Neumann János Egyetem

GAMF Műszaki és Informatikai Kar

Webprogramozás II. kurzus

Első beadandó feladat dokumentációja

Készítette:

Burka Márk

2022.04.28.

**Tartalomjegyzék**

A program futtatásához szükséges gépi/nyelvi igények ................................................. 3

A program felépítése ....................................................................................................... 3

A public mappa tartalma ................................................................................................. 3

Az app mappa tartalma .................................................................................................... 3

A config mappa tartalma ................................................................................................. 4

A helpers mappa tartalma ................................................................................................ 4

A libraries mappa tartalma ............................................................................................... 4

A controllers mappa tartalma ........................................................................................... 5

A models mappa tartalma ................................................................................................ 7

A views mappa tartalma .................................................................................................. 9

Továbbfejlesztési lehetőségek ........................................................................................ 10

Feladatok teljesítése ........................................................................................................ 11

Képernyő képek .............................................................................................................. 12

**A program futtatásához szükséges gépi/nyelvi igények**

A program futtatásához a XAMPP programban Apach szerver és MySQL szerver indítása szükséges. A program fájlait a ../xampp/htdocs mappa alá kell elhelyezni, valamint a az sql fáljt importálni a MySQL szerverre.

Bongészőből a localhost/webro2bead1 link beírásával az oldal kezdőképernyője betöltődik, innen érhetjük el a program többi részét.

A program elérhető GitHub-ról is a következő linkről: <https://github.com/BurkaMark/webpro2bead1>

**A program felépítése**

Strukturálisan két fő részre oszthatóak a fájlok: a frontend oldal, melynek fájlai a public mappa alatt találhatóak, illetve a beakend oldal, melyek az app mappa alatt helyezkednek el. Továbbá mind a public, mind az app, mind a gyökérkönyvtárban található egy-egy .htaccess fájl is, melyek biztosítják, hogy a felhasználók csak a publikus fájlokhoz férhessenek hozzá. A program több része a felhasználók számára nem elérhetőek, csak a program által átadott adatok.

**A public mappa tartalma**

Ez a mappa tartalmazza az összes olyan fájlt, amelyeket a felhasználó is elérhet. Jelen esetben két fájlt található itt: az index.php és a css mappa alatt található styles.css fájl. A styles.css tartalmazza az összes elérhető oldal stílusát, kinézeti formázását.

**A public/index.php fájl**

Megnyitásakor meghívja a require.php programfájlt, így indítja el a programot.

**Az app mappa tartalma**

Ez a mappa tartalmazza az összes olyan fájlt, mely a program futásához, az adatbázishoz való kapcsolódáshoz, vagy épp az adatok megjelenítéséhez szükséges. Hat mappát (config, controllers, helpers, libraries, models, views) és a require.php programfájlt tartalmazza.

**Az app/require.php fájl**

Meghívja a három alapfájlt (Core.php, Controller.php és Database.php), a konfigurációs fájlt (config.php) és egy segéd programfájlt (session\_helper.php), valamint inicializálja a Core osztályt.

**A config mappa tartalma**

Itt található a configurációs fájl, a config.php.

**Az app/config/config.php fájl**

Globális változóknak ad meg értéket. Az első négy változó (DB\_HOST, DB\_USER, DB\_PASS és DB\_NAME) az adatbázishoz való csatlakozáshoz szükséges adatokat tárolják. Az APPROOT az aplikáció gyökérkönyvtárát rögzíti, az URLROOT az oldalak alap url-jét tárolja, míg a SITENAME a honlap elnevezését.

**A helpers mappa tartalma**

Ebben a mappában található a session\_helper.php fájl, amely egy kiegészítő, segédfájl, melyet a program bármely részéből elérhetünk.

**Az app/helpers/session\_helper.php fájl**

A session\_start() funkció elindít egy munkamenetet. Az isLoggedIn() funkció megvizsgálja, hogy $\_SESSION szuper globális változó user\_id mezője rendelkezik-e értékkel, azaz a felhasználó bejelentkezett-e. Az eredménynek megfelelően igaz, vagy hamis értékkel tér vissza.

**A libraries mappa tartalma**

Ebben a mappában található a három alapprogram, a Controller.php, a Core.php és a Database.php, melyek felelősek a program működésének irányításáért és az adatbázishoz való csatlakozásáért.

**Az app/libraries/Controller.php fájl**

A model() funkció hívja meg az input-ként megadott modelt az app/models mappa alól, majd inicializálja azt.

A view() funkció hívja meg az input-ként megadott nézeti fájlt az app/views mappából, valamint átadja neki az input-ban megadott $data sort.

**Az app/libraries/Core.php fájl**

A konstruktor először eltárolja az url-t egy változóban a getUrl() funkció segítségével.

A getUrl() funkció a $\_GET szuper globális változó url mezőjét vizsgálja, hogy tartlmaz-e értéket. Amennyiben igen, először jobbról levágja a /-t, eltávolítja az url-be nem illő karaktereket, végül szétbontja szekciókra a /-ként. Az így visszakapott értéket átadja a konstruktorban a $url változónak.

Ezt követően a konstruktor ellenőrzi, hogy a $url első mezője tartalmaz-e értéket. Amennyiben igen, megnézi, hogy létezik-e ilyen elenevezésű kontroller programfájl. Ha igen, akkor azt állítja be, mint jelenlegi kontroller, amennyiben nem, úgy marad az alapérték, azaz a Pages, majd törli a $url első elemét.

Ugyanezt megteszi a $url második elemével, csak itt már funkciót keres a már megadott kontroller fájlban. Ha nem talál, akkor az alapértelmezett index értéket tárolja és törli a második elemet is.

Végül ellenőrzik a konstruktor, hogy maradt-e ezeken kívül még eleme a $url-nek. Ha igen, akkor azokat paraméterenként adja át a fent már megadott kontroller funkciónak.

**Az app/libraries/Database.php fájl**

A konstruktor összegyűjti a csatlakozáshoz szükséges adatokat, majd létrehozza a PDO kapcsolatot a program és az adatbázis között.

A query() funkció előkészíti az input-ként megkapott SQL utasítást az adatbázisnak vló átadására.

A bind() funkció összekapcsolja az input-ban megkapot paramétert, az inputban megkapot értékkel.

Az execute() funkció végrehajtja az előkészített SQL parancsot.

A resultSet() funkció összeszedi az össze olyan rekordot az adatbázisból, amely megfelel az előkészített SQL parancsban meghatározott feltételeknek.

A single() funkció megkeresi az első olyan rekordot az adatbázisban, ami megfelel az előkészített SQL parancsban meghatározott feltételeknek.

