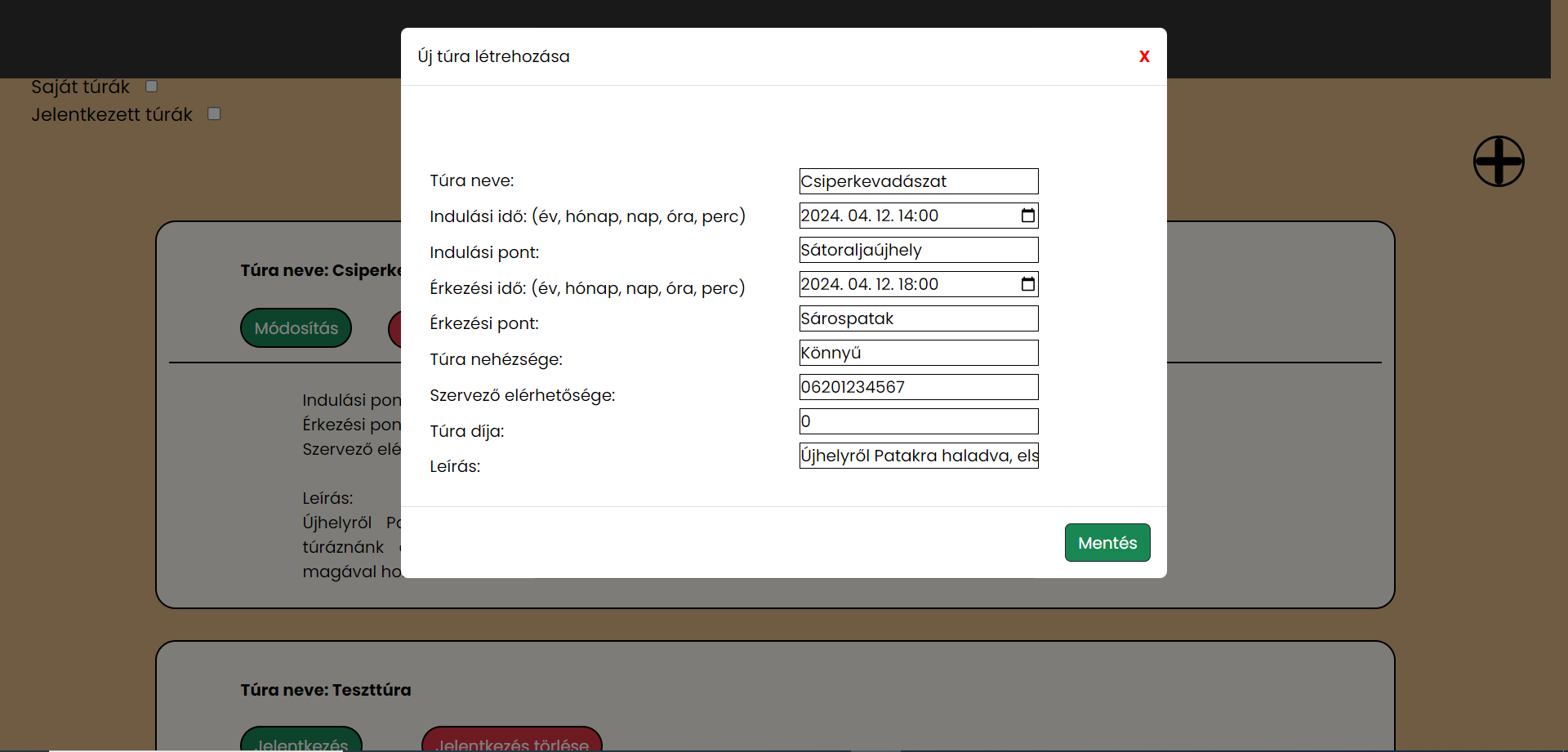
### Túrák

A túra aloldalon láthatjuk azokat a túrákat, amelyekre jelentkezhetünk, és amelyeket mi hoztunk létre. Ahhoz, hogy létrehozhassunk túrát vagy jelentkezhessünk egyre, egy profillal kell, hogy rendelkezzünk, amivel be is kell, hogy jelentkezzünk, és meg kell adnunk a születési időnket. **FONTOS**, hogy a születési időnket csak **EGYSZER** adhatjuk meg, és a túra létrehozásához illetve a túrára való jelentkezéshez be kell, hogy töltsük a 18. életévünket.

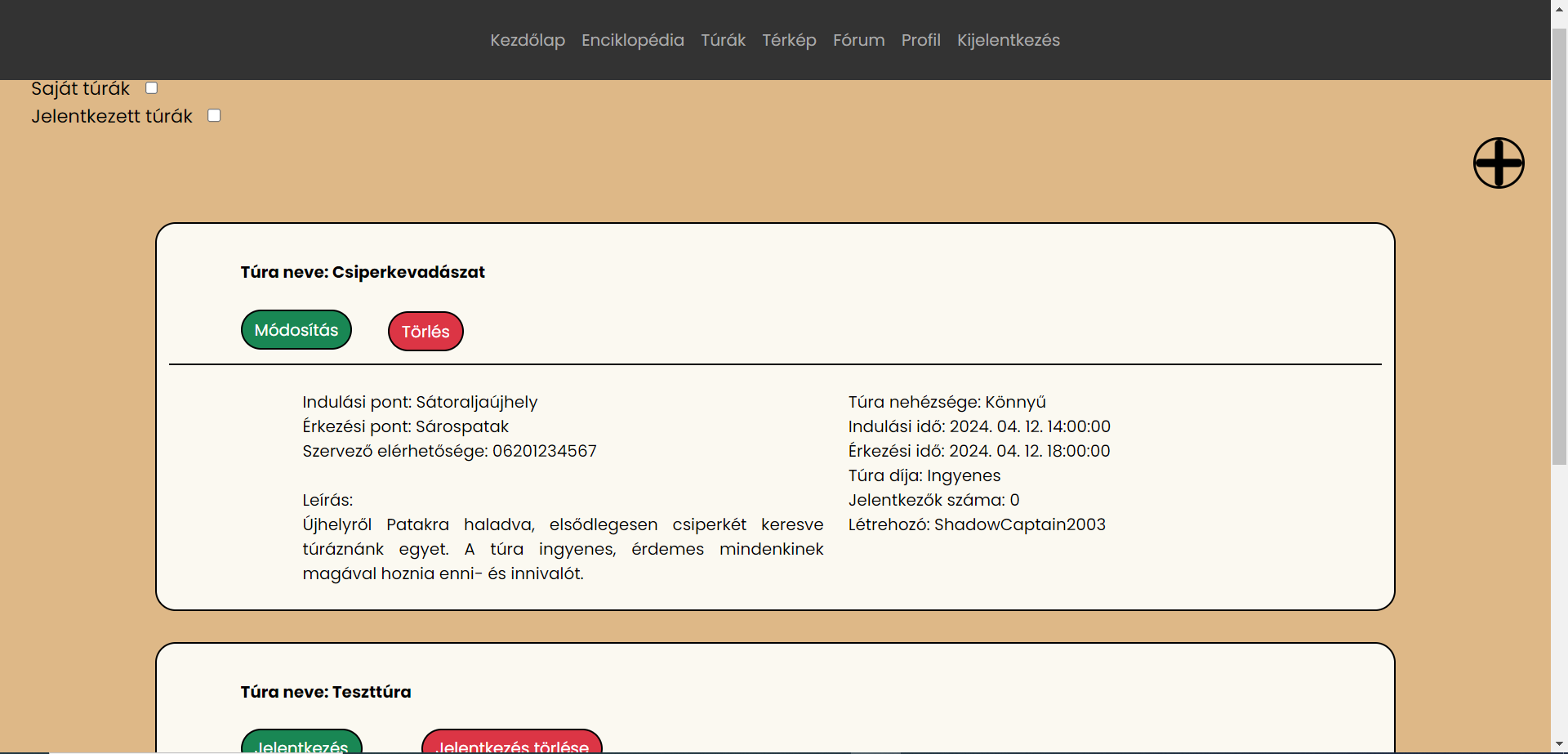
Az esetben, ha nem bejelentkezve látogatjuk meg az aloldalt, a jelentkezés opció nem jelenik meg.



