**Book My Time programdokumentáció**

**1. Témaválasztás indoklása**

A **BookMyTime** projekt célja egy olyan online időpontfoglaló rendszer létrehozása, amely egyszerűsíti és digitalizálja a szolgáltatások foglalását mind a vállalkozások, mind az ügyfelek számára.

A témaválasztás indoklása több szempontból is alátámasztható:

1. **Piaci igény és digitalizáció**  
   Napjainkban egyre több kis- és középvállalkozás tér át az online időpontfoglalásra, mivel ez hatékonyabbá teszi az ügyfélkezelést, csökkenti az adminisztrációs terheket és minimalizálja a félreértéseket. A felhasználók számára pedig kényelmesebb megoldást nyújt, hiszen gyorsan és egyszerűen foglalhatnak időpontot a különböző szolgáltatásokhoz.
2. **Automatizált és átlátható ügyfélkezelés**  
   A hagyományos, telefonon vagy személyesen történő időpontfoglalás sok esetben időigényes és hibalehetőségeket rejt magában. Egy jól működő online rendszer lehetővé teszi az időpontok valós idejű kezelését, az emlékeztetők küldését, valamint az esetleges lemondások vagy módosítások kezelését.
3. **Célcsoport és felhasználási terület**  
   A rendszer főként kisvállalkozások – például fodrászok, kozmetikusok, orvosok – számára készült, akiknek szükségük van egy megbízható, könnyen kezelhető online platformra. Az ügyfelek számára pedig gyors és egyszerű foglalási lehetőséget biztosít, amely bármikor elérhető.
4. **Szakmai fejlődés és kihívások**  
   A projekt kiváló lehetőséget biztosít a modern webfejlesztési technológiák (Node.js, Vue.js, MySQL) alkalmazására, valamint az adatbázis-kezelés, az API-fejlesztés és a felhasználói hitelesítés mélyebb megismerésére. Emellett a valós idejű szinkronizáció és a fizetési rendszer integrációja is izgalmas kihívást jelent.

Összességében a **BookMyTime** egy hasznos, valós piaci igényekre reagáló alkalmazás, amely nemcsak technológiai, hanem gazdasági és üzleti szempontból is indokolt választás.

## **Backend (Node.js) Függvények**

A backend felelős az adatok kezeléséért, a felhasználók hitelesítéséért és a foglalások adminisztrálásáért. A következő függvényeket használjuk:

### ****1. Foglalások kezelése****

* **createBooking(req, res)**
  + Új foglalást hoz létre.
  + Az ügyfél elküldi a szükséges adatokat (pl. szolgáltatás, időpont, ügyfél ID).
  + A függvény ellenőrzi az adatokat, majd elmenti azokat az adatbázisba.
  + Ha sikeres, visszaküldi az új foglalás adatait.
* **getBookings(req, res)**
  + Lekérdezi az adott felhasználó foglalásait.
  + Az ügyfél ID-ja alapján megkeresi az adatbázisban a kapcsolódó foglalásokat.
  + Az adatokat JSON formátumban visszaküldi a frontendnek.
* **updateBooking(req, res)**
  + Frissíti egy létező foglalás adatait.
  + Például módosítható az időpont, a státusz vagy más adatok.
  + Az ID alapján kikeresi a megfelelő rekordot, és frissíti az adatbázisban.
* **deleteBooking(req, res)**
  + Törli a megadott foglalást.
  + Az ID alapján megkeresi az adatbázisban, majd eltávolítja azt.

### ****2. Felhasználók kezelése (Autentikáció)****

* **registerUser(req, res)**
  + Új felhasználót regisztrál.
  + Ellenőrzi, hogy az e-mail cím még nem foglalt.
  + A jelszót biztonságosan hashelve tárolja az adatbázisban.
  + Ha sikeres, visszaküld egy megerősítő üzenetet.
* **loginUser(req, res)**
  + Ellenőrzi a felhasználó bejelentkezési adatait.
  + Ha az e-mail és a jelszó helyes, generál egy JWT tokent.
  + A tokent visszaküldi a frontendnek, amely azt a későbbi kéréseknél használja.
* **verifyToken(req, res, next)**
  + Middleware függvény, amely minden védett végpont előtt lefut.
  + Ellenőrzi, hogy a felhasználó rendelkezik-e érvényes JWT tokennel.
  + Ha nincs érvényes token, a rendszer nem engedi a további műveleteket.

### ****3. Kategóriák kezelése****

* **getCategories(req, res)**
  + Lekérdezi az összes szolgáltatási kategóriát az adatbázisból.
  + Ezeket a frontend megjeleníti az ügyfél számára.
* **addCategory(req, res)**
  + Új kategóriát ad az adatbázishoz.
  + Az adminisztrátor vagy vállalkozó hozzáadhatja a saját szolgáltatásait.

