**Fejlesztői Dokumentáció**

****

**Tartalom jegyzék**

Tartalomjegyzék

[1. Front End 4](#_Toc195789684)

[1.1. UserComponent 4](#_Toc195789685)

[1.2. UserComponent.html 7](#_Toc195789686)

[1.3. HomeComponent 10](#_Toc195789687)

[1.4. HomeComponentHTML 11](#_Toc195789688)

[1.5. LoginComponent 13](#_Toc195789689)

[1.6. LoginComponentHTML 15](#_Toc195789690)

[1.7. RegisterComponent 16](#_Toc195789691)

[1.8. RegisterComponentHTML 19](#_Toc195789692)

[1.9. WebshopPageComponent 22](#_Toc195789693)

[1.10. WebshopPageComponentHTML 24](#_Toc195789694)

[1.11. AddProductComponent 25](#_Toc195789695)

[1.12. AddProductComponentHTML 28](#_Toc195789696)

[1.13. CartPageComponent 30](#_Toc195789697)

[1.14. CartPageComponentHTML 32](#_Toc195789698)

[1.15. DeliveryPageComponent 35](#_Toc195789699)

[1.16. DeliveryPageComponentHTML 37](#_Toc195789700)

[1.17. EditProductDialogComponent 39](#_Toc195789701)

[1.18. EditProductDialogComponentHTML 41](#_Toc195789702)

[1.19. NavbarComponent 43](#_Toc195789703)

[1.20. NavbarComponentHTML 45](#_Toc195789704)

[1.21. PricesComponent 47](#_Toc195789705)

[1.22. PricesComponentHTML 48](#_Toc195789706)

[1.23. ProductDetailsDialogComponent 49](#_Toc195789707)

[1.24. ProductDetailsDialogComponentHTML 50](#_Toc195789708)

[1.25. ServiceComponent 53](#_Toc195789709)

[1.26. ServiceComponentHTML 55](#_Toc195789710)

[1.27. CartService 57](#_Toc195789711)

[1.28. ProductService 58](#_Toc195789712)

[1.29. Cartosztály 59](#_Toc195789713)

[1.30. ApiService 59](#_Toc195789714)

[1.31. AppRoutingModule 62](#_Toc195789715)

[1.32. AppModule 64](#_Toc195789716)

[1.33. AuthGuard 65](#_Toc195789717)

[1.34. AuthService 66](#_Toc195789718)

[1.35. BookingService 67](#_Toc195789719)

[1.36. ProductService 68](#_Toc195789720)

[1.37. ShoppingCartService 69](#_Toc195789721)

[1.38. UserService 70](#_Toc195789722)

[1.39. Loading component, HTML 71](#_Toc195789723)

[1. Back End 71](#_Toc195789724)

[1.1 BookingController 71](#_Toc195789725)

[1.2 CartController 74](#_Toc195789726)

[1.3 ProductController 78](#_Toc195789727)

[1.4 UsersController 80](#_Toc195789728)

[1.5 BookingConfirmationMail 85](#_Toc195789729)

[1.6 DeliveryConfirmationMail 86](#_Toc195789730)

[1.7 RegistrationSuccessful 88](#_Toc195789731)

[1.8 Modellek 89](#_Toc195789732)

[1.9 Booking 89](#_Toc195789733)

[1.10 Cart 91](#_Toc195789734)

[1.11 Delivery 92](#_Toc195789735)

[1.12 Product 94](#_Toc195789736)

[1.13 User 95](#_Toc195789737)

[1.14 Migrációk 98](#_Toc195789738)

[1.15 add**\_**role**\_**to**\_**user**\_**table 98](#_Toc195789739)

[1.16 create**\_**bookings**\_**table 99](#_Toc195789740)

[1.17 create**\_**products**\_**table 100](#_Toc195789741)

[1.18 add**\_**time**\_**to**\_**bookings**\_**table 101](#_Toc195789742)