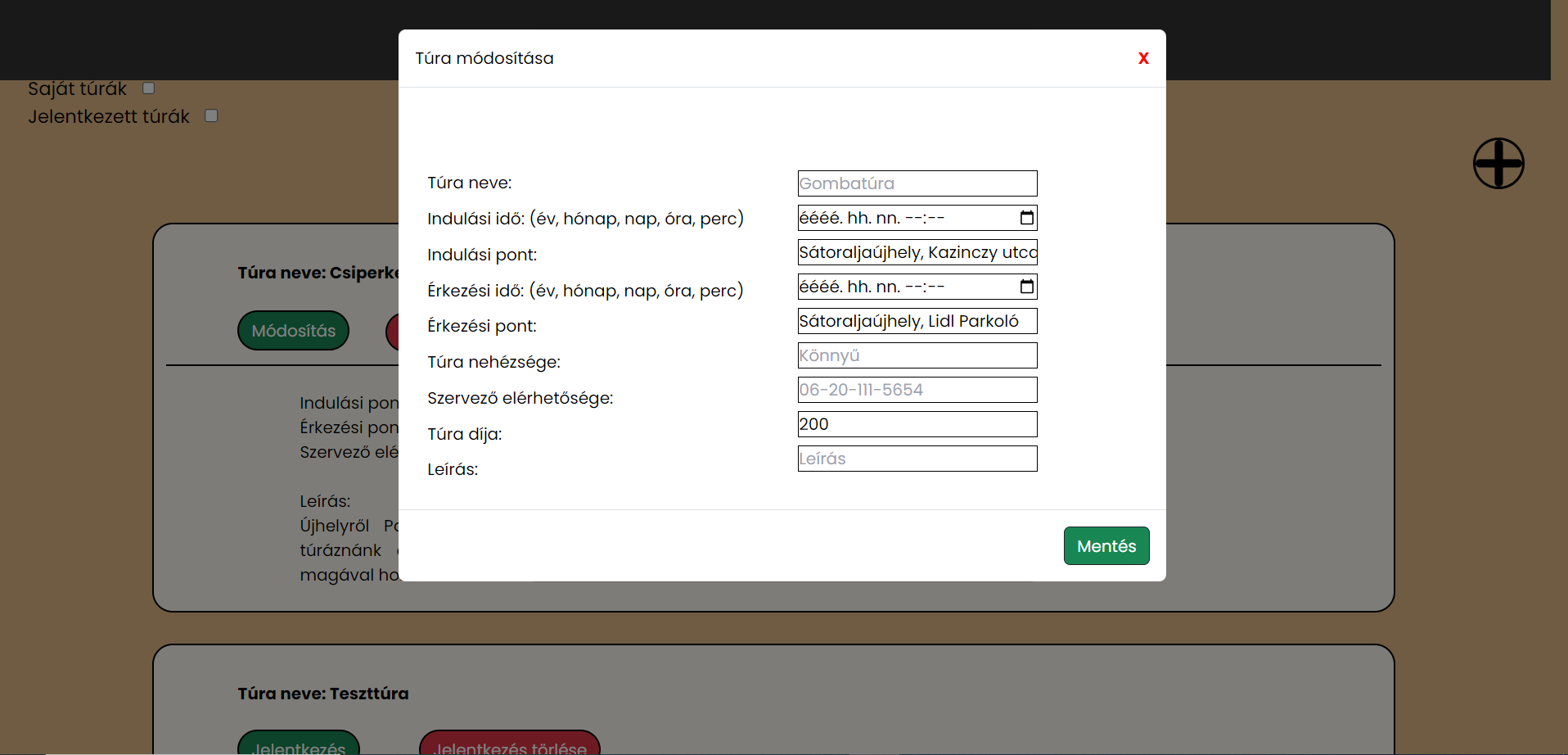
Túrát A jobb felső sarokban lévő plusz gombra nyomva tudunk létrehozni. Egy túrának meg kell adni a nevét, az indulási idejét (az indulási idő nem lehet kisebb, mint a beállításkor lévő aktuális idő), az indulás helyét, az érkezési időt (az érkezés ideje nem lehet kisebb, mint az indulás ideje), az érkezési hely, hogy a létrehozó szerint milyen nehéz a túra. Meg lehet adni egy elérhetőséget (abban az esetben, ha nem adunk meg elérhetőséget, a felhasználó email címe kerül megadásra automatikusan), a túra díjját, ha van (abban az esetben, ha a túra ingyenes, a nullát hagyjuk, mint érték, amely ingyenesként fog megjelenni), és egy leírást, amelyben egyéb, fontosnak tartott információkat oszthatunk meg, például az útvonalat, vagy, hogy mit tervezünk a túra alatt. Ha mindent helyesen töltöttünk ki és nem történt semmi probléma, a Mentés gombra kattintva az oldal visszadob minket a túrákhoz, ahol már láthatjuk is megjelenve a túránkat.

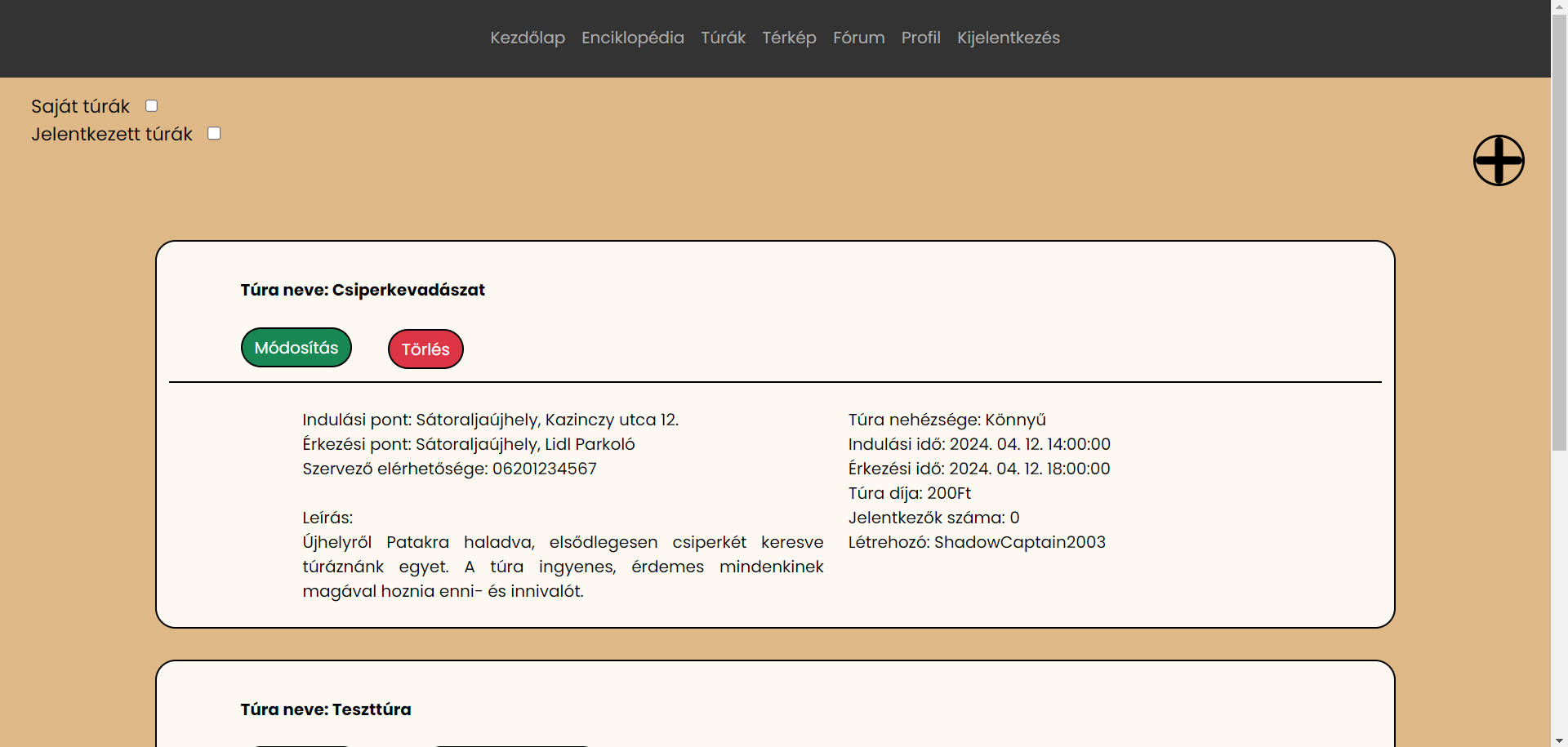


A jelentkezés gombra kattintva hozzáadódunk a túrához, láthatjuk is, hogy a jelentkezők száma növekszik. A mellette található jelentkezés törlésére kattintva pedig visszavonjuk a jelentkezésünk. Bár mindkét gomb megjelenik, a jelentkezés gomb többszöri megnyomása ellenére nem jelentkezünk többször, ugyanígy ha töröljük a jelentkezést, hiába nyomjuk meg többször, csak egyszer vonjuk vissza a jelentkezésünket (ha van egyáltalán).  
A bal felső sarokban található kis gombokkal szűrhetjük a túrákat. A saját túrákra kattintva az általunk létrehozott túrákat láthatjuk, a Jelentkezett túráknál pedig azokat a túrákat, amikre jelentkeztünk.



Az általunk létrehozott túráknál két másik gomb jelenik meg; a túra módosítása, és annak törlése. Módosítás esetén szinte ugyanaz az ablak jelenik meg, mint a létrehozáskor. Itt, csak azokat az adatokat, amiket módosítani szeretnénk írjuk be a megfelelő helyre (vagyis nem szükséges az egészet újra kitölteni), majd a Mentés gombra kattintva a változtatásaink mentésre kerülnek, és ahogy visszadob minket az oldal a túrákhoz, láthatjuk, hogy az adatok frissültek.