## **Frontend (Vue.js) Függvények**

A frontend felelős a felhasználói felületért és az adatok megfelelő megjelenítéséért. Az alábbi függvények kapcsolódnak a backendhez:

### ****1. Foglalások kezelése****

* **fetchBookings()**
  + A backend API-tól lekéri a felhasználó foglalásait.
  + Az adatokat egy bookings változóba menti, amely a felületen megjelenik.
* **createBooking()**
  + Elküldi az új foglalás adatait a backendnek.
  + Ha sikeres, frissíti a bookings listát.
* **updateBooking()**
  + A módosított foglalási adatokat elküldi a backendnek.
* **deleteBooking(id)**
  + Töröl egy foglalást a megadott azonosító alapján.

### ****2. Felhasználói hitelesítés****

* **login()**
  + Az e-mail és jelszó elküldése a backend felé.
  + Ha sikeres, a JWT tokent elmenti a localStorage-ba.
* **logout()**
  + Kitörli a mentett JWT tokent és kijelentkezteti a felhasználót.

### ****3. Kategóriák kezelése****

* **fetchCategories()**
  + Lekéri az elérhető kategóriákat az adatbázisból.
* **addCategory()**
  + Új kategória létrehozása az adminisztrátori felületen keresztül.

## **Backend Változók (Node.js)**

* **db** – Az adatbáziskapcsolatot tárolja (db.js fájlban).
* **jwtSecret** – A JSON Web Token titkos kulcsa, amely a hitelesítéshez kell.
* **FRONTEND\_URL** – A frontend URL-je (http://localhost:5173/).
* **PORT** – A szerver által használt port (pl. 3000).

# BookMyTime - Bejelentkezés Dokumentáció

## Bejelentkezés folyamata

A BookMyTime rendszer bejelentkezési funkciója biztosítja a felhasználók számára, hogy hitelesítsék magukat és hozzáférjenek a fiókjukhoz. A bejelentkezési folyamat során az e-mail cím és a jelszó ellenőrzése történik.

### Bejelentkezés menete és feltételei

1. A felhasználó megadja az e-mail címét és jelszavát a bejelentkezési űrlapon.
2. Az adatok Vue.js segítségével v-model bindingon keresztül kerülnek tárolásra.
3. A handleLogin függvény elküldi a bejelentkezési kérelmet a szervernek az axios.post segítségével.
4. A szerver ellenőrzi az e-mail cím és jelszó helyességét:
   * Ha az adatok helyesek, a szerver visszaküldi a felhasználóhoz tartozó hitelesítési tokent és további adatokat (pl. jogosultsági szint, azonosító).
   * Ha az adatok hibásak, hibaüzenetet küld vissza.
5. A sikeres hitelesítés után a kapott adatokat a rendszer a localStorage-ban tárolja, hogy a későbbi munkamenetek során felhasználható legyen.
6. A bejelentkezett felhasználó jogosultságának megfelelően átirányítás történik a megfelelő oldalra:
   * **Vállalkozó esetén**: /vallalkozoHome
   * **Általános felhasználó esetén**: /

### Használt változók és függvények

* **Változók:**
  + email: a felhasználó által megadott e-mail cím (ref változó Vue.js-ben)
  + password: a felhasználó által megadott jelszó (ref változó Vue.js-ben)
  + errorMessage: a hibaüzenetek tárolására szolgáló változó
* **Függvények:**
  + handleLogin(): A bejelentkezési folyamatot kezeli, elküldi az adatokat a szervernek.
  + localStorage.setItem(): A szervertől kapott hitelesítési adatokat elmenti.
  + router.push(): A megfelelő oldalra irányítja a felhasználót a jogosultsága alapján.

## Kapcsolatfelvétel

A kapcsolatfelvételi funkció lehetővé teszi, hogy a felhasználók üzenetet küldjenek az oldal üzemeltetőinek.

### Kapcsolatfelvétel menete

1. A felhasználó megadja a nevét, e-mail címét és üzenetét az űrlapon.
2. Az adatok Vue.js v-model segítségével tárolódnak a komponens állapotában.
3. A submitForm függvény meghívja a sendEmail függvényt.
4. A sendEmail függvény az emailjs könyvtár segítségével elküldi az adatokat az e-mail küldési szolgáltatáshoz.
5. Sikeres küldés esetén visszajelzés jelenik meg a felhasználónak, hogy az üzenet sikeresen elküldésre került.
6. Hiba esetén a felhasználó értesítést kap, hogy az üzenet küldése sikertelen volt.