A rowCount() funkció megszámolja a rekordok számát az adatbázisban, amelyek megfelelnek az előkészített SQL parancsban meghatározott feltételeknek.

**A controllers mappa tartalma**

Ebben a mappában találhatóak a kontroller fájlok. Közös mindegyikben az, hogy mind a Controller.php fájl kibővítése, annak örökösei.

**Az app/contollers/Mnb.php fájl**

Az index() funkció üres értékekkel inicializálja a nézeti mnb fájlt.

A GetExchangeRates() funkció először kiszűri a bemeneti értékeket minden felesleges karaktertől, majd átensek egy validációs folyamaton, melyben egyrészt ellenőrzésre kerül, hogy kitöltésre kerültek-e, a megfelelő formátumban adta meg a felhasználó az értékekekt, illetve a megadott két valuta szerepel-e az adatbázisban (ValidateCurrencies() funkció). Amennyiben valamely fázisában hibára fut a validáció, a felhasználó hibaüzenetet kap.

Amennyiben hibamentesen lefut a validáció, a program kapcsolódik a MNB soap szolgáltatásához. A visszakapott string-et xml-ben tároljuk, majd egys két dimenziós mátrixba konvertáljuk át.

Az így megkapott mátrixból kikeressük a felhasználó által bekért valutákhoz tartozó adatokat és végül kiszámoljuk a két valuta közti átváltási értékeket mindkét irányba és átadjuk a nézeti fájlnak az adatokat.

A ValidateCurriencies() funkció csatlakozik az MNB soap szolgáltatásához. A visszakapott értéket egy sorba rendezi, majd megvizsgálja, hogy az input-ként megadott valuta szerepel-e a sorban. Az eredménynek megfelelően igaz, vagy hamis értékkel tér vissza.

**Az app/controllers/Pages.php fájl**

Az index() funkció betölti az index nézeti fájlt.

**Az app/controllers/Posts.php fájl**

A konstruktor meghívja a Post modelt.

Az index() funkció betölti az összes bejegyzés rekordot az adatbázisból és átadja a post/index nézeti programnak.

A create() funkció először ellenőrzi, hogy a felhasználó bejelentkezett-e már. Csak bejelentkezett felhasználók készíthetnek új bejegyzéseket.

Ezt követően eltávolítja a fölösleges karaktereket a bemeneti mezőkből és ellenőrzi, hogy minden mező kitöltésre került-e.

Amennyiben minden rendben, átadjuk a bejegyzést a Post model addPost funkciójának, az adatbázisban és megjelenítésre kerül a posts/create nézeti programban.

Az update() funkció a create() funkcióhoz hasonlóan ellenőrzi, hogy a felhasználó bejelentkezett-e, de azt is, hogy ugyanaz a felhasználó-e, mint aki létrehozta a módosítandó bejegyzést. Csak a bejegyzést eredetileg létrehozó módosíthatja annak tartalmát.

Ezt követően ugyanúgy átesik az ellenőrzéseken, mint a create() funkcióban, kiegészítve azzal, hogy ellenőrzi, hogy legalább a bejegyzés címe, vagy tartalma változott-e, illetve az input-ban megadott id-nak megfelelő bejegyzést választja ki.

Végül, ha minden rendben van, megkapja a módosított bejegyzést a Post model updatePost funkciója és megjelenítésre kerül a posts/update nézeti programmal.

A delete() funkció szintén ellenőrzi, hogy ugyanaz a felhasználó legyen bejelentkezve, mint aki létrehozta a bejegyzést. Majd megkeresve az input-ban megadott id-nak megfelelő bejegyzést átadja a Post model deletePost funkciójának.

**Az app/controllers/Trails.php fájl**

A konstruktor meghívja a Trail modelt.

Az index() funkció üres értékekkel inicializálja a nézeti trail fájlt.

A getTrailByName() funkció először eltávolítja a felesleges karaktereket a megadott névből, majd az alapján a Trail model getTrailByName() funkciót meghívva, megkeresi a tanösvényt és átadja a trail nézeti programnak a kapott adatokat. Amennyiben valamilyen hiba történt a keresés során, a megfelelő visszatérési értékhez megfelelő hibaüzenetet iratjuk ki.

A getTrailBySettlement() funkció hasonlóan működik a getTrailByName() funkcióhoz, csak a Trail model getTrailBySettlement() funkcióját alkalmazza.

A getTrailByNationalPark() funkció szintén csak abban különbözik a getTrailByName()-től, hogy a Trail model getTrailByNationalPark() funkcióját hívja meg.

**Az app/controllers/Users.php fájl**

A konstruktor meghívja a User modelt.

A register() funkció először kiszűri a felesleges karaktereket a bevitt adatokból. Ezt követően a meghatározott feltételek alapján ellenőrzi, hogy a bevitt értékek megfelelő formátumuak-e. Továbbá azt is ellenőrzi, hogy a megadott felhasználói név és e-mail cím foglalt-e már, regisztrált-e már velük valaki, valamint a jelszó megfelel-e a minimális feltételeknek.

Ha minden hibátlan, a jelszó átesik egy hash generáláson és a User model register funkciónak átadásra kerül minden adat.

Végül meghívásra kerül a users/login nézeti program (a regisztráció nem jár automatikus bejelentkezéssel).

A regisztrációhoz a users/register nézeti program kerül alkalmazásra.

A login() funkció először mentesíti a megadott adatokat a felesleges karakterektől, majd a User model login funkcióját meghívva megpróbálja bejelentkeztetni a felhasználót.

Amennyiben sikerül és a funkció igaz értékkel tér vissza, meghívásra kerül a createUserSession() funkció, mely átadja a $\_SESSION szuper globális változónak a felhasználó adatait és átirányítja a pages/index nézeti programra.

Amennyiben nem sikerül a bejelentkezés, úgy hibaüzenet kiírásával marad a useres/login nézeti programnál.

A logout() funkció törli a $\_SESSION változóvból a felhasználó adatait, majd a useres/login nézeti programot hívja meg.

**A models mappa tartalma**

Itt találhatóak az MVC model model programjai: Post.php, Trail.php, User.php.

**Az app/models/Post.php fájl**

A konstruktor létrehozza a kapcsolatot az adatbázissal úgy, hogy inicializálja a Database osztályt.

A findAllPosts() funkció összegyűjti az összes bejegyzést a bejegyzesek táblázatból, létrehozás dátuma szerint csökkenő sorrendben, majd átadja a kapott adatokat a Posts kontrollernek.

Az addPost() előkészít egy új bejegyzés betöltését a bejegyzesek táblába, összeköti az átadott adatokat az egyes mezőkhöz, majd végrehajtja az SQL utasítást. A művelet sikerességét visszaadja a Posts kontrollernek.