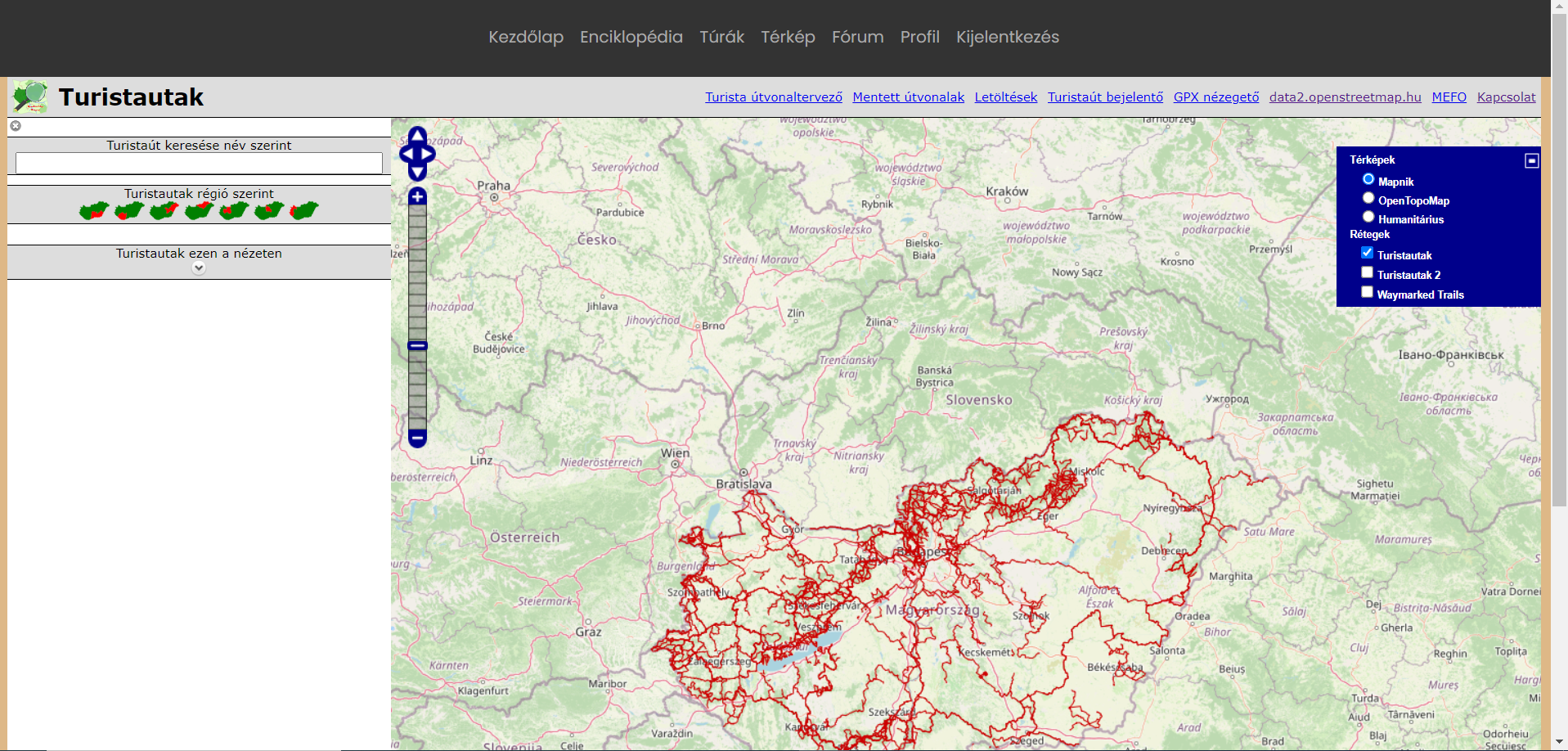
A túra törlésre kattintva töröljük túránkat, mely így nem fog megjelenni se nekünk, se a többi felhasználónak.

**FIGYELEM: Az oldal jelenleg nem küldd sem emailben, sem más formátumban értesítést, ha elmarad a túra, így érdemes időközönként megnézni, hogy a túra, amire jelentkeztünk, az még mindig indul-e.**

### Térkép

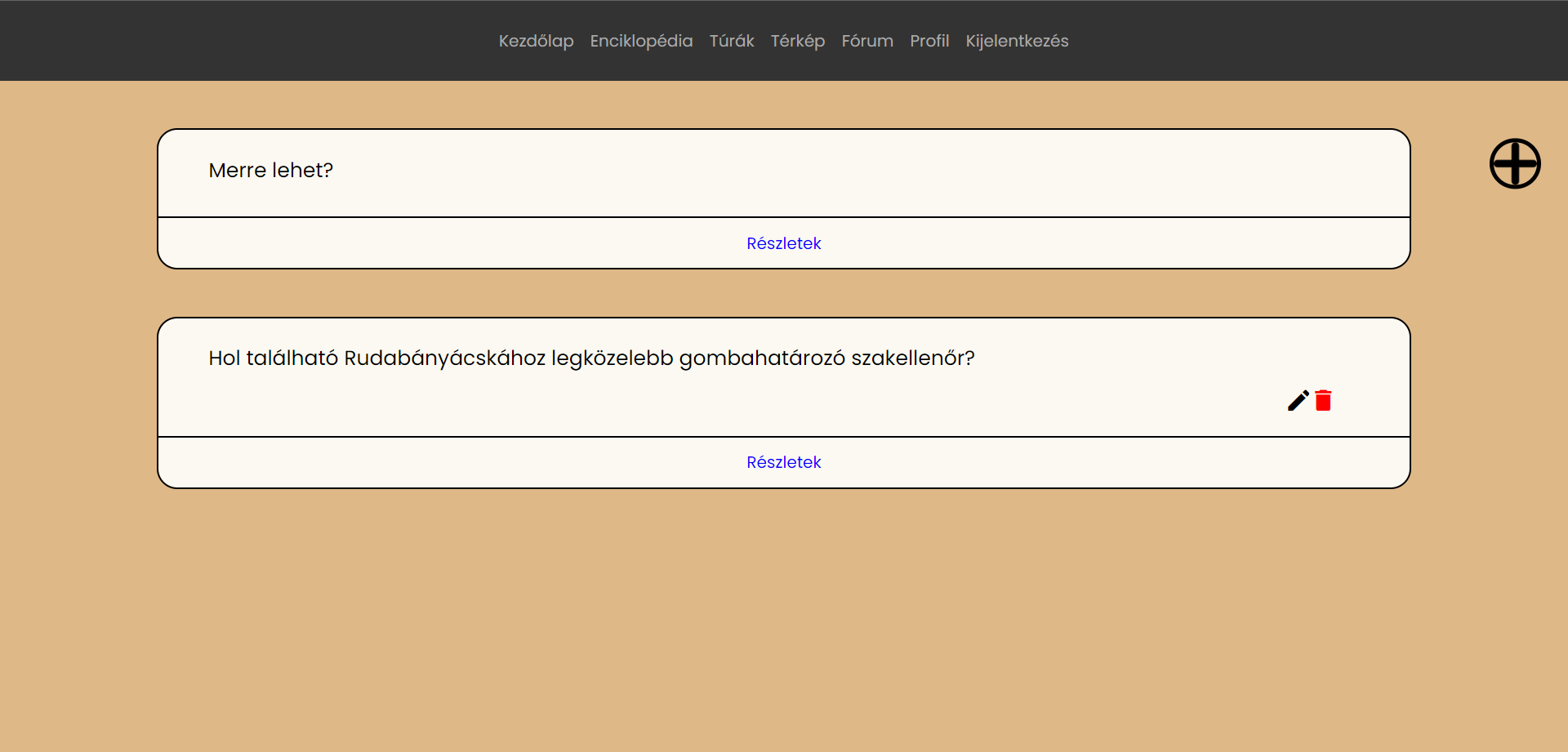
A térkép aloldalon a <https://turistautak.openstreetmap.hu/> oldal található beépítve. A célunk ezzel az volt, hogy a felhasználóknak megkönnyítsük a tájékozódást és a túraszervezést. Lehetőség van régiót kiválasztani, mely után az aloldal bal oldalt kihozza az adott régióban található túrákat. Ezek nevére rákattintva részletes információkat tudhatunk meg róluk, mint például az összetávolságot. A térkép jobb oldalán további szűrési lehetőségeink vannak.   
A térkép felső menüsávján további opcióink vannak, mint például útvonaltervezés, azokat lementeni, visszanézni.