### Használt változók és függvények

* **Változók:**
  + formData.name: a felhasználó neve
  + formData.email: a felhasználó e-mail címe
  + formData.message: a felhasználó által megadott üzenet
  + errorMessage: hibaüzenetek tárolására szolgál
  + successMessage: sikeres küldés esetén megjelenő visszajelzés
* **Függvények:**
  + submitForm(): Meghívja az sendEmail függvényt az űrlap elküldésekor.
  + sendEmail(): Az emailjs.send() metódus segítségével elküldi az üzenetet.
  + emailjs.send(): Külső e-mail küldési szolgáltatást hív meg az üzenet továbbítására.

Ezek a funkciók biztosítják, hogy a felhasználók könnyedén kapcsolatba léphessenek a szolgáltatóval és visszajelzést adhassanak az oldal működéséről.

# **Adatvédelem oldal dokumentációja**

## **1. Az Adatvédelem oldal célja**

Az Adatvédelem oldal célja, hogy tájékoztassa a felhasználókat a Book My Time rendszer adatkezelési gyakorlatáról. Az oldalon részletezésre kerülnek az adatgyűjtési, -tárolási és -feldolgozási szabályok, valamint az adatvédelmi irányelvek.

## **2. A komponens felépítése**

Az Adatvédelem oldal egy Vue komponensben került megvalósításra, amely HTML szekciókat tartalmaz az adatvédelmi szabályzat különböző pontjainak megjelenítésére.

### 2.1. HTML szerkezet

A komponens egy <section> elemet tartalmaz adatvedelem osztállyal, amely az adatvédelmi tájékoztató különböző részeit strukturált formában jeleníti meg.

Főbb részek:

* **Címek:** Az oldal címe "Általános Szerződési Feltételek (ÁSZF)".
* **Hatálybalépési dátum:** Az aktuális szabályzat hatálybalépési dátumát tartalmazza.
* **Szolgáltató adatai:** A Book My Time szolgáltató hivatalos adatai.
* **Szolgáltatás célja:** Az időpontfoglaló rendszer rövid bemutatása.
* **Felhasználói fiókok és regisztráció:** A felhasználói regisztrációval kapcsolatos információk.
* **Időpontfoglalás:** A foglalási rendszer működésének ismertetése.
* **Lemondási és módosítási feltételek:** Időpontok törlésének vagy módosításának szabályai.
* **Díjazás és fizetés:** Az online fizetések és visszatérítések módja.
* **Felelősség kizárása:** A szolgáltató felelősségvállalásának korlátai.
* **Adatvédelmi irányelvek:** Az adatkezelés részletei és az adatbiztonsági elvek.
* **Szerződés megszüntetése:** A felhasználói fiók törlésének szabályai.
* **Egyéb rendelkezések:** Az ÁSZF módosításának feltételei.

### 2.2. Stílus és kinézet

A dokumentációban nem részletezett CSS stílusokat az alkalmazás központi stíluslapjai határozzák meg. Az adatvedelem osztály biztosítja az oldal megfelelő formázását és olvashatóságát.

## **3. Komponens működése**

Az Adatvédelem oldal statikus információkat jelenít meg, így nincs szükség dinamikus adatbetöltésre vagy interaktív funkciókra. Az oldal kizárólag tájékoztatásra szolgál, és nem tartalmaz Vue komponens metódusokat vagy adatokat.

## **4. Használati esetek**

Az oldal a következő esetekben kerül megjelenítésre:

1. A felhasználók az adatvédelmi szabályzatot szeretnék elolvasni.
2. A regisztráció vagy foglalás előtt tájékozódni szeretnének az adatkezelési feltételekről.
3. A vállalkozások szeretnék ellenőrizni, hogy milyen adatvédelmi előírásokat kell betartaniuk.

Az Adatvédelem oldal biztosítja a felhasználók számára az adatkezelési elvek és feltételek átláthatóságát. Az oldal statikus tartalmú, és a Vue.js alapvető sablonszerkezetére épül, így könnyen karbantartható és bővíthető a későbbiekben.

# **Rólunk oldal dokumentációja**

## **1. Bevezetés**

A "Rólunk" oldal bemutatja a BookMyTime célkitűzéseit, szolgáltatásait és előnyeit, amelyeket az időpontfoglalás terén biztosít. Célja, hogy a felhasználók betekintést nyerjenek a rendszer működésébe és előnyeibe.

## **2. Oldal struktúra**

Az oldal főbb elemei:

* **Cím**: Rólunk
* **Bevezető szöveg**: Az oldal rövid bemutatása
* **Előnyök listája**: Az alkalmazás legfőbb jellemzői
* **Célkítűzések**: A szolgáltatás alapelvei és hosszútávú tervei

## **3. Tartalmi elemek**

### 3.1. Bevezetés

Az oldal egy barátságos és informatív hangnemben üdvözli a felhasználókat, bemutatva a BookMyTime szerepét az időpontfoglalás terén.