A findPostById() funkció előkészít egy ID alapú keresést a bejegyzesek táblában, összeköti a megadott ID-t az ID mezővel, majd a single() funkció meghívásával végrehajtja az SQL parancsot. A megkapott rekordot átadja a Posts kontrollernek.

Az update() funkció előkészít egy bejegyzés frissítését a bejegyzesek táblában, egy ID alapú azonosítás alapján, majd összekötí a megadott adatokat a megfelelő mezőkkel és végül végrehajtja az SQL utasítást. A frissítés sikerességét átadja a Posts kontrollernek.

A deletPost() funkció előkészít egy törlési parancsot a bejegyzesek táblában, egy ID alapú azonosítás alapján, majd összeköti a megadott ID-t az ID mezővel és végrehajtja az SQL utasítást. A törlés sikerességét átadja a Posts kontrollernek.

**Az app/models/Trail.php fájl**

A konstruktor létrehozza a kapcsolatot az adatbázissal úgy, hogy inicializálja a Database osztályt.

A getTrailByName() funkció előkészít egy keresést az ut táblában név alapján és összekapcsolja a megadott name értékét a név mezővel. A sungle() funkció meghívásával végrehajtja az SQL parancsot és, amennyiben sikeres volt a keresés, összegyűjti a megkapott rekord adatait.

Ezt követően a telepules táblában való keresést készíti elő, összekapcsolja az előző keresés alapján megkapot település ID-ját a táblázat ID mezőjével és a single() funkcióval ismét végre hajtja az SQL parancsot. Ha sikeres volt ez a keresés is, a kapott rekord értékeit eltárolja.

Végül az np táblában való keresést végzi el az előzőekhez hasonlóan, csak itt a második keresésben megkapott nemzeti park ID-t köti össze a tábla ID mezőjével. Ha sikeres volt ez a keresés is, eltárolja a kapot rekord adatait, majd az összegyűlt adatokat átadja a Posts kontrollernek.

A getTrailBySettlement() funkció hasonlóan működik, mint a getTrailByName() funkció, annyi eltéréssel, hogy az első keresés a település neve alapján történik a telepules táblában, a második a megkapott település ID alapján keres az ut táblában, a harmadik a nemzeti park ID alapján az np táblában.

A getTrailByNationalPark() funkció szintén hasonlít a getTrailByName() funkcióhoz, csak ebben az esetben előbb az np táblában keresünk a nemzeti park neve alapján, majd a megkapott nemzeti park ID alapján keresünk a telepules táblában és végül a megkapott település ID alapján keresünk az ut táblában.

**Az app/models/User.php fájl**

A konstruktor létrehozza a kapcsolatot az adatbázissal úgy, hogy inicializálja a Database osztályt.

A register() funkció előkészít egy új felhasználó rögzítését a felhasznalok táblába, majd összeköti az adatokat a táblázat mezőivel, végül végrehajtja az SQL parancsot. A rekord rögzítésének sikerességét átadja a Users kontrollernek.

A login() funkció előkészít egy keresést a felhasznalok táblában username alapján, összekapcsolja a megadott felhasználói nevet a username mezőivel és a single() funkcióval végrehajtja az SQL parancsot. A keresés alapján megkapott rekord jelszó értékét összehasonlítja a megadott hash-elt jelszó értékével. Amennyiben egyezik, átadja a keresés eredményéül kapot rekord adatait a Users kontrollernek. Amennyiben nem, false értéket ad át.

A findUserByUsername() funkció hasonlóan működik a login() funkcióhoz, csak itt nem történik jelszó ellenőrzés. A Users kontrollernek a keresés sikerességét adja át

A findUserByEmail() funkció mindössze annyiban tér el a findUserByUsername() funkciótól, hogy nem username, hanem email alapján történik a keresés.

**A views mappa tartalma**

Ebben található az összes nézeti programfájl. Egyes programok almappákba vannak rendezve a könnyebb kezelhetőség miatt. Ezek az includes (mely tartalmazza a head.php és a navigation.php fájlokat), a posts (mely tartalmazza a create.php, az index.php és a user.php fájlokat) és a users (mely tartalmazza a login.php és a rtegister.php fájlokat) mappák. A views mappában találhatóak az index.php, az mnb.php és a trail.php fájlok.

**Az app/views/includes/head.php fájl**

Ez az a fájl, amelynek tartalam minden nézeti fájl legelején meghívásra kerül. Itt került rögzítésre a head tag, mely tartalmazza többek között a karakter kódolást, a webolfal nevét, a styles.css fájl hivatkozását, valamint az alkalmazott betűtípus hivatkozását. Illetve itt jelenik meg a bejelentkezett felhasználó családi neve, keresztneve és felhasználói neve is.

**Az app/views/includes/navigation.php fájl**

Itt került meghatározásra a navigációs sáv tartalma, illetve az egyes gombokhoz hozzárendelt linkek.

**Az app/views/posts/create.php fájl**

Ebben a fájlban került rögzítésre az új bejegyzés elkészítésének felülete, mely tartalmaz egy input mezőt a bejegyzés címéhez, egy textarea szövegdobozt a bejegyzés tartalmának és egy Létrehozás gombot, mellyel elmenthetjük a bejegyzést.

**Az app/views/posts/index.php fájl**

Ez a fájl határozza meg a bejegyzések főoldalát. Amennyiben bejelentkezett felhasználó tekintí meg az oldalt, úgy egy Új bejegyzés gombot láthat elsőként. Ezt követően jönnek a bejegyzések, egymás után. Minden egyes bejegyzésnél, amennyiben bejelentkezett felhasznló tekinti meg őket és ő a bejegyzés létrehozója, megjelenik egy Módosítás gomb és egy Törlés gomb, melyekkel módosíthatja, vagy törölheti az adott bejegyzést.

Az Új bejegyzés gomb átirányítja a felhasználót a create.php programba, míg a Módosítás gomb az update.php programba.

**Az app/views/posts/update.php fájl**

A create.php programhoz hasonló elrendezésű, mindössze az alsó gomb különbözik, valamint a korábbi adatok betöltésre kerültek a mezőkbe.

**Az app/views/users/login.php fájl**

A bejelentkezési felület programfájlja. Két input mezőből és egy Belépés gombből áll, valamint egy regisztrációs link szerepel a felület alján, mely átirányítja a felhasználót a register.php programba.

**Az app/views/useres/register.php fájl**

A regisztráció nézeti fájlja, mely hat input mezőből és egy Regisztrálás gombból áll.

**Az app/views/index.php fájl**

Az applikáció főoldala és egyben a bemutatkozásé. Három fő szekcióból áll, melyek mindaddig rejtve vannak, amíg a felhasználó rá nem kattint az egyes gombokra(Rólunk, Rövid történetünk, Kapcsolat).