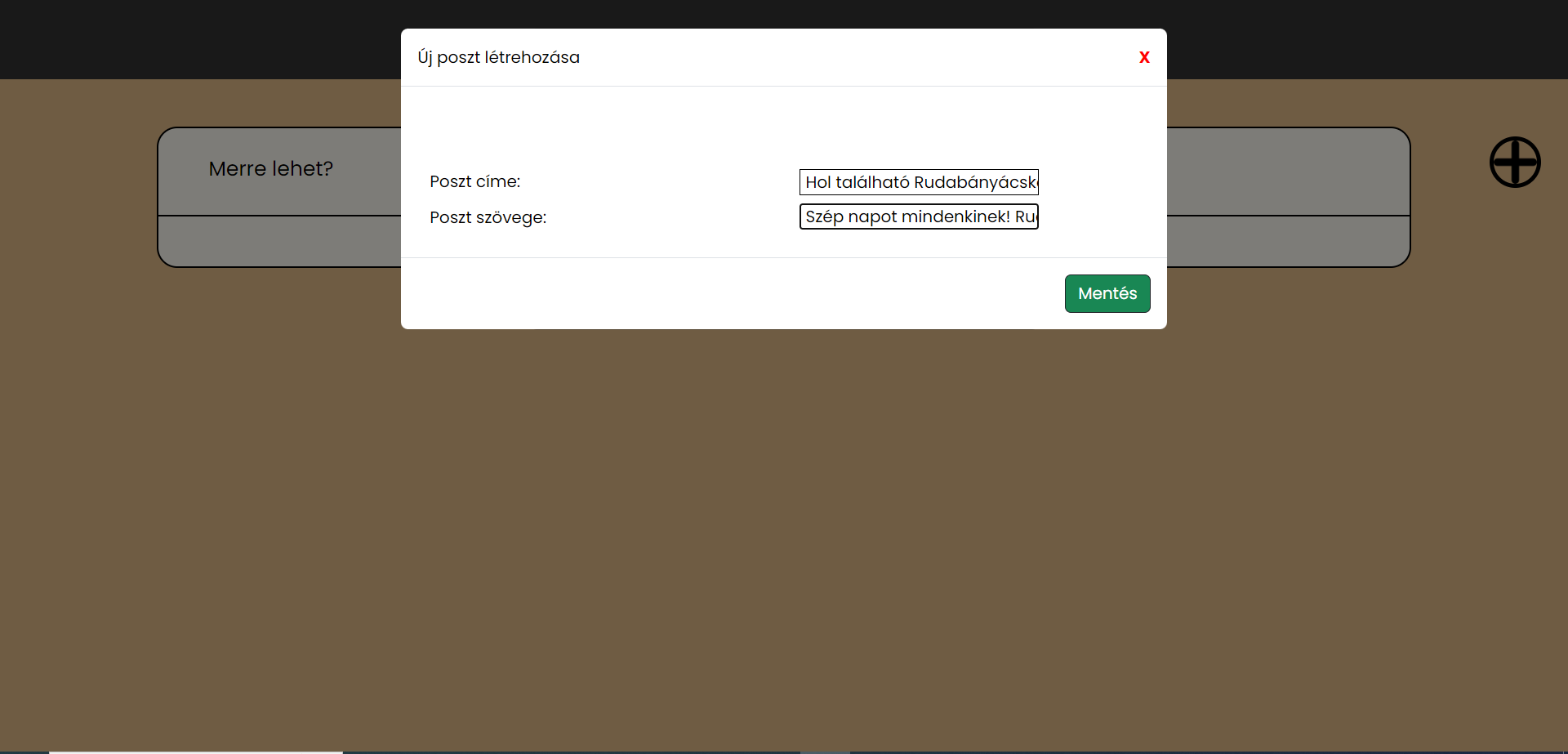
További információt a térképről a [*https://data2.openstreetmap.hu/*](https://data2.openstreetmap.hu/)oldalon található, mely egyenesen elérhető a térkép menüsávján is.



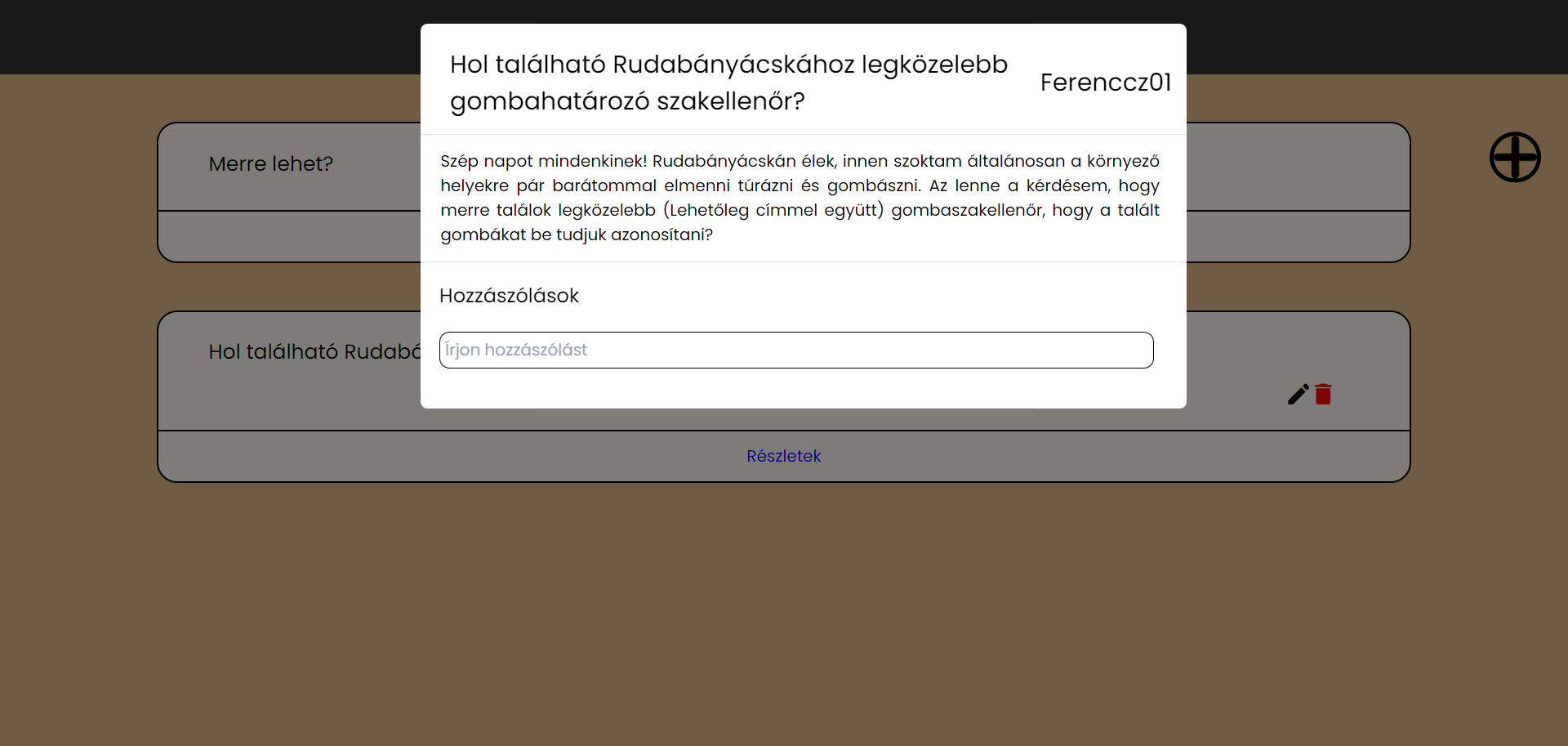
### Fórum

A fórum aloldalon egy közös kommunikációra alkalmas részt találunk, amelyen bármilyen bizonytalanság, kérdés, érdeklődés, információhiány esetén nyugodtan és bátran megkérdezhetjük a többi felhasználót. Az aloldal felhasználói fiók nélkül is megtekinthető, ám, poszt létrehozásához és másposztjai alá hozzászólás írásához felhasználói profil és bejelentkezés szükséges.

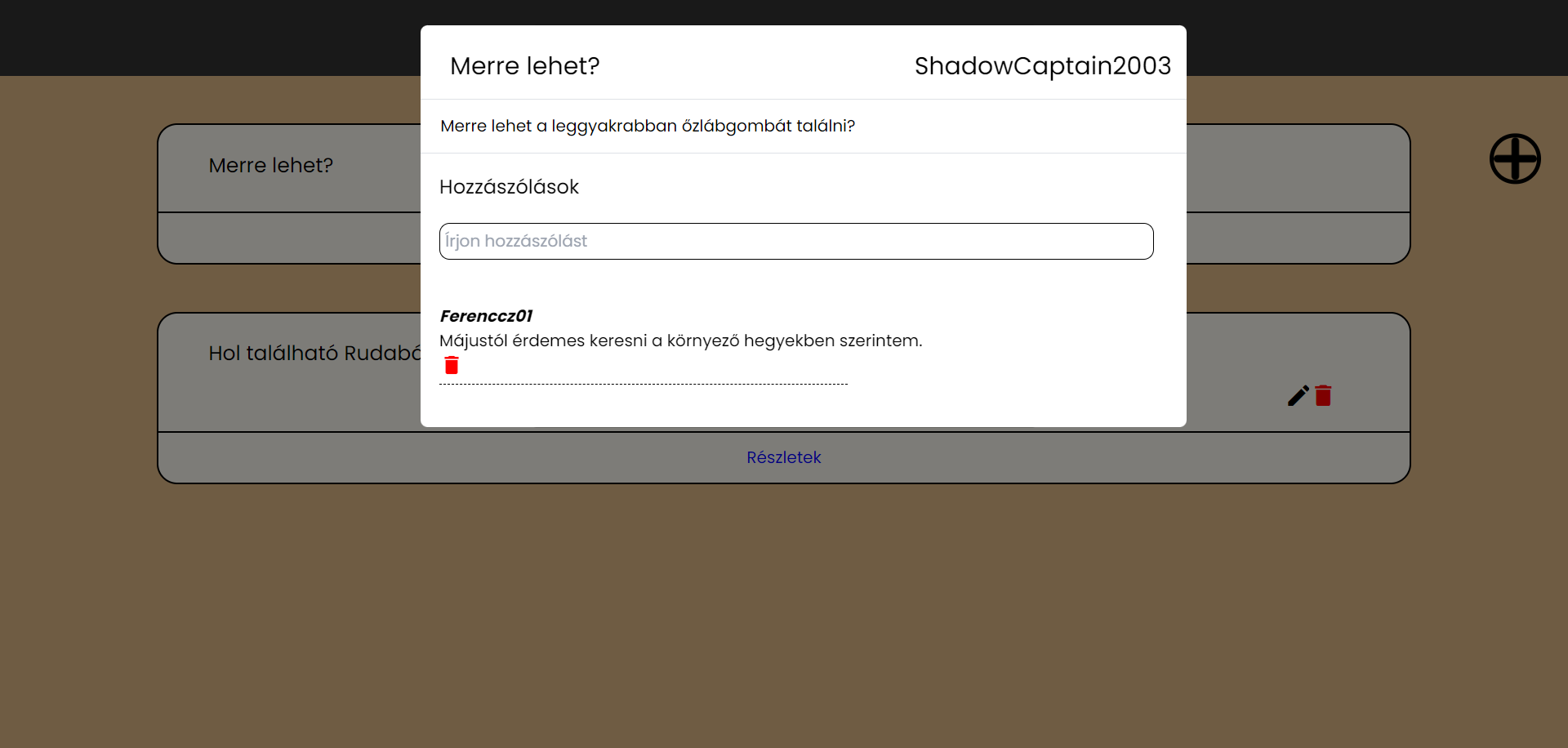


Fórumot – hasonlóan, mint a túráknál – a plusz gomb megnyomásával lehetséges. Ekkor megjelenik egy felugróablak, amelyben meg kell adnunk a poszt címét (ez lehetőleg legyen minél lényegre törőbb és rövidebb) és a poszt tartalmát (ez tartalmazza a bővebb kifejtést). Ha mindkettőt kitöltöttük, a mentés gombra rányomva létre is jön a posztunk.