### 3.2. Miért válasszon minket?

* Egyszerű és intuitív kezelőfelület
* Automatikus értesítések az ügyfeleknek
* Teljesen testreszabható időpontfoglalás

### 3.3. Célkítűzések

A megbízhatóság és a felhasználóbarát megoldások biztosítása, valamint a folyamatos fejlesztés az új piaci igényekhez és technológiákhoz igazodva.

## **4. Fejlesztési szempontok**

* **Reszponzív design**: Minden eszközön megfelelő megjelenés biztosítása.
* **SEO-optimalizálás**: Az oldal keresőbarát címek és meta adatok beállítása.
* **Gyors betöltés**: Minimalizált képek és optimalizált kód alkalmazása.

## Dokumentáció: registerChoose oldal

### Áttekintés

A registerChoose oldal a felhasználóknak lehetőséget biztosít a regisztráció típusának kiválasztására. Két lehetőség közül választhatnak: **Vállalkozóként regisztrálni** vagy **Ügyfélként regisztrálni**. A felhasználó a választott lehetőségre kattintva átirányítódik a megfelelő regisztrációs oldalra.

### Struktúra

#### 1. **HTML Sablon (**template**)**

A registerChoose oldal felépítése a következő elemeket tartalmazza:

* **Regisztráció típusának választása**:  
  Két gomb található a felületen, melyek közül az egyik a vállalkozó regisztrációjára, a másik az ügyfél regisztrációjára irányítja a felhasználót.
  + "Vállalkozóként regisztrálok" gomb (redirectToBusinessRegistration metódus)
  + "Ügyfélként regisztrálok" gomb (redirectToClientRegistration metódus)
* **Oldal elrendezése**:  
  A képernyő két oszlopra van osztva:
  + **Bal oldali oszlop**: A regisztráció típusának választására szolgáló gombokkal és szöveggel.
  + **Jobb oldali oszlop**: Egy kép jelenik meg, amely a regisztrációval kapcsolatos vizuális információkat nyújt.

#### 2. **CSS Osztályok és Elrendezés**

A következő CSS osztályok biztosítják az oldal reszponzív és jól elrendezett megjelenését:

* container-fluid: Az oldal teljes szélességét kihasználó elrendezés.
* d-flex, justify-content-center, align-items-center: A bal oldali oszlop tartalmának középre igazítása.
* min-vh-100: A teljes magasságú oldal biztosítása.
* gold-button: A regisztráció típusának választásához tartozó gombok stílusa.

#### 3. **Kép**

A jobb oldali oszlopban található kép (/foto.jpg) vizuálisan támogatja az oldalon található információkat.

### Funkciók

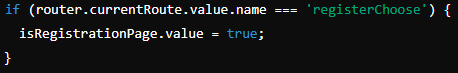
#### 1. **Regisztráció típusának választása**

* Az oldal betöltődésekor két gomb jelenik meg a felhasználónak:
  + **Vállalkozóként regisztrálok**: Erre kattintva a felhasználó átirányítódik a vallalkozoRegister oldalra, ahol a vállalkozói regisztráció kezdődhet.
  + **Ügyfélként regisztrálok**: Erre kattintva a felhasználó a register oldalra kerül, ahol ügyfélként regisztrálhat.

A választott gombok a router.push metódus segítségével irányítják át a felhasználót a megfelelő regisztrációs oldalra.

#### 2. **Oldal típusa (**isRegistrationPage**)**

A isRegistrationPage változó tárolja, hogy az aktuális oldal a regisztrációs választási oldal (registerChoose) vagy más oldal-e. Ezt a logikát a következő kódval ellenőrizzük:



Ez a feltételes logika biztosítja, hogy az oldal a megfelelő regisztrációs választóképernyőt jelenítse meg.

### Komponens és Metódusok

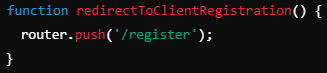
#### 1. redirectToBusinessRegistration **metódus**

Ez a metódus akkor fut le, amikor a felhasználó a "Vállalkozóként regisztrálok" gombra kattint. Az oldal átirányítja a felhasználót a vállalkozói regisztrációs oldalra:

#### 

#### 2. redirectToClientRegistration **metódus**

Ez a metódus akkor fut le, amikor a felhasználó az "Ügyfélként regisztrálok" gombra kattint. Az oldal átirányítja a felhasználót az ügyfél regisztrációs oldalra:



### Használat

Ez az oldal a felhasználók számára egy egyszerű, vizuálisan vonzó módot kínál a regisztrációs típus kiválasztására. A két választási lehetőség segítségével a felhasználók gyorsan és könnyedén kiválaszthatják, hogy vállalkozóként vagy ügyfélként kívánnak regisztrálni, és az oldal automatikusan a megfelelő regisztrációs felületre irányítja őket.