A szövegek elrejtéséért és megmutatásáért a hideUnhideText() JavaScript funkció felel, mely a megadott ID alapján ellenőrzi, hogy az adott szöveg jelenleg rejtett-e, vagy sem és láthatóvá, vagy rejtetté állítja ennek megfelelően. Alapértelmezetten minden szöveg rejte van.

**Az app/views/mnb.php fájl**

A Magyar Nemzeti Bank-tól lerkérdezett adatok megejelenítésért felelős nézeti fájl. A felső részében két input mezőben adható meg, hogy mely devizákra lennénk kíváncsiak, majd az Átváltás gomb megnyomásával megjelenítésre kerülnek a kapott értékéek.

A kapott értékeket két táblázatban jelenítjuk meg, mindkét átváltási irányt egy-egy táblázatban.

**Az app/views/trail.php fájl**

A tanösvények nézeti programfájlja. Három input mező (Tanösvény neve, A település neve, A nemzeti park neve) és a hozzájuk tartozó Keresés gomb jelenik meg. A keresés eredménye egy táblázatba kerül feltöltésre.

**Továbbfejlesztési lehetőségek**

A tanösvények oldal kaphatna egy térkép felületet, melyen jelölésre kerülnének az egyes ösvények. Google térkép potenciálisan használható.

A bejegyzések mellett egy új oldalon események rögzítésére is lenne lehetőség egy naptárban, így a felhasználók szervezhetnének közös kirándulásokat, illetve az admin tudna fontos eseményeket, rendezvényeket is megjelölni.

Közösségi oldalakhoz való csatlakozás.

Képek megosztásának lehetősége (a bejegyzések jelenleg csak szöveges formában működnek).

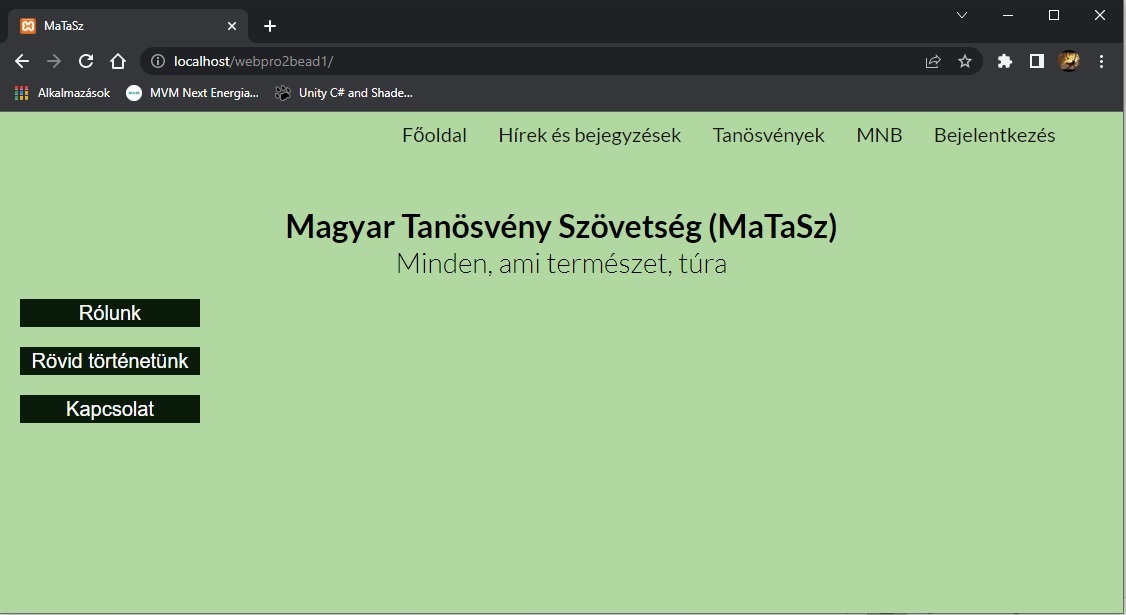
**Feladatok teljesítése**

Az alábbiakban a kiadott feladatok teljesítése szerepel.