[1.19 add**\_**gender**\_**to**\_**bookings**\_**table 102](#_Toc195789743)

[1.20 create**\_**deliveries**\_**table 103](#_Toc195789744)

[1.21 add**\_**email**\_**to**\_**bookings**\_**table 104](#_Toc195789745)

[1.22 create**\_**carts**\_**table 104](#_Toc195789746)

[1.23 DatabaseSeeder 106](#_Toc195789747)

[1.24 Útvonalak 107](#_Toc195789748)

[1.25 web 110](#_Toc195789749)

**Bevezető**

A projekt célja egy korszerű, funkcionálisan sokoldalú weboldal létrehozása egy fodrászszalon számára, amely magában foglalja az alapvető bemutatkozó funkciókat, egy online időpontfoglalási rendszert, valamint egy egyszerű webshopot is, ahol hajápolási termékek vásárolhatók. A weboldal koncepciója egy valós igényből indult ki: a nagynéném aki fodrászként dolgozik, megkeresett bennünket azzal a kéréssel, hogy szeretne egy olyan digitális felületet, amely segíti a vállalkozása fejlődését, és megkönnyíti az ügyfeleivel való kapcsolattartást.

A mai digitális világban egy vállalkozás online jelenléte elengedhetetlen. Egy átlátható, könnyen kezelhető weboldal nemcsak a szolgáltatások bemutatását segíti elő, hanem növeli az ügyfelek bizalmát is, különösen, ha a látogatók lehetőséget kapnak arra, hogy egyszerűen időpontot foglaljanak, vagy akár termékeket rendeljenek otthonról. A célunk az volt, hogy egy olyan rendszert hozzunk létre, amely technológiailag megbízható, ugyanakkor a felhasználók számára intuitív és könnyen használható.

A webshop modul lehetővé teszi a szalonban forgalmazott termékek például samponok, balzsamok, hajolajak online értékesítését, ezzel új bevételi forrást nyitva a vállalkozás számára. Az időpontfoglalási rendszer segítségével pedig az ügyfelek gyorsan, néhány kattintással lefoglalhatják a számukra megfelelő időpontot, ezáltal csökken a telefonos egyeztetés szükségessége, és hatékonyabbá válik az időgazdálkodás.

Ez a projekt nemcsak gyakorlati lehetőséget adott számunkra a webfejlesztési ismereteink alkalmazására, hanem hozzájárult egy kisvállalkozás digitális fejlődéséhez is. A munka során fontos szempont volt, hogy a kész rendszer könnyen bővíthető és karbantartható legyen, hogy hosszú távon is használható maradjon. Összességében úgy gondoljuk, hogy a projekt hasznos és értékteremtő, mivel egy konkrét, valós probléma megoldására ad gyakorlati választ.

# Front End

# UserComponent

A UserComponent felelős a felhasználói adatok megjelenítéséért, módosításáért, valamint a foglalások kezeléséért. A komponens adminisztratív műveleteket is lehetővé tesz, mint például admin jog adása vagy elvétele más felhasználóktól. Továbbá lehetőséget biztosít a jelszó módosítására és a felhasználónév frissítésére is.

**ngOnInit**

* **Leírás**:  
  Inicializálja a komponenst, beállítja a nyelvet az URL alapján, valamint ellenőrzi a felhasználó tokenjét. Ha nincs token, automatikusan átirányít a bejelentkező oldalra.
* **Funkciók**:
  + Token ellenőrzés a localStorage-ból.
  + Nyelv beállítása az ActivatedRoute alapján.
  + Meghívja a getUserInfo metódust az adatok betöltéséhez.
  + Beállítja, hogy a felhasználó admin/szuperadmin-e.

**getUserInfo**

* **Leírás**:  
  Lekéri a bejelentkezett felhasználó adatait és szerepkörét.
* **Funkciók**:
  + Token ellenőrzés és továbbítás az API híváshoz.
  + Hibakezelés, ha nincs token vagy ha sikertelen az API válasz.
  + A válasz alapján frissíti a userInfo és userRole mezőket.