## Dokumentáció: vallalkozoRegister oldal

### Áttekintés

A vallalkozoRegister oldal a vállalkozók számára készült regisztrációs felület, ahol a felhasználók megadhatják személyes adataikat és a regisztrációhoz szükséges információkat, majd elküldhetik a rendszerbe. Az oldal több érvényesítést végez a mezők kitöltése során, beleértve a jelszó erősségét, a jelszó és annak megerősítésének egyezőségét, valamint a telefonszám formátumát.

### Struktúra

#### 1. **HTML Sablon (**template**)**

A vallalkozoRegister oldal az alábbi elemeket tartalmazza:

* **Regisztrációs űrlap**: A felhasználónak ki kell töltenie a következő mezőket:
  + **Név**: A felhasználó teljes neve.
  + **E-mail**: A felhasználó e-mail címe.
  + **Jelszó**: A felhasználó által választott jelszó.
  + **Jelszó megerősítése**: A jelszó másolata annak megerősítésére.
  + **Telefonszám**: A felhasználó telefonszáma.
  + **ÁSZF elfogadása**: Checkbox a felhasználó számára, hogy elfogadja az Általános Szerződési Feltételeket (ÁSZF).
* **Hibaüzenetek**:
  + A jelszó erőssége, ha nem felel meg a biztonsági követelményeknek.
  + A két jelszó megerősítése nem egyezik.
  + Érvénytelen telefonszám esetén hibaüzenet jelenik meg.
* **Submit gomb**: A regisztráció véglegesítése, amely akkor válik aktívvá, ha minden mező helyesen van kitöltve és az ÁSZF-et elfogadták.

#### 2. **CSS Osztályok és Elrendezés**

A következő CSS osztályok biztosítják az oldal reszponzív és jól elrendezett megjelenését:

* container-fluid: Az oldal teljes szélességét kihasználó elrendezés.
* d-flex, justify-content-center, align-items-center: A bal oldali oszlop tartalmának középre igazítása.
* min-vh-100: A teljes magasságú oldal biztosítása.
* register-form: A regisztrációs űrlap stílusa.

#### 3. **Kép**

A jobb oldali oszlopban található kép (/foto.jpg) vizuálisan támogatja az oldalon található információkat.

### Funkciók

#### 1. **Jelszó erősségi ellenőrzés**

A jelszó erősségét az oldal automatikusan ellenőrzi a következő szabályok alapján:

* A jelszónak legalább 8 karakter hosszúnak kell lennie.
* Kisbetűt, nagybetűt és számot kell tartalmaznia.

Ha a jelszó nem felel meg ezen követelményeknek, az oldal egy figyelmeztetést jelenít meg a felhasználó számára.

#### 2. **Jelszó megerősítése**

A felhasználónak meg kell erősítenie a jelszót. Ha a két jelszó nem egyezik, egy hibaüzenet jelenik meg, jelezve a problémát.

#### 3. **Telefonszám érvényesítése**

A telefonszám mező ellenőrzi, hogy a felhasználó helyes telefonszámot adott-e meg, a következő regex mintát követve: ^\+?(\d{1,3})?(\d{8})$. Ha a megadott szám nem felel meg, a rendszer egy hibaüzenetet jelenít meg.

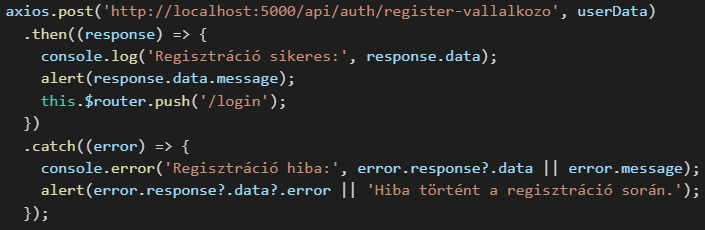
#### 4. **Form Validáció**

A isFormValid számított tulajdonság (computed property) határozza meg, hogy a regisztrációs űrlap érvényes-e. Az űrlap akkor érvényes, ha:

* Minden mező ki van töltve.
* A jelszó és a jelszó megerősítése megegyezik.
* A jelszó eléri a minimális erősséget.
* A telefonszám helyes formátumban van.

#### 5. **Regisztrációs kérelem elküldése**

Amikor a felhasználó a "Regisztráció" gombra kattint, a handleRegistration metódus fut le. Ha az űrlap érvényes, az adatokat (név, e-mail, jelszó, telefonszám) elküldi a backend API-nak a POST kérésen keresztül:

Ha a regisztráció sikeres, a felhasználó átirányításra kerül a bejelentkezési oldalra.