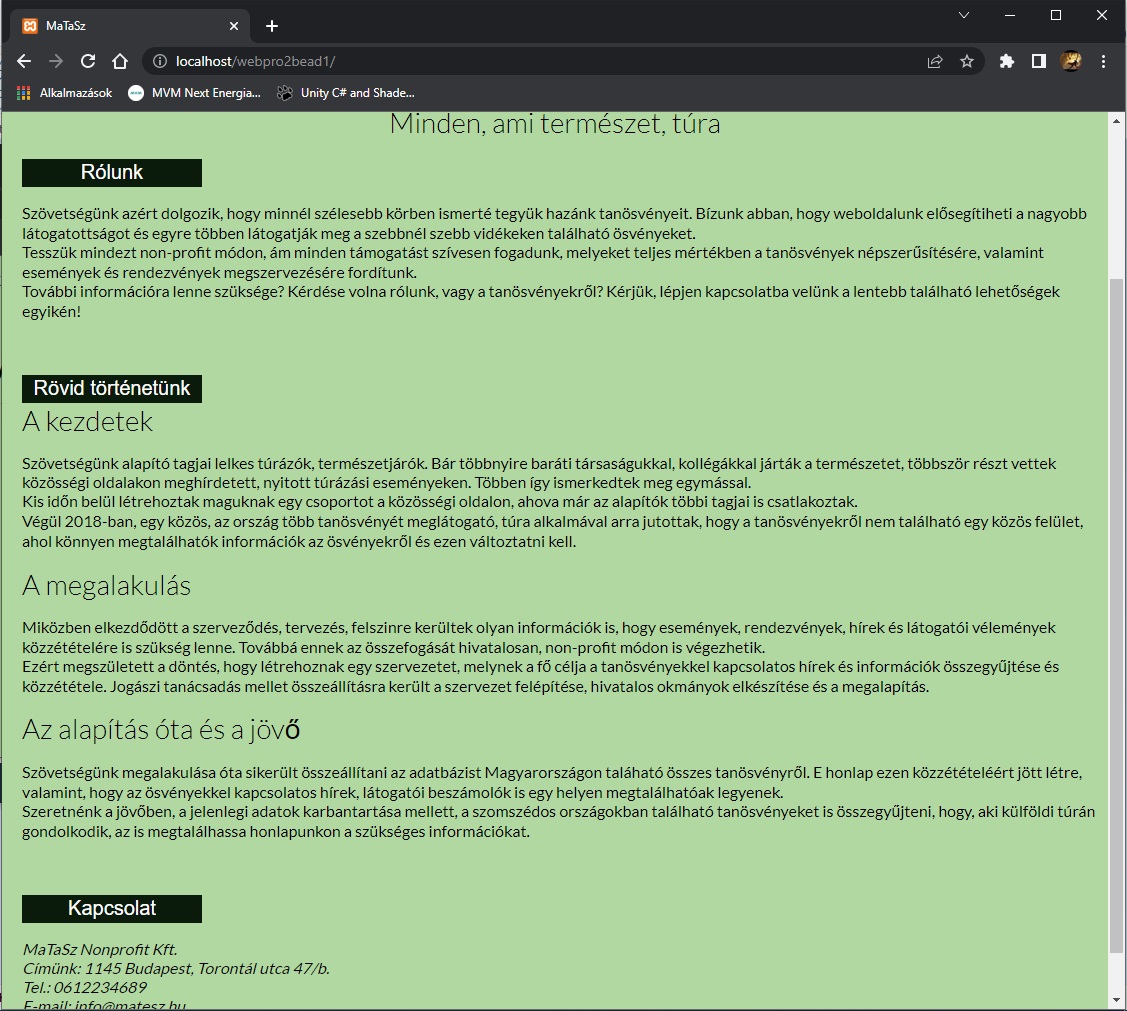
1. Reszponzív bemutatkozó oldal: az app/controllers/Pages.php kontrollerrel meghívott app/views/index.php került megvalósításra. A kihelyezett gombok megnyomására előhozható, vagy elrejthető a bemutatkozó szöveg egyes részei.
2. Bejelentkezés, regisztráció és a bejelentkezett felhasználó kijelzése: az app/controllers/Users.php, az app/models/User.php és az app/views/users/login.php, valamint az app/views/users/register.php programokkal történik a regisztráció és a ki-/bejelentkezés. Az app/helpers/session\_helper.php program ellenőrzi, hogy bejelentkezett-e a felhasználó, majd az app/views/includes/head.php programmal minden oldal tetején megjelenítjük a bejelentkezett felhasználó család- és utónevét, valamint a felhasználói nevét. A 3. feladat d) pontjának megfelelően a regisztráció nem jár automatikus bejelentkezéssel, helyette átirányításra kerül a bejelentkezési felületre.
3. Híroldal: az app/controllers/Posts.php, az app/models/Post.php, az app/views/posts/create.php, modify.php és index.php programokkal került megvalósításra.
4. Objektum orientált PHP: minden kontroller fájl az app/libraries/Controller.php programban definiált Controller osztályból származik.
5. MVC model alkalmazása: az app mappában található a controllers, a models és a views mappa, melyben a megfelelő szerepet kiszolgáló programok helyezkednek el a modelnek megfelelően.
6. SOAP kiszolgáló: az app/controllers/Trails.php, az app/models/Trail.php és az app/views/trail.php programok felelnek, hogy a tanösvényekről szóló adatbázishoz kapcsolódjön az aplikáció, majd a felhasználó által megadott keresési feltétel alapján megjelenítsék az adatokat.
7. MNB SOAP kiszolgáló: az app/controllers/Mnb.php és az app/views/mnb.php programok felelnek azért, hogy a felhasznál által megadott keresési feltételeknek megfelelő valuta átváltási adatokat kérjen az MNB adatbázisából.
8. Internetes tárhelyre való feltöltés: a nethely.hu került választásra, mint platform, ám sajnos az ingyenses tárhely nem ad lehetőséget ennyire összetett program futtatására. A hibaüzenet: AH00124: Request exceeded the limit of 10 internal redirects due to probable configuration error. Use 'LimitInternalRecursion' to increase the limit if necessary. Use 'LogLevel debug' to get a backtrace.
9. GitHub használata: a program fejlesztése során folyamatosan feltöltésre került a pillanatnyi állapota a programnak. A GitHub eléréséhez a link megtalálható a 3. oldalon.
10. Dokumentáció készítése: jelen dokumentáció tesz eleget e feladatnak.