A posztok tartalmát és a hozzájuk írt hozzászólásokat az adott poszt alatt található Részletek megnyomásával tekinthetjük meg.



Mind mi magunk, mind pedig mások is hozzászólhatnak a posztunkhoz, a posztokhoz. Miután begépeltük az elküldésre szánt hozzászólást, az Enter lenyomásával hozzá is adjuk a válaszunk.



Azokat a kommenteket, amiket mi hoztunk létre, a törlés ikonra kattintva törölhetjük is.

Azoknál a posztoknál, amiket mi hoztunk létre, két ikon található: az első a módosítás, a második a törlés. A módosítás hasonlóképp történik, mint a túráknál; csak azt kell beírni, amit módosítani szeretnénk, itt viszont érdemes előbb kimásolnunk a módosítandó szöveget, és bemásolni, az után átírni azt. A törlésre kattintva pedig töröljük a posztot.

# 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>

Egyéb dokumentumok:

* etika.pdf
* szabályok.pdf

## Fejlesztéshez használt technológiák, szoftverek

### Frontend:

### Frontend:

Fejlesztői környezet: Webstorm

Fejlesztői környezetek közül, azért esett a választásom a Webstormra, mivel nagyon hasonló az IntelliJ-hez amit már jól ismerek és több segítséget nyújtott a frontend szerver fejlesztése során, mint pl. a Visual Studio Code

Szerver-, futtatási környezet: Angular

Források:

* <https://angular.io/>
* <https://ng-bootstrap.github.io/#/home>
* <https://stackoverflow.com/>

### 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.

Források, linkek:

* <https://nodejs.org/en>
* <https://expressjs.com/>
* <https://jwt.io/>
* <https://sequelize.org/>
* <https://www.npmjs.com/package/bcrypt>
* <https://validatejs.org/>
* <https://www.youtube.com/watch?v=YNyGD4rakmc&list=PLkqiWyX-_Lov8qmMOVn4SEQwr9yOjNn3f>
* <https://www.youtube.com/watch?v=-RCnNyD0L-s&t=1771s>
* <https://www.youtube.com/watch?v=0Hu27PoloYw&t=1s>
* <https://www.youtube.com/watch?v=F-sFp_AvHc8&t=1228s>
* <https://www.youtube.com/watch?v=zb8VPxpm_ME>
* <https://www.youtube.com/watch?v=mbsmsi7l3r4>
* <https://www.youtube.com/watch?v=dX_LteE0NFM&t=244s>
* <https://www.youtube.com/watch?v=favjC6EKFgw&t=5s>
* <https://stackoverflow.com/questions/74269225/sequelize-update-via-patch-how-to-process-each-possible-result>
* <https://stackoverflow.com/questions/50932046/how-do-i-make-a-sequelize-query-that-also-returns-data-from-a-table-linked-to-fr>
* <https://stackoverflow.com/questions/48221007/timezone-proplem-in-sequelize?rq=3>
* <https://stackoverflow.com/questions/63087457/different-time-in-mysql-database-timestamp-and-new-datetime-function>

### Adatbázis:

Adatbázis program és verzió: Xampp, 10.4.28-MariaDB

Az adatbázis kezelő program választása a Xamppra esett mivel az elmúlt tanulmányaink alatt jártasságot szereztünk benne és a projekthez szükséges adatok tarolására alkalmas elemek találhatóak benne.

Az ek diagram elkészítése az online.visual-paradigm ingyenes verziójában történt.

# **Frontend**

***Felelős: Tán Gergő***

Frontend szerverünk a Frontend mappában található. A projektünk Angular 17-es verziót használ, ezért továbbfejlesztése ebben ajánlott. Elsősorban szükségünk lesz egy node\_modules mappára és függőségeire, amiket az npm i --force paranccsal tudunk telepíteni. Ezután a terminálba lépve, ng serve vagy ng serve –open (Meg is nyitja a weboldalt) parancsokkal tudjuk futtatni a szervert

**app.config**

Az app.config fájlban állítom be a routes és a HttpClientModule-nak a provider-eit.

A HttpClientModule segítségével küldök http kéréseket a Backend felé a többi komponensben.

**app.routes**

Az app.config fájlban meghívott route-okat itt deklarálom.

Először megadom az útvonalat (ez a ’path’: ’’-el történik) és hozzácsatolom az elérési úton használni kívánt komponenst. A path-eket a Navbar komponensben használtam fel.

{'path': 'enciklopedia', component: EnciklopediaComponent}

felhasznaloAdatObj.ts

Ebben a fájlban interface-ek lettek létrehozva, amikben a felhasználó adatai vannak tárolva különböző eseteknél pl.: regisztráció, bejelentkezés.

export interface signupObj {  
 userID: string  
 felhasznalonev: string;  
 email: string;  
 jelszo: string;  
 jelszoUjra: string  
}

auth.service.ts

Az egyik, ha nem a legfontosabb service fájl a frontend szerveren. Itt van megírva a regisztráció, a bejelentkezés és a kijelentkezés.

Legelőször a constructor lett létrehozva, amiben példányosítottam a HttpClient-t és a Router-t.

Ezt követte a routeok megírása, a backend mappában található routes-ok alapján.

private regisztracioRoute = "http://localhost:3000/regisztracio";  
private bejelentkezesRoute = "http://localhost:3000/bejelentkezes";

Ezek után 3 változó lett létrehozva. A tokenIdjson tárolja a bejelentkezéskor visszakapott tokent, így más komponensekben nem szükséges lekérni, csak a konstruktorban példányosítani ezt a service réteget és könnyedén el lehet érni.

A regHiba és a bejHiba a regisztráció és a bejelentkezés során a backendről visszakapott hiba üzeneteket megjelenítésében segít az adott aloldalon.

error => {  
 if (error.error.type === "Nincsfnev") {  
 this.regHiba = "Nincsfnev"  
 }

else if (error.error.type === "Nemjofnev") {  
 this.regHiba = "Nemjofnev"  
}



ujFelhasznalo metódus végzi a regisztrációs folyamatot.

Http PUT kérés használatával elküldöm a felhasznalo objektumot a fent megírt regisztracioRoute-ra. A subscribe metódussal hozzáférek a backendről érkezett válaszhoz amennyiben a kommunikáció sikeres volt és nem ütközött hibába. Siker esetén a backend küld egy success változót, true értékkel. Ezt vizsgálom egy elágazásban és ha igaz akkor a router.navigate metódussal a főoldalra navigálom a felhasználót.

Hiba esetén a fenti kódban bemutatott módon kezelem a backendről érkező válaszokat.

A bejelentkezes metódus nem sokban tér el az ujFelhasznalo metódustól.

Http POST metódust küldök a megfelelő elérési útra szintén egy felhasznalo objektummal, annyi eltéréssel, hogy az itt használatba vett interface kevesebb adat tárolására alkalmas.

Ezen felül, success: true válasz esetén a localStorage használatával letárolom a visszakapott tokent „access” néven.. Ezután a felhasználó szintén navigálásra kerül a főoldalra.

A kijelentkezes metódus már tartalmaz egy headert is a HttpHeaders használatával.

Ebben a headerben küldöm el a JWT tokent amit bejelentkezésnél tárolok le.

A metódus törzsében ismételten helyet kap az elérési út, a body-ban nem küldök adatot, ezért az üresen marad és végül a header kerül meghívásra.

Amennyiben a metódus sikeresen lefut, a token eltávolításra kerül a localStorage-ból és szintén navigálom a felhasználót a kezdőlapra, ezzel teljesítve a kijelentkezés folyamatát.

Bejelentkezés komponens

bejelentkezes.html

Ezen oldalon található a bejelentkezés panel.



Fentről lefelé haladva a Bejelentkezés felirat alatt található egy kisebb üres hely, itt jelenik meg egy esetleges hiba esetén a hibaüzenet.

Ezt követően 2 input mező található, egy az email címnek, egy a jelszónak. Itt az [ngModel] segítségével a beírt adatot egy a felhasznaloAdatObj.ts-ben létrehozott interface-ben tárolom, majd továbbküldésre kerül a backend számára.

A bejelentkezés gomb megnyomására meghívódik a bejelentkezes metódus.

bejelentkezes.ts

A constructorban példányosítom az AuthService fájlt. Üres string értéket adok a loginObj interface adatainak. A bejelentkezesGomb metódus meghívja az authservice-ben deklarált bejelentkezes metódust.

Enciklopédia komponens

enciklopedia.html

Az oldal tetején található egy keresősáv, mellette egy kérdőjel.