#### 6. **ÁSZF elfogadása**

A felhasználónak el kell fogadnia az ÁSZF-et a regisztráció előtt. Ha az ÁSZF nincs elfogadva, a regisztrációs gomb le van tiltva.

### Komponens és Metódusok

#### 1. handleRegistration **metódus**

Ez a metódus felelős az űrlap elküldéséért és a regisztrációs adatok API-ba való továbbításáért. Ha a regisztráció érvényes, a felhasználó regisztrálása megtörténik, és a felhasználó átirányításra kerül a bejelentkezési oldalra.



### Összegzés

A vallalkozoRegister oldal egy jól strukturált regisztrációs felület, amely biztosítja, hogy a vállalkozók megfelelő adatokat adjanak meg, és megfeleljenek a biztonsági előírásoknak. Az oldal érvényesítési mechanizmusokat használ a jelszó erősségének, a jelszavak egyezőségének és a telefonszám formátumának ellenőrzésére. A sikeres regisztráció után a felhasználó átirányításra kerül a bejelentkezési oldalra.

## **Dokumentáció: RegistrationPage (Ügyfél Regisztráció)**

### ****Leírás****

A RegistrationPage oldal a felhasználói regisztrációt teszi lehetővé egy ügyfél számára. Az oldalon található regisztrációs űrlap a következő adatokat kéri a felhasználótól: teljes név, e-mail cím, jelszó, jelszó megerősítés, telefonszám, valamint az ÁSZF elfogadása.

### ****Funkciók****

* **Név**: A felhasználó teljes neve.
* **E-mail cím**: A felhasználó e-mail címe, amely a regisztrációhoz szükséges.
* **Jelszó**: A felhasználó választott jelszava, amelynek meg kell felelnie bizonyos erősségi kritériumoknak.
* **Jelszó megerősítése**: A felhasználó által megadott jelszó újraellenőrzése.
* **Telefonszám**: A felhasználó telefonszáma, amely validálásra kerül.
* **ÁSZF elfogadása**: A felhasználó beleegyezése az ÁSZF elfogadásába.
* **Regisztráció**: A felhasználó regisztrálása a rendszerbe, amennyiben minden mező helyesen van kitöltve és az ÁSZF-et elfogadták.

### ****Adatok****

A regisztrációs űrlap a következő adatokat várja el a felhasználótól:

* **name**: A felhasználó teljes neve.
* **email**: A felhasználó e-mail címe.
* **password**: A felhasználó jelszava.
* **confirmPassword**: A jelszó megerősítése.
* **phone**: A felhasználó telefonszáma.
* **termsAccepted**: Jelölőnégyzet, amely a felhasználó beleegyezését jelzi az ÁSZF elfogadásába.

### ****Validáció****

1. **Jelszó erősség**: A jelszónak legalább 8 karakter hosszúnak kell lennie, tartalmaznia kell kisbetűt, nagybetűt és számot.
2. **Jelszó megerősítése**: A két jelszónak meg kell egyeznie.
3. **Telefonszám**: A telefonszámnak érvényes formátumban kell szerepelnie, amely a következő mintát követi: +XXX-XXXX-XXXX vagy XXX-XXXX-XXXX.
4. **Form Validation**: A regisztráció csak akkor lesz érvényes, ha minden mező ki van töltve, a jelszó erősségi követelmények teljesülnek, és az ÁSZF-et elfogadták.

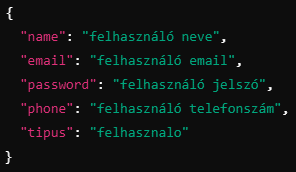
### ****Hibaüzenetek****

* **Jelszó erőssége**: Ha a jelszó nem felel meg a követelményeknek, hibaüzenetek jelennek meg, amelyek tájékoztatják a felhasználót a szükséges javításról.
* **Jelszó megerősítése**: Ha a két jelszó nem egyezik, egy hibaüzenet figyelmezteti a felhasználót.
* **Telefonszám**: Ha a telefonszám nem felel meg a szabályoknak, a rendszer hibaüzenetet ad, és kéri a helyes formátumú telefonszám megadását.
* **ÁSZF**: Ha a felhasználó nem fogadja el az ÁSZF-et, a regisztráció gomb inaktív marad, amíg el nem fogadja azt.

### ****API integráció****

* Az adatokat az űrlap elküldi egy HTTP POST kérés formájában a következő API endpointra:
* http://localhost:5000/api/auth/register

Az API az adatokat a következő formátumban várja:



A válasz alapján a felhasználó kap egy üzenetet, amely tájékoztatja a sikeres regisztrációról, vagy hiba esetén egy hibaüzenetet jelenít meg.