**Képernyő képek**

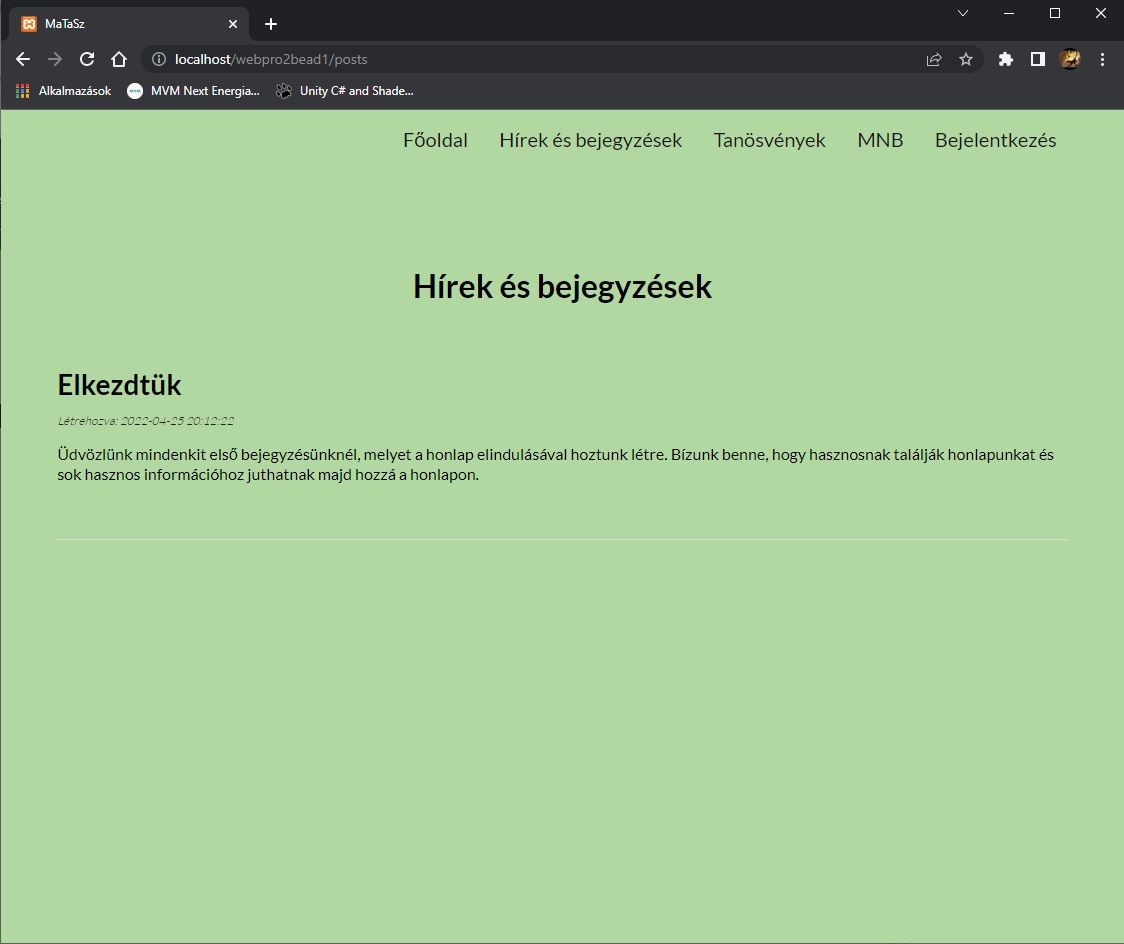
Kezdőoldal, szövegek rejtve



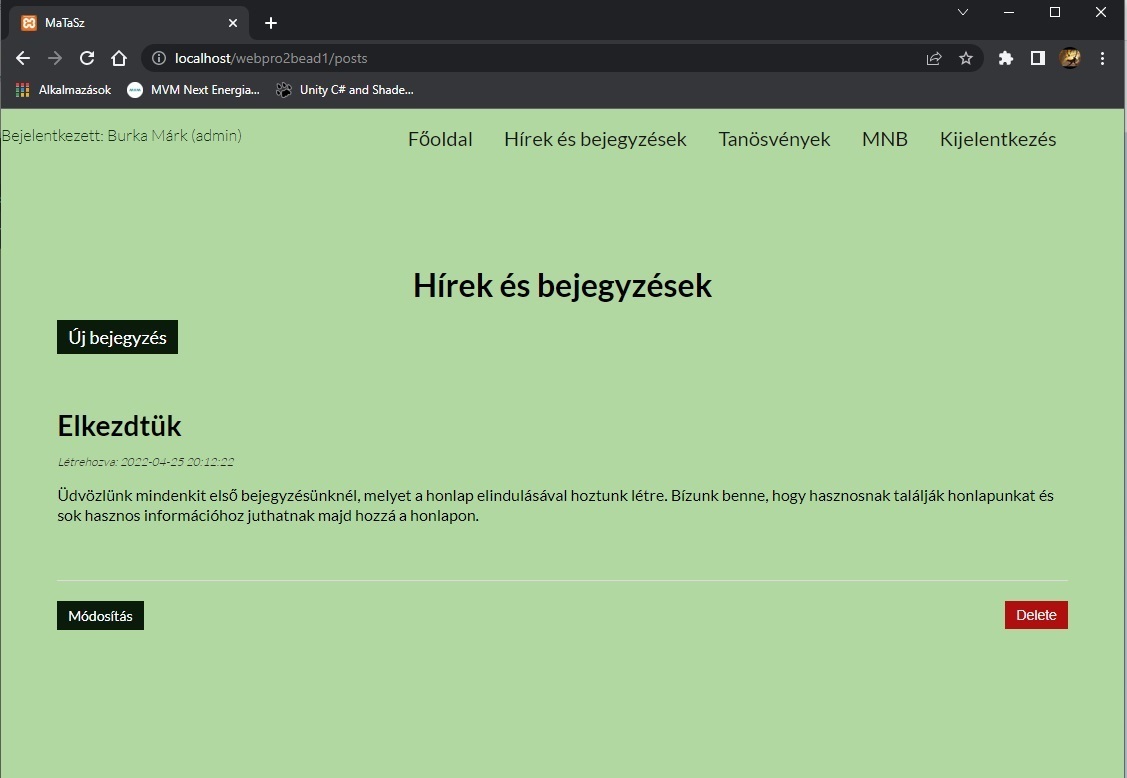
Kezdőoldal, szövegek láthatóak



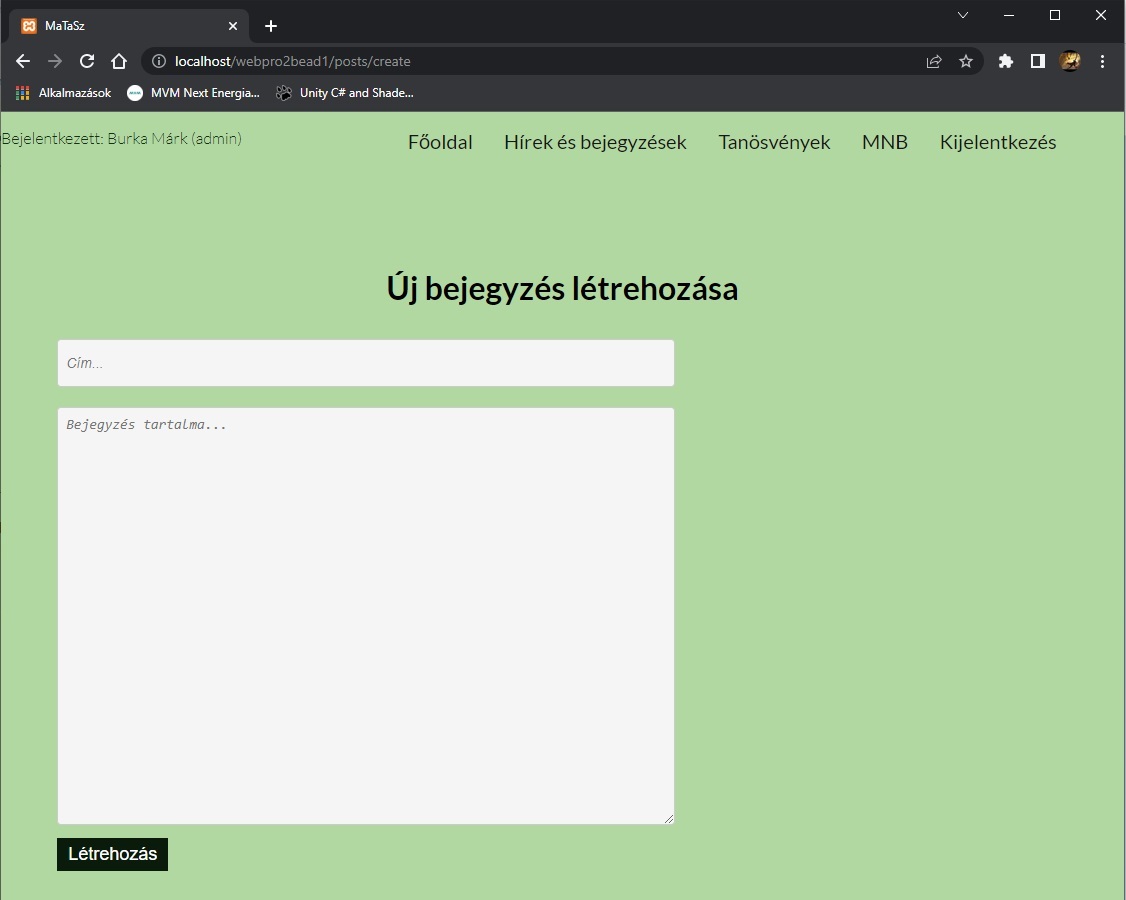