**logout**  
Kijelentkezteti a felhasználót a rendszerből.

* **Funkciók**:
  + API hívás a szerverhez a kijelentkezéshez.
  + A token eltávolítása a localStorage-ból.
  + Átirányítás a bejelentkező oldalra.

**Admin jogok kezelése**

* **giveAdmin**
  + E-mail alapján admin jogot ad egy felhasználónak.
* **depriveAdmin**
  + E-mail alapján megvonja az admin jogot egy felhasználótól.
* **Funkciók mindkettőnél**:
  + API hívás a megfelelő művelethez.
  + Visszajelzés a felhasználónak sikeres vagy sikertelen végrehajtásról.
  + Adatok frissítése a getUserInfo meghívásával.

**Felhasználónév módosítás**

* **changeUsername**
  + A felhasználó új nevet adhat meg.
  + Üres érték esetén hibaüzenet jelenik meg.
  + API hívás történik a név frissítésére.
  + Sikeres művelet után a UsernameDiv bezárásra kerül, az adatok frissülnek.

**Jelszó módosítása**

* **changePassword\_div / PasswordDiv\_close**
  + Megnyitja vagy bezárja a jelszó módosító felületet.
* **PasswordDiv\_next**
  + Ellenőrzi a megadott régi jelszót API híváson keresztül.
  + Ha helyes, továbblépteti a felhasználót az új jelszó megadására.
* **changePassword**
  + Validációkat futtat: üres mező, hossz, egyezés, formátum, régi jelszóval való azonosság.
  + Ha minden feltétel teljesül, meghívja a changePassword\_api metódust.
* **changePassword\_api**
  + API hívás az új jelszó mentésére.

**Foglalások kezelése**

* **loadUserBookings / openBookingsModal / closeBookingsModal**
  + Betölti és megjeleníti a felhasználó saját foglalásait.
  + A getUserBookings metódus lekérdezi a foglalásokat e-mail és szerepkör alapján.
* **loadAllBookings / openAllBookingsModal / closeAllBookingsModal**
  + Admin funkció: összes foglalás betöltése és megjelenítése.
  + Foglalások dátum és idő szerint rendezve.
* **deleteBooking**
  + Egy adott foglalás törlése.
  + Tokenes hitelesítés mellett HTTP DELETE kérést küld.

**Egyéb vezérlő metódusok**

* **give / deprive / previousStep**
  + Az admin jog kezelő lépések közötti navigáció.
* **Username / Username\_back**
  + A név módosítására szolgáló nézet megnyitása vagy visszalépés.
* **addProductPage**
  + Átirányít a termékkezelő oldalra.
* **showAllBookedDates**
  + Gombnyomás esemény, amely későbbi fejlesztéshez tartalmaz helykitöltő logikát.

**Segédfüggvények**

* **containsPunctuation**
  + Ellenőrzi, hogy a megadott jelszó tartalmaz-e tiltott írásjeleket.
* **getUserRoleTranslationKey**
  + Szerepkör kulcs visszaadása a megadott számból (0, 1, 2).

## UserComponent.html

Ez a sablon a UserComponent vizuális megjelenéséért felelős, és dinamikusan rendereli a felhasználói információkat, szerepkör-alapú funkciókat (admin, superadmin), valamint lehetőséget ad a felhasználónév és jelszó módosítására, illetve a foglalások megtekintésére.

**Felhasználói információk (user-div-info)**

Cél: A bejelentkezett felhasználó adatainak megjelenítése és az esetleges hibák kezelése.

Elemek:

* h1: "User Information" cím, fordítással.
* \*ngIf="userInfo": Ha elérhető a userInfo objektum, megjelenik:
  + Szerepkör (userRole) fordítása: getUserRoleTranslationKey(userRole) | translate
  + Név megjelenítése és név változtató gomb (Username() hívása)
  + E-mail cím megjelenítése
* Hibaüzenetek:
  + userError: Nem sikerült lekérni az adatokat.
  + userloginError: Nem bejelentkezett felhasználó.
  + logoutError: Sikertelen kijelentkezés.
  + Betöltés állapota: ha nincs userInfo és nincs hiba → "Loading user information..."