A keresősáv jelen időben frissíti az oldal tartalmát. A gomba magyar illetve latin neve alapján lehet keresni. A keresőbe beírt adatot [ngModel] segítségével tárolom egy változóban, ez alapján végzi a keresést a kereses.ts amire lejjebb bővebben kitérek.

A mellette található kérdőjel ikonra az egeret ráhúzva, egy felugró ablak jelenik meg, aminek tartalma egy kis segítség a gördülékenyebb információhoz jutás érdekében. Ezt az ngbTooltip segítségével valósítottam meg.

Az oldalon lefelé haladva található egy táblázat ami tartalmazz minden gombánk 3 számunkra legfontosabb adatát és a legelső sorban a gombáról készült képet.

A fejlécsor csak akkor jelenik meg ha a keresés után van találat. A táblázatban lévő gombákat \*ngFor segítségével generáltam illetve egy filtert állítottam be a keresőben megadott értékre.

Ha a kersőben megadott értékre nincs találat akkor a táblázat helyén egy szöveg jelenik meg, miszerint „Nincs a keresésnek megfelelő találat”.

A gomba képére kattintva egy lenyíló ablak jelenik meg. Ez NgbModal segítségével készült.

Itt található a gomba összes adata soronként, egy táblázatban felsorolva. Az adatbázisban több helyen lett megadva egy-egy adat szám formában, ezért elágazások használatával adtam vissza a megfelelő szöveges adatot.

enciklopedia.ts

Ezen fájlban a constructorban példányosítottam a HttpClient-t, létrehoztam a szükséges elérési utat, majd a gomba.ts-ben megírt interface adatainak 0 illetve üres string értékeket adtam. Létrehoztam két új változót, egyet gombak néven ami egy Gomba[] és egy searchtext nevő stringet ami a keresés folyamán kapja meg értékét. Példányosítottam az NgbModal-t.

A fájl implementélja az Oninit metódust. Ez a metódus az oldal betöltésekor fut le. Ezen belül kérem le a gombákat és adataikat egy http GET metódussal. A megfelelő route megadása után a subscribe metóduson belül letárolom a fent létrehozott gombak változóba a válaszban érkező tömböt.

kereses.ts

Ez a fájl implementálj a PipeTransform metódust a megfelelő működés érdekében. A metódus fejében megadom azt a tömböt amiből keresni fog és azt az értéket amire keresnie kell. Ha a tömb valamiért üres akkor egy üres tömbbel tér vissza, ha a keresés marad az alpértelmezett üres string akkor visszaadja a tömbböt amiből keresne. A keresett szöveget kisbetűsre állítja.

Ezek után adom meg, hogy mely adatok alapján keressen, jelen esetben a gomba magyar illetve latin neve ez.

transform(gombak: any[], searchText: string): any[] {  
 if(!gombak) return [];  
 if(!searchText) return gombak;  
 searchText = searchText.toLowerCase();  
 return gombak.filter( it => {  
 return it.Magyar\_nev.toLowerCase().includes(searchText) || it.Latin\_nev.toLowerCase().includes(searchText)  
 });  
}

Felhasználó komponens

felhasznalo.html

Az oldal tetején üdvözli a felhasználót, felhasználóneve alapján.

Bal oldalt található egy profilkép, alatta 2 darab beviteli mezővel. Itt lehet megadni a telefonszámot és a születés dátumát, az eddig megszokott [ngModel] alkalmazásával. Ha a felhasználó már beállította telefonszámát, akkor a placeholderben az adatbázisban tárolt szám kerül megjelenítésre, illetve ha születési dátumát is megadta, akkor az input mező többet nem jelenik meg, helyette a letárolt dátum kap helyet. Az alatta található Adatok mentése gombbal lehet véglegesíteni az adatbevitelt. Ez alatt egy profil törlése gomb kapott helyet, amivel a felhasználó törölni tudja profilját.

A jobb oldalon kapott helyet az eddig a felhasználó által létrehozott posztok címe, ezek \*ngFor segíítségével lettek generálva a válaszban visszakapott tömbből.

felhasznalo.ts

A fájl szintén implementálja az Oninit metódust. Ezen metódusban lekérem a tokent a localStorage-ból, ellenőrzöm, hogy van- e token. Ha nincs egy sima return-nel tér vissza. Ha van token, akkor egy token dekódolás következik a jwtDecode és JwtPayload segítségével. A tokenből visszakapott adatok alapján dolgozik a html fájl. Ugyan ebben a metódusban egy GET kérés megy végbe ami visszaadja az adatbázisban letárolt telefonszámot és születési dátumot.

Ezek után már csak a két gombra lett írva egy-egy metódus ami meghívja a megfelelő funkciót.

felhasznalo.service.ts

A constructorban elvégzett példányosítások után megírtam a megfelelő elérési utakat és értéket adtam az interface adatainak.

Az adatMentes metódus fej részében elküldöm a felhasználó adatait amit aa két input mezőben adott meg. Ha a backendről érkező válasz sikeres akkor a lap lefrissíti magát.

A sajatPosztLekeres metódusban az interface segítségével letárolom a visszaérkezett adatokat a posztokról.

A profiltorles egy DELETE metódus ami egy header-t küld benne a tokennel. Ha a backend válasza success:true akkor a localStorage-ből törlöm a tokent, a felhasználót a főoldalra navigálom és frissítem az oldalt, ezzel véglegesítve ezt az eljárást.

Footer Komponens

A footer komponens a főoldalon jelenik meg a lap alján. Ebben csak egy kapcsolat lett elhelyezve.

Fórum komponens

forum.html

Az oldalon blokkokba rendezve találhatók az eddig megírt posztok amik \*ngFor-ral lettek generálva. Ha az adott posztot a felhasználó hozta létre akkor 2 ikon jelenik meg amik a mat-icon segítségével lettek létrehozva. Ezek a módosítás és a törlés funkciók.

Jobb oldalt található egy plusz gomb amire rákattintva, megnyílik egy modal, ebben lehet létrehozni a poszt adatain. Megnyitva ezt 2 input mező jelenik meg, ezek az eddig bemutatott konvenciókat követik. A lenyíló ablak alján egy Mentés gomb található, ami véglegesíti a folyamatot. A poszt létrehozásának feltétele, hogy a felhasználó be legyen jelentkezve.

A poszt szerkesztésére kattintva ugyan az az ablak jelenik meg mint a létrehozásnál, más címmel, és más funkcióval.

A részletek feliratra kattintva megjelenik a poszt, szerzője és a hozzá tartozó hozzászólások is. Hozzászólást az input mezőbe kattintva, a kívánt kommentet megírva és az ENTER gombot megnyomva tudunk létrehozni. Ha a hozzászólást mi írtuk, lehetőségünk van törölni azt. Komment létrehozásra is ugyan az a szabály vonatkozik, mint a poszt létrehozásra.

forum.ts

A fájl első soraiban a eddig megszokott, alapvető működéshez szükséges részek vannak lekódolva. Az onInit metóduson belül a fent létrehozott tömbbe kerül a GET metódusból visszakapott adat.

PosztMegnyitasModal metódus akkor fut le amikor a részletek fülre kattintunk. Itt egy újabb GET metódus fut le, viszont itt már nem csak a psoztot, hanem a hozzá tartozó kommenteket is visszakapom.

forum.service.ts

posztLetrehozas metódusban PUT kéréssel elküldöm a a létrehozni kívánt poszt adatait. Ha sikeres válasz jön backend-ről akkor az oldal frissíti magát. Ellenkező esetben a hiba típusa alapján értesíti a felhasználót az adott hibáról.

A posztModositas ugyan azon az elven alapul mint a létrehozás viszont itt PATCH metódust használtam.

posztTorles metódusban a headerben a token mellett elküldöm a törölni kívánt poszt azonosítóját.. Ha sikeres itt is frissítem az oldalt.

A komment létrehozása és törlése egy az egyben a poszt létrehzás és törlés alapján készült.