### ****Funkciók részletes leírása****

1. **Regisztrációs űrlap**: A felhasználó adatainak begyűjtése az űrlapon történik. Az űrlap tartalmazza a szükséges mezőket: név, e-mail, jelszó, jelszó megerősítése, telefonszám, és egy checkbox az ÁSZF elfogadására.
2. **Jelszó validálása**: A felhasználónak erős jelszót kell választania, amely minimum 8 karakter hosszú, tartalmaz kisbetűt, nagybetűt és számot. Ha a jelszó nem felel meg ezeknek a kritériumoknak, a rendszer tájékoztatja a felhasználót a hibáról.
3. **Telefon validálása**: A rendszer ellenőrzi, hogy a megadott telefonszám érvényes formátumban van-e. Ha nem, a rendszer hibát jelez, és kéri a helyes telefonszámot.
4. **Form validálás és gomb állapota**: A regisztrációs gomb csak akkor lesz aktív, ha minden mező ki van töltve és az ÁSZF-et is elfogadták.
5. **Sikeres regisztráció**: Ha a regisztráció sikeres, a felhasználót átirányítja a login oldalra, és egy értesítést kap a sikeres regisztrációról.
6. **Hiba esetén**: Ha bármi hiba történik a regisztráció során, például ha a felhasználó adatai nem megfelelőek vagy az API nem válaszol, a rendszer egy hibaüzenetet jelenít meg a felhasználónak.

### ****Követelmények és függőségek****

* **Vue.js**: A regisztrációs oldal Vue.js alapú, és a form validálás JavaScript-ben történik.
* **Axios**: Az axios könyvtárat használjuk HTTP kérések küldésére.
* **Router**: A regisztráció után a felhasználó átirányításra kerül a login oldalra a Vue Router segítségével.

### ****Stílus****

A regisztrációs oldal reszponzív, és két részből áll: egy bal oldali regisztrációs űrlapból és egy jobb oldali képből. A háttérkép és a formázás biztosítja a vizuálisan vonzó elrendezést.

### ****BookMyTime MySQL Adatbázis Részletes Dokumentáció****

Az alábbi dokumentáció a **BookMyTime** projekt MySQL adatbázisának részletes leírását tartalmazza, beleértve a táblákat, azok célját a rendszerben, az oszlopok szerepét, valamint a táblák közötti kapcsolatokat és azok indoklását.

## **1. Adatbázis Áttekintés**

* **Adatbázis neve:** bookmytime
* **Használt motor:** InnoDB (a tranzakciókezelés és adatkonzisztencia miatt)
* **Karakterkódolás:** utf8mb3\_hungarian\_ci
* **Fő cél:** Online időpontfoglaló rendszer, amely lehetővé teszi a felhasználók számára, hogy különböző szolgáltatókhoz időpontot foglaljanak.

### ****A rendszer fő funkciói:****

1. **Felhasználók kezelése** – Regisztráció, bejelentkezés, jogosultságkezelés.
2. **Vállalkozások kezelése** – Szolgáltatók regisztrációja és szolgáltatásaik listázása.
3. **Szolgáltatások létrehozása** – Egy vállalkozás meghatározhatja, milyen szolgáltatásokat nyújt.
4. **Időpontok kezelése** – A szolgáltatásokhoz kapcsolódó elérhető időpontok tárolása.
5. **Foglalások kezelése** – A felhasználók és vállalkozók által lefoglalt időpontok rögzítése.

## **2. Táblák és Kapcsolatok**

### ****2.1.**** felhasznalo ****– Felhasználók****

**Célja:** Ez a tábla tárolja a weboldal felhasználóit, akik lehetnek ügyfelek vagy vállalkozók.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

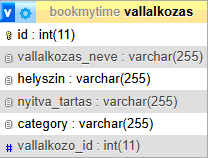
****

**Kapcsolatai:**

* **foglalasok táblával:** Ha az ügyfél foglal egy időpontot, az ő felhasznalo\_id-ja bekerül a foglalasok táblába.
* **vallalkozas táblával:** Ha a felhasználó vállalkozó, akkor egy vagy több vállalkozást regisztrálhat.

### ****2.2.**** vallalkozas ****– Vállalkozások****

**Célja:** Ez a tábla tartalmazza azokat a vállalkozásokat, amelyek szolgáltatásokat nyújtanak.

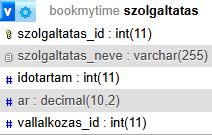


**Kapcsolatai:**

* **felhasznalo táblával:** Minden vállalkozás egy vállalkozóhoz tartozik (vallalkozo\_id).
* **szolgaltatas táblával:** Egy vállalkozás több szolgáltatást is kínálhat.