**Superadmin funkciók (user-div-superadmin)**

Feltétel: userRole == '2'

Cél: A superadmin szerepkörű felhasználók külön jogosultságainak kezelése, mint admin jogok kiosztása vagy visszavonása.

Lépések (currentStep alapján):

1. Alap gombok:
   * Give Admin
   * Deprive Admin
   * Product Management
   * View All Bookings
2. Admin jogosultság adása:
   * E-mail bemenet [(ngModel)]="giveEmail"
   * giveAdmin() hívás
3. Admin jogosultság visszavonása:
   * E-mail bemenet [(ngModel)]="depriveEmail"
   * depriveAdmin() hívás
4. Sikerüzenet
5. Hibaüzenet

Stílus: Animált megjelenítés fade-in osztállyal.

**Admin funkciók (user-div-admin)**

Feltétel: userRole == '1'

Cél: Admin szerepkörű felhasználók jogosultságai, mint a termékkezelés és foglalások megtekintése.

Gombok:

* addProductPage()
* openAllBookingsModal()

**Felhasználónév módosítása (user-div-username-body)**

Feltétel: UsernameDiv === true

Cél: A felhasználó módosíthatja a felhasználónevét.

Mezők és validációk:

* [(ngModel)]="newUsername" – max. 20 karakter
* Hiba: ha üres – usernameError
* Gombok:
  + Módosítás: changeUsername()
  + Vissza: Username\_back()

Saját foglalások (user-div-info2)

Gomb:

* openBookingsModal()

Összes foglalás (Admin/Superadmin) (user-div-info2)

Feltétel: userRole === '2' || userRole === '1'

Gomb:

* openAllBookingsModal()

**Foglalások modálablakok**

Saját foglalások (isBookingsModalOpen):

* Lista: \*ngFor="let booking of userBookings"
* Mezők: dátum, idő, nem
* Hiba: ha nincs foglalás

Összes foglalás (isAllBookingsModalOpen):

* Lista: \*ngFor="let booking of allBookings"
* Mezők: dátum, idő, nem, e-mail
* Törlés: deleteBooking(booking.id)
* Hiba: ha nincs foglalás

Jelszó módosítása (user-div-password-body)

Feltétel: PasswordDiv === true

Lépések:

1. Régi jelszó megadása (user\_div\_password\_div\_input\_number === 1)
   * Validációk: üres, hibás jelszó, betöltés
2. Új jelszó és megerősítés (input\_number === 2)
   * Validációk:
     + Üres mezők
     + Nem egyező jelszavak
     + Nem megengedett karakterek
     + Minimális/hossz maximális hiba
     + Azonos az előzővel
     + API hiba
3. Sikeres változtatás (PasswordDiv\_ok) – visszajelzés

**Kilépés és jelszó módosító gomb (user-div-button)**

* changePassword\_div() – jelszó módosítási felület megnyitása
* logout() – kijelentkezés

## HomeComponent

A **HomeComponent** egy Angular komponens, amely a kezdőoldal animációinak és nyelvi beállításainak kezeléséért felel. Az [IntersectionObserver](about:blank) API segítségével figyeli az elemek megjelenését a nézetben, és különböző animációkat alkalmaz az elemek osztályai alapján. Emellett a komponens támogatja a dinamikus nyelvváltást az URL paraméterek alapján.

**Főbb funkciók**

**Animációk kezelése**

* Az [IntersectionObserver](about:blank) API figyeli az elemek megjelenését a nézetben.
* Az elemek osztályai alapján különböző animációkat rendel hozzájuk.
* Az animációk reszponzívak, a képernyő méretéhez igazodnak.