Bejegyzések, bejelentkezés nélkül



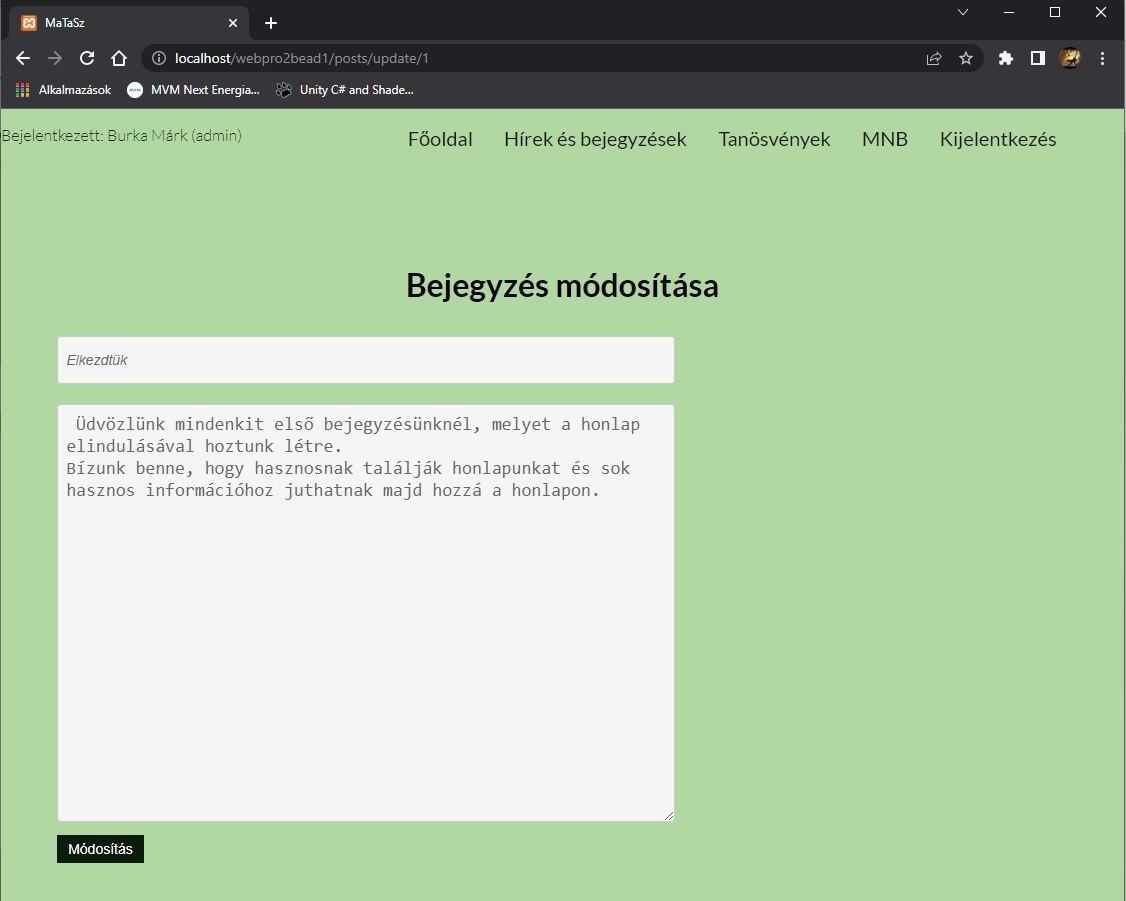
Bejegyzések, bejelentkezve



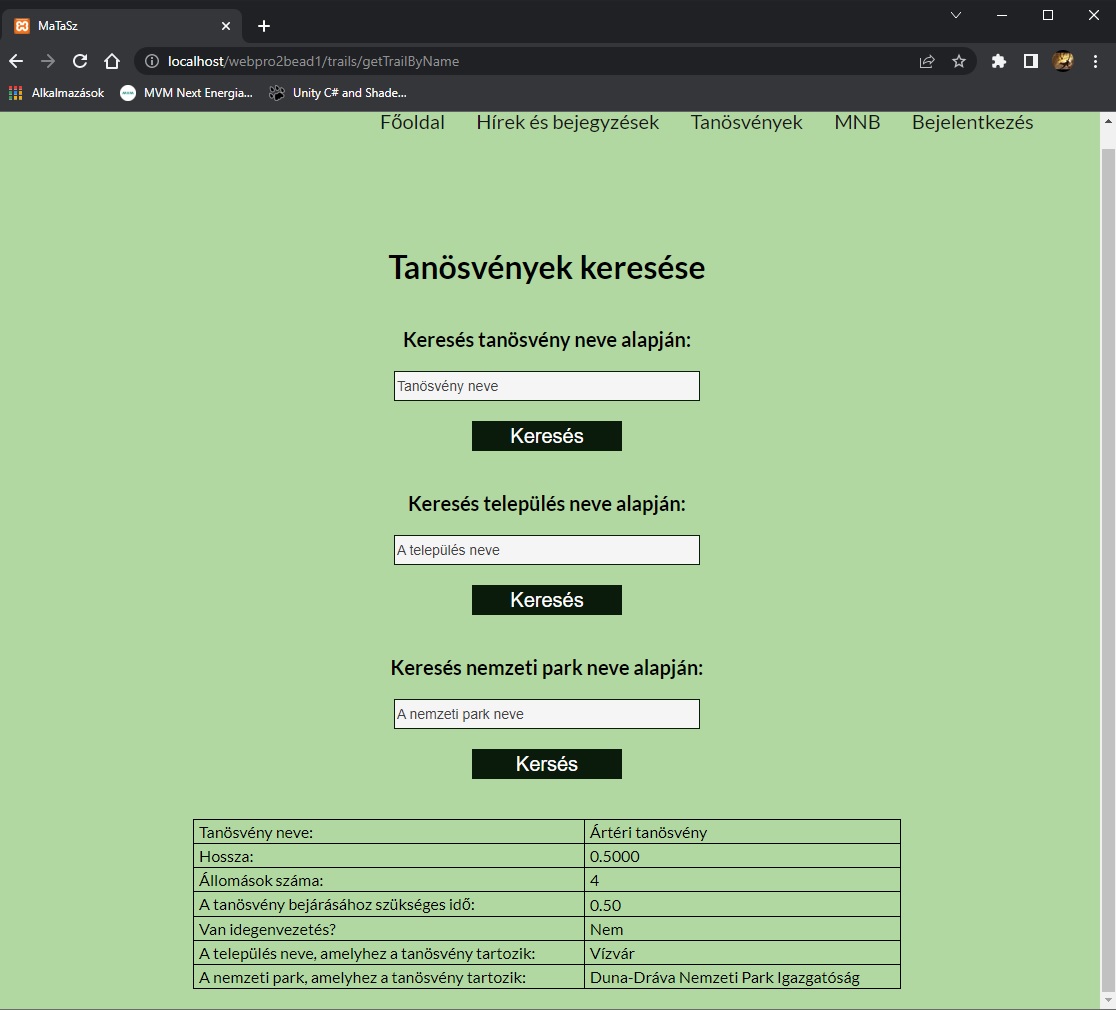
Új bejegyzés létrehozása