### ****2.3.**** szolgaltatas ****– Szolgáltatások****

**Célja:** Egy vállalkozás által nyújtott konkrét szolgáltatások (pl. hajvágás, masszázs) listázása.

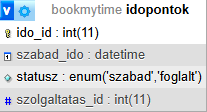


**Kapcsolatai:**

* **vallalkozas táblával:** Egy vállalkozás több szolgáltatást is kínálhat.
* **idopontok táblával:** Egy szolgáltatásnak több elérhető időpontja lehet.

### ****2.4.**** idopontok ****– Időpontok****

**Célja:** A szolgáltatásokhoz tartozó foglalható időpontokat tárolja.



**Kapcsolatai:**

* **szolgaltatas táblával:** Egy időpont egy konkrét szolgáltatáshoz tartozik.
* **foglalasok táblával:** Ha egy időpontot lefoglalnak, megjelenik a foglalasok táblában.

### ****2.5.**** foglalasok ****– Foglalások****

**Célja:** A felhasználók által lefoglalt időpontokat rögzíti.



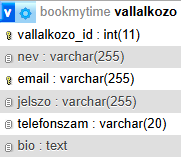
**Kapcsolatai:**

* **felhasznalo táblával:** Minden foglalás egy ügyfélhez tartozik.
* **idopontok táblával:** Az időpontok foglalási állapotát frissíti.
* **szolgaltatas és vallalkozas táblával:** Minden foglalás egy adott szolgáltatáshoz és vállalkozáshoz tartozik.

Igazad van, a **vallalkozo** tábla is fontos része az adatbázisnak, hiszen a rendszerben a vállalkozók külön szerepkörrel rendelkeznek, és ők kezelik a szolgáltatásaikat. Most kiegészítem a dokumentációt ezzel a táblával és annak kapcsolataival.

## **2.6.** vallalkozo **– Vállalkozók**

**Célja:**  
Ez a tábla tartalmazza a vállalkozókat, vagyis azokat a felhasználókat, akik szolgáltatásokat kínálnak és időpontfoglalásokat fogadnak el.



**Kapcsolatai:**

* **felhasznalo táblával:** Minden vállalkozó egy felhasználó is egyben (felhasznalo\_id = vallalkozo\_id), és csak azok a felhasználók lehetnek vállalkozók, akiknek a felhasznalo.szerep = 'vallalkozo'.
* **vallalkozas táblával:** Egy vállalkozó több vállalkozást is létrehozhat.
* **szolgaltatas és foglalasok táblákkal:** A vállalkozó által nyújtott szolgáltatásokat és foglalásokat kezeli.

## **Kapcsolatok a** vallalkozo **táblával**

1. **felhasznalo → vallalkozo** (1:1 kapcsolat)
   * Egy vállalkozó mindig egy felhasznalo rekordhoz kapcsolódik.
   * Egy felhasználó csak akkor lehet vállalkozó, ha ezt a státuszt megkapja.
2. **vallalkozo → vallalkozas** (1:N kapcsolat)
   * Egy vállalkozó több vállalkozást is regisztrálhat.
   * Minden vállalkozás egy konkrét vállalkozóhoz tartozik.
3. **vallalkozas → szolgaltatas** (1:N kapcsolat)
   * Egy vállalkozás több szolgáltatást is kínálhat.
4. **szolgaltatas → idopontok → foglalasok** (N:N kapcsolat)
   * Egy szolgáltatásnak több elérhető időpontja lehet, amelyeket az ügyfelek lefoglalhatnak.
   * A vállalkozó látja a foglalásokat és módosíthatja azok állapotát.

## **Miért van szükség külön** vallalkozo **táblára?**

* Ha csak a felhasznalo tábla tárolná a vállalkozókat, nem tudnánk külön kezelni a vállalkozók adatait (pl. adószám, weboldal).
* A vállalkozókhoz egy vagy több vállalkozás is tartozhat, ezért kell egy külön **vallalkozas** tábla is.
* A vállalkozók és a vállalkozásaik közötti kapcsolat jobban követhető és bővíthető, például később hozzáadhatunk további mezőket a vállalkozói profilokhoz.

## **3. Táblák közötti kapcsolatok magyarázata**

* Egy felhasználó több vállalkozást is regisztrálhat.
* Egy vállalkozás több szolgáltatást is kínálhat.
* Egy szolgáltatásnak több elérhető időpontja lehet.
* Az időpontokat a felhasználók vagy vállalkozók foglalhatják le.
* A foglalás rögzíti, hogy melyik felhasználó melyik időpontra foglalt.