**Nyelvi támogatás**

* Az URL paraméterekből ([lang](about:blank)) olvassa ki az aktuális nyelvet.
* Az @ngx-translate/core könyvtár segítségével állítja be a nyelvet.

**Konstruktor**

A konstruktor figyeli az URL paramétereket az [ActivatedRoute](about:blank) segítségével. Ha a [lang](about:blank) paraméter elérhető, frissíti az aktuális nyelvet, és meghívja a [TranslateService](about:blank) megfelelő metódusát a nyelv beállításához.

**Funkciók**

* Az aktuális nyelv ([currentLang](about:blank)) frissítése az URL alapján.
* A fordítási szolgáltatás ([TranslateService](about:blank)) inicializálása.

**ngOnInit**

**Leírás**

Az [ngOnInit](about:blank) metódus jelenleg üres, de későbbi bővítésekhez használható. Például inicializálási logikák vagy adatok betöltése kerülhet ide.

**ngAfterViewInit**

Az [ngAfterViewInit](about:blank) metódus felelős az animációk inicializálásáért és az elemek megfigyeléséért az [IntersectionObserver](about:blank) segítségével.

**Funkciók**

1. **Reszponzív küszöbértékek**:
   * A képernyő méretétől függően állítja be az [IntersectionObserver](about:blank) küszöbértékét.
   * Mobil eszközökön alacsonyabb küszöbértéket használ, hogy az animációk hamarabb induljanak.
2. **Animációk alkalmazása**:
   * Az elemek osztályai alapján különböző animációkat rendel hozzájuk.
3. **Megfigyelés megszüntetése**:
   * Az [IntersectionObserver](about:blank) leállítja az elemek megfigyelését, miután az animációk lefutottak, ezzel optimalizálva a teljesítményt.

**Tulajdonságok**

**items**

Egy egyszerű tömb, amely placeholderként szolgál. Jelenleg nem használ konkrét funkciót, de későbbi bővítésekhez előkészített.

**children**

Az összes #child referencia változóval rendelkező DOM elem lekérdezése. Ezek az elemek kerülnek megfigyelésre az [IntersectionObserver](about:blank) által.

**meteorContainer**

Az összes #meteorContainer referencia változóval rendelkező DOM elem lekérdezése. Jelenleg nem használ konkrét funkciót, de későbbi bővítésekhez előkészített.

## HomeComponentHTML

A **HomeComponent** HTML sablonja a kezdőoldal vizuális megjelenítéséért felel. A sablon különböző szekciókat tartalmaz, amelyek animációkkal és fordításokkal gazdagítják a felhasználói élményt. A tartalom reszponzív, és a nyelvi támogatás az @ngx-translate/core könyvtár segítségével valósul meg.

**Főbb szekciók**

**1. Bevezető szekció**

* A divcontainer osztályú szekció tartalmazza az üdvözlő szövegeket és képeket.
* Az elemek fordítási kulcsokkal (translate pipe) jelennek meg, így többnyelvű támogatást biztosítanak.
* Az animációk a #child referencia változóval ellátott elemekhez kapcsolódnak.

**2. Időpontfoglalás szekció**

* A divcontainer II osztályú szekció az időpontfoglalás előnyeit mutatja be.
* Képek és szövegek kombinációja, amelyek fordítási kulcsokkal jelennek meg.
* Az animációk a szövegekre és képekre egyaránt alkalmazhatók.

**3. Webshop szekció**

* A divcontainer IV osztályú szekció a webshopot népszerűsíti.
* Tartalmazza a termékek előnyeit felsoroló szövegeket és egy képet.
* Az animációk a szövegekre (good1, good2, good3) és a képre (shop) külön kerülnek alkalmazásra.

**4. Gyors szállítás szekció**

* A divcontainer III osztályú szekció a gyors és biztonságos szállítást emeli ki.
* Szövegek és képek kombinációja, amelyek fordítási kulcsokkal jelennek meg.
* Az animációk a szövegekre (