# 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: node ./src/app

## 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, csakúgy mint 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 három 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 harmadikat a profilon használjuk, oda jelenítjük meg a felhasználó által létrehozott posztok neveit. 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* | | | |
| /bejelentkezes | post | Megtekinti a bejelentkezési adatokat, összehasonlítja a letároltakkal és a felhasználónak egy tokent készít, amit visszaküldd a Frontendnek. | 200 – Sikeres bejelentkezés!  400 - Hibás felhasználónév vagy jelszó!  500 - Szerver hiba |
| *Kijelentkezés* | | | |
| /kijelentkezes | post | Kijelentkezteti a felhasználót, a tokenjét a feketelistába rakja. | 200 – Sikeres kijelentkezés  500 – Szerver hiba |
| *Túrák* | | | |
| /turak/osszes | get | Visszaadja az összes létrehozott túrát, abban az esetben, ha a felhasználó be van jelentkezve, visszaadja a felhasználó által jelentkezett túrák egyedi azonosítóját is. | 200 – Sikeres lekérdezés  404 - Nincs túra, amire jelentkezni lehetne!  500 – Szerver hiba |
| /turak/jelentkezett | get | Vissszaadja azokat a túrákat, amiket a felhasználó hozott létre. | 200 – Sikeres lekérdezés  404 – Ön még nem hozott létre túrát!  500 – Szerver hiba |
| /turak | get | Visszaadja a felhasználó által jelentkezett túrákat. | 200 – Sikeres lekérdezés  404 – Ön még nem jelentkezett egy túrára sem!  500 – Szerver hiba |
| /turak | put | A felhasználó létrehoz egy túrát az általa megadott adatokkal. | 201 – A túra sikeresen létre lett hozva!  400 – Valamelyik adat hiányzik  403 – Legalább 18 évesnek kell lenni, hogy túrát hozhass létre!  500 – Szerver hiba |
| /turak | post | A felhasználó jelentkezik egy általa kiválasztott túrára. | 200 – Sikeres jelentkezés!  400 – Valamelyik adat hiányzik  403 - Legalább 18 évesnek kell lenni, hogy túrát hozhass létre!  500 – Szerver hiba |
| /turak | patch | A túra létrehozója módosítja a túrát. | 200 – Vagy sikeres a módosítás, vagy nincs változás a letárolt adatok és a módosítások között  404 – Nem található túra az adott túra id-val  500 – Szerver hiba |
| /turak/jelentkezett | delete | Az adott túrára történő jelentkezés törlése. | 200 – A jelentkezés visszavonása sikeres!  404 - Ilyen felhasználói azonosítóval nincs jelentkezés erre a túrára!  500 – Szerver hiba |
| *Enciklopédia* | | | |
| /enciklopedia | get | Vasszaküldi az összes gombát és azok adatait a Frontendnek. | 200 – Sikeres adatküldés  500 – Szerver hiba |
| *Profil* | | | |
| /profil | get | Visszaadja a felhasználó adatait. | 200 – Sikeres adatküldés  500 – Szerver hiba |
| /profil | patch | A felhasználó módosítja a telefonszámát és/vagy születési idejét | 200 - Vagy sikeres a módosítás, vagy nincs változás a letárolt adatok és a módosítások között  404 – Nem található felhasználó az adott id-val  500 – Szerver hiba |
| /profil | delete | A felhasználó törli a profilját, az általa létrehozott túrák ”törlődnek”, ahogy a jelentkezései is | 200 – Sikeres profil törlés  500 – Szerver hiba |
| *Fórum* | | | |
| /forum/osszes | get | Visszaadja az összes fórum adatát. | 200 – Sikeres adatküldés  404 – Nincsenek posztok  500 – Szerver hiba |
| /forum | get | Visszaadja a kiválasztott poszt adatait a hozzászólásokkal együtt. | 200 – Sikeres adatküldés  404 – Nincs poszt ilyen id-val  500 – Szerver hiba |
| /forum | put | A felhasználó posztot hoz létre. | 201 – Sikeres létrehozás  400 - Valamelyik adat hiányzik  500 – Szerver hiba |
| /forum | patch | A poszt létrehozója módosítja a posztot. | 200 – Vagy sikeres a módosítás, vagy nincs változás a letárolt adatok és a módosítások között  404 – Nem található poszt az adott poszt id-val  500 – Szerver hiba |
| /forum | delete | A poszt létrehozója törli a posztot. | 200 – Sikeres törlés  400 – Hiányzik valamelyik adat  404 – Nem található poszt az adott id-val  500 – Szerver hiba |
| /forum/komment | put | A felhasználó hozzászólást hoz létre. | 201 – Hozzászólás sikeresen létrehozva  400 – Hiányzik valamelyik adat  500 – Szerver hiba |
| /forum/komment | delete | A felhasználó törli a hozzászólását. | 200 – Sikeres törlés  400 – Hiányzik valamelyik adat  404 – Nem található hozzászólás az adott id-val  500 – Szerver hiba |

# **Adatbázis**

***Felelős: Erdei Gábor***

MushroomTour adatbázis célja: A tervezés során elsődleges szempont volt, hogy a gombák olyan információit, amik nem szakmai gombászok számára kevésbé egyértelműek vagy ismeretlenek, könnyedén megtalálhassák és erdőben talált gombák esetén képesek legyenek összevetni a látottakat az adatbázisban letárolt, és így az oldalon megjelenített adatokkal.

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

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

**Kódolás**: utf8\_hungarian\_ci

## feketelista

Kijelentkezett felhasználok tokenjeit tároljuk, amik még nem jártak le.

**id (primary key)** int(11) A feketelista elsődleges azonosítói

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

## felhasznalok

A tábla a felhasználók adatait és elérhetőségét tartalmazza, a jelszót kódolva tároljuk így az adatbázis számára ismeretlen az eredeti jelszó, ezen felül tárolva van az első létrehozás és a legutóbbi módosítás dátuma.

A Szuletesi\_ido Telefon\_szam: engedélyezett a null.

**User\_id(Primary key)** varchar(255) A felhasználó azonosítója

**Felhasznalonev** varchar (32) A bejelentkező felhasználó neve

**Jelszo** varchar (32) A felhasználó jelszava kódolva, tárolva

**Email** varchar (60) A felhasználó email címe

**Szuletesi\_ido** date A felhasználó

születési éve

**Telefon\_szam**  varchar (12) A felhasználó

Telefon száma

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

**createdAt** datetime A felhasználónak első bejelentkezésének

az ideje óra:perc:másodperc

**updatedAt** datetime A felhasználónak utolsó belépésénelnek

az ideje óra:perc:másodperc

## gomba

A gomba tábla tartalmazza az összeszedett információinkat (a dokumentáció írásának időpontjában) 96 darab gombáról. 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.

**Gomba\_Id (auto\_increment)**  int(11) Elsődleges kulcs az adatok pontos darabszámának meghatározása érdekében

**Latin\_nev** varchar (50) A külföldi vagy másképpen nevezett gomba könnyebb azonosítás érdekében

**Magyar\_nev** varchar (50) Zemplénben magyar nyelvű gomba neve

**Nemzetseg** varchar (40) A gomba nemzetségének meghatározása okán

**Ethezes\_erteke** tinyint (4) 0-ehető 1-mégező 2-nem ehető 3-védett

**Feltetel** varchar (50) Van-e az étkezéshez valami fontos információ

**Leiras** varchar (255) 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,

**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) Termőtest milyen talajon terem és kedvel, hol keresse azt a felhasználó

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

**Termoretegtarto\_tipusok** varchar (25) Természetben található termőrétegtartó

**Kalap\_forma** varchar (30) Természetben található kalapforma meghatározása

**Kalap\_felszin** varchar(35)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 lehetennek 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,

**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 (35) A készült kép tulajdonosának a neve

**Kep\_neve** varchar (40) Kép neve

## kommentek

A fórum kommentjei tárolására.

**Komment\_id (primary key)** varchar(255) A forum kommentjeinek a

azonosítója

**Kommentelo** varchar(255) A kommentelő azonosítója

**Komment** text Maga a komment ami megjelenik

**Poszt (Foreign key)** varchar(255) A megjelenő post azonosítója

## posztok

A tábla a poszt adatait tartalmazza; a poszt címét, tartalmát, illetve a létrehozóját, illetve az első létrehozás és a legutóbbi módosítás dátumát..

**Poszt\_id (Primary key)** varchar (255) Post Id

**Postolo\_neve (Foreign key)** varchar (255)Felhasználó neve

**Cim** varchar (255) Kép egyedi azonosítója

**Szoveg** varchar (255) A commentek

**createdAt** datetime A felhasználónak első bejelentkezésének

az ideje óra:perc:másodperc

**updatedAt** datetime A felhasználónak utolsó belépélének

az ideje óra:perc:másodperc

## turak

A tábla a tervezett túrákat tartalmazza, amin részt vehetnek a felhasználók. Tároljuk a létrehozó egyedi azonosítóját, a túra nevét, a kezdés időpontját és helyét, a befejezés időpontját és helyét, az útvonal nehézségét, a szervező elérhetőségét, a túra díját, egy leírást amit a felhasználó ad meg a túráról, illetve egy boolean-t, amivel a túra elmaradását jelezzük. Ezen felül a létrehozás és a legutóbbi módosítás dátumát is.

**Tura\_id(Primary key)** varchar(255) A túra azonosítója

**Tura neve** varchar(255) A túra meghirdetett neve

**Letrehozas** varchar (255) A szervező és felelős

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

óra:perc:másodperc:2023-10-01 08:00:0

**Indulas**\_**hely** varchar (255) 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 (255) A túra érkezésének helye

**Utvonal**\_**nehezseg** varchar (255) A túra nehézsége könnyű/közepes/nehéz

**Szervezo**\_**elerhetosege** varchar (255) 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)

**Leiras** text A Tura leírása

**createdAt** date A felhasználónak első bejelentkezésének

Az ideje óra:perc:másodperc

**updatedAt** date A felhasználónak utolsó

bejelentkezésének az ideje óra:perc:másodperc

## turarajelentkezestabla

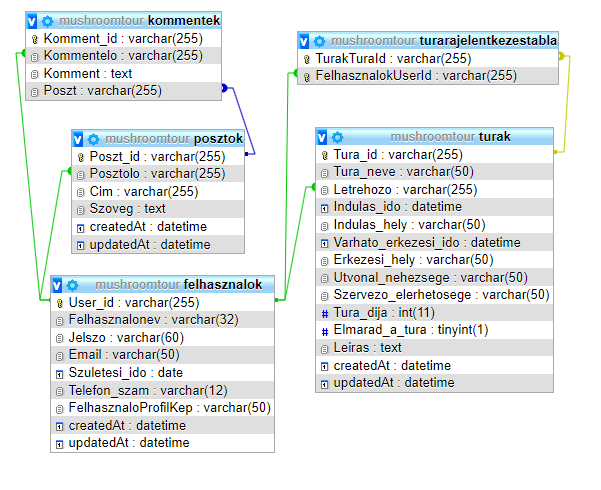
A tábla tartalmazza ki melyik túrára jelentkezett.