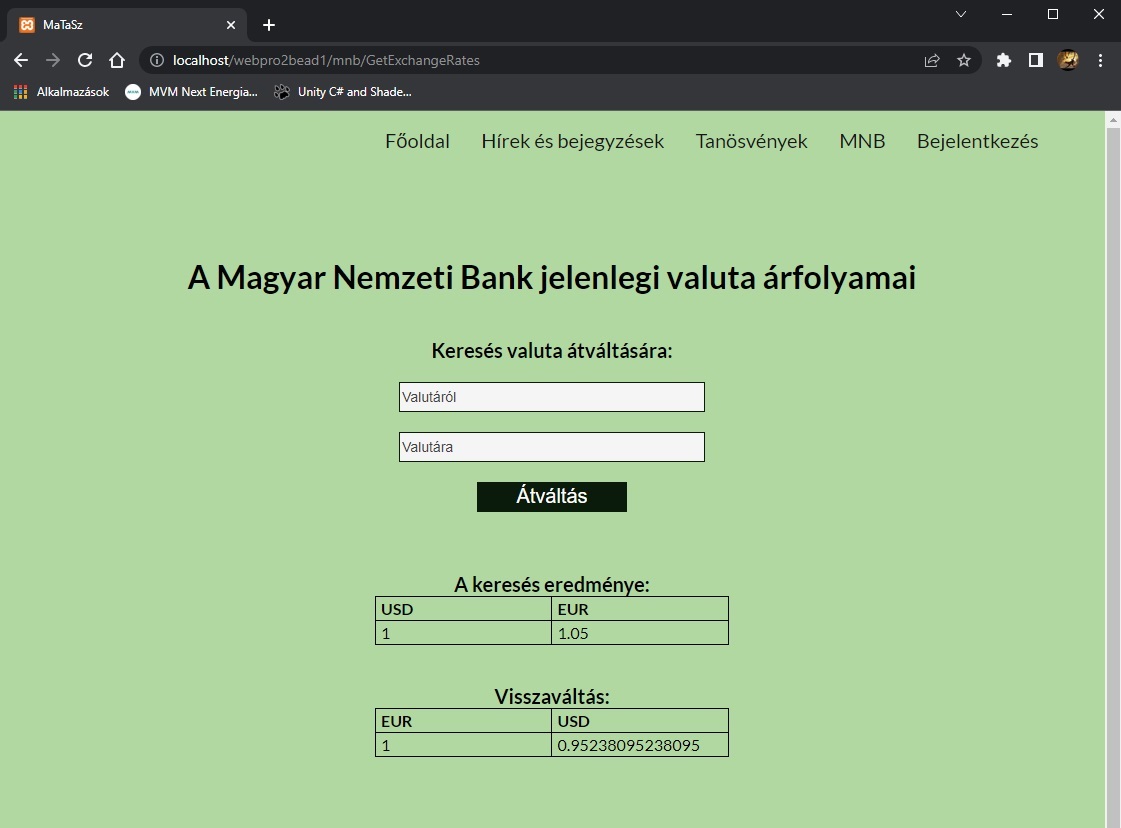
Meglévő bejegyzés módosítása



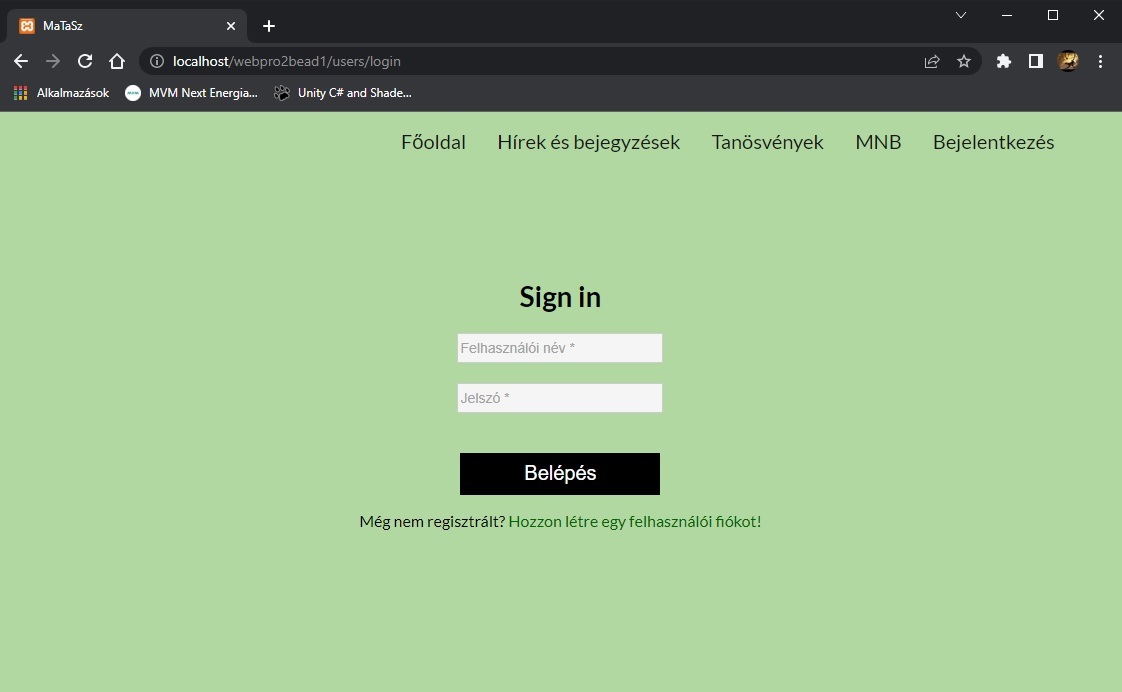
Tanösvények keresés után



MNB keresés után



Bejelentkezés



Regisztráció