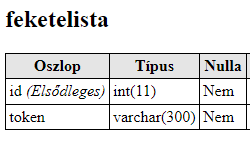
**FelhasznalokUserId (Foreign key)** varchar(255) A túrára jelentkezők azonosítója

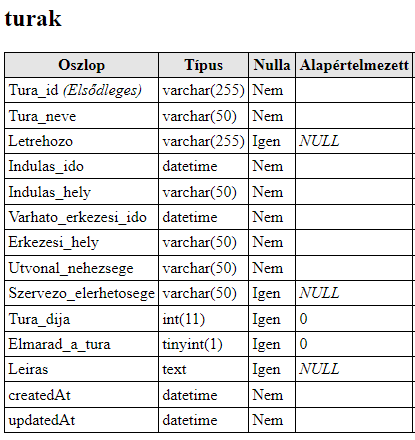
**TurakTuraId (Foreign key)** varchar(255) A túra azonosítója

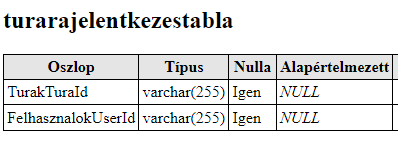
# Adatbázis Képek

Adatbázis kapcsolatok

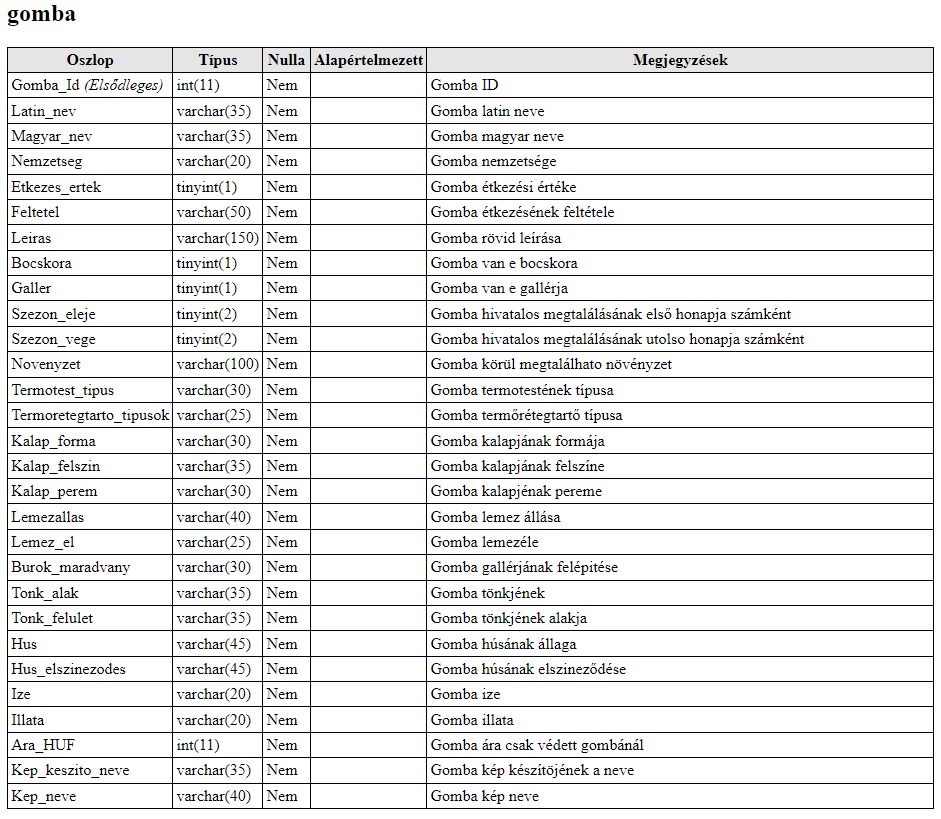


****









# E-k diagram





# Tesztelés

## Tesztelési terv

Név: Erdei Gábor

Projekt neve: MushroomTour

## 1.Bevezetés

* 1. Testelési terv hatókörének célja

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

A teszt végrehajtásáért Erdei Gábor felel.

* 1. Elvárások

Az olvasó ismeri a rendszert

Erdei Gábor Felelős a tesztadatok előállításáért

A program az ebben a dokumentumban meghatározott teszttervek alapján fut.

## 2. Tesztelés

Tesztesetek alábbi, táblázatban található adatok alapján lettek végrehajtva.

## 2.1 Regisztrációs oldal

**Feladat** A bejelentkezés sikeres legyen, felvegye az adatokat az adatbázisba, a jelszót kódólt formában.

**Hiba lehetőség**: Felhasználónév: Kevesebb, mint 10 karaktert ad meg.

Email: @ és pont domain végződés .

Jelszó: Nem felel meg az alábbi kritériumoknak: Nagybetű kisbetű és szám, minimum 8, maximum 32 karakter. A másik lehetőség, hogy 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 | Sikeres belépés |  |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@.com  Jelszó: Laskagomba12  Jelszó ismét: Laskagomba12 | A megadott email cím nem felel meg az elvárásoknak! | A megadott email cím nem felel meg az elvárásoknak! |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@gmail.com  Jelszó: Laskagomba  Jelszó ismét: Laskagomba12 | A megadott jelszavak nem egyeznek egymással! | A megadott jelszavak nem egyeznek egymással! |
| Felhasználó név: Első tesztelő  Email: Teszt.email.00@gmail.com  Jelszó: Laskagomba12  Jelszó ismét: Laskagomba12 | Mivel a felhasználó már létezik:  Már van regisztrált felhasználó ezzel a felhasználónévvel! | Már van regisztrált felhasználó ezzel a felhasználónévvel! |
| Feltételek megnyitása és elfogadása | Megnyílnak a dokumentumok és elfogadáskor végbemegy a regisztráció. | Megnyílnak a dokumentumok és elfogadáskor végbemegy a regisztráció. |

## 2.2 Belépés oldal

**Feladat:** Az oldal a megfelelő email cím és jelszó párossal beengedje a felhasználót és engedi túrát létrehozni

**Hiba** **lehetőség**: A email cím és jelszó páros nem megfelelő vagy nincs a regisztrált felhasználók között

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Email cím: Teszt.email.00@gmail.com  Jelszó: Laskagomba12 | A felhasználót beengedi. | A felhasználót beengedi. |
| Email cím: Teszt.email.22@gmail.com  Jelszó: Laskagomba12 | Nem létező felhasználó, ily | Nem létező felhasználó, mivel még nincs létrehozva |

## 2.3 Túrajelentkezés

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Egy másik felhasználó által létrehozott túrára való jelentkezés. | Sikeres jelentkezés | Sikeres jelentkezés |
| A saját túrám törlése | Sikeres törlés | Sikeres törlés |

## 2.4 Térkép

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Térkép megjelenítése | Sikeres megjelenés | Sikeres megjelenés |

## 2.5 Enciklopédia

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Mind a 96 gombát kiírja fajta szerint abc sorrendben | Megjelenik az összes gomba faj. | Megjelenik az összes gomba faj. |
| kucsmagomba | Az összes kucsmagombát kiírja | Az összes kucsmagombát kiírja |
| Russula | Az összes galambgombát kiírja. | Az összes galambgombát kiírja. |
| Csiga | Nem ír ki semmit mivel csigagombák nincsenek az adatbázisban. | Nem írt ki semmit |
| Képre kattintva megjelennek a gomba adatai | Megjelenek a gomba helyes adatai | Megjelentek a gomba helyes adatai |

## 2.6 Profil oldal

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Profil törlése | Sikeres törlés | Sikeres törlés |
| Bejelentkezés nélkül ne jelenjen meg | Nem jelenik meg | Nem jelenik meg |

## 2.7 Túrák oldal

Előfeltétel: Egy sikeres bejelentkezés, nélküle nem tudjon jelentkezni csak megnézni.

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Megjelennek a túrák | Megjelentek | Megjelentek |
| Új túra létrehozása bejelentkezés nélkül | Nem lehet létrehozni | Nem lehet létrehozni |
| Új túra létrehozása bejelentkezve 18 év alatt | Nem lehet létrehozni | Nem lehet létrehozni |

## 2.8 Fórum oldal

Előfeltétel: Szükséges bejelentkezés, nélküle csak meg tekintés

|  |  |  |
| --- | --- | --- |
| Bemenet | Várható eredmény | Valós eredmény |
| Sikeres megjelenés | Megjelenés | Megjelenés |
| Posztlétrehozás:  Cím: Hol tudok szegfűgombát találni?  Szöveg: Sziasztok, én új vagyok, és azt szeretném, meg tudni hol lehet szegfűszeg gombát találni Széphalom környékén | A poszt létrejön | A poszt létrejön |
| Sajátom törlése | Sikeresen törölve | Sikeresen törölve |

# 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 bővítés.

* Vármegye és országszintű kiterjesztés:

Vármegye és az Alföld gomba világának megjelenítése, illetve a magashelységek gombavilágának feltüntetése.

* Tanulási lehetőségek Gomba kvíz, tanfolyamszervezés:

Tájékoztatás az esetleges tanfolyamokról, amiket az egyesület szervez.

Különböző tanulást segítő játékos feladatot PL:

* + Egy gomba képe alatt két vagy több gombanév és a helyeset kell kiválasztani.
  + Egy gombának a neve és alatt 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 hangsúly, hogy azokat ne gyűjtsék.
* Google naptárral való összehangolás:

Túrára való jelentkezéskor, 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 való felhasználása.

* 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
* A fenti könyvhöz készült fotópályázat
* Zempléni Gombász Egyesület fajlistája 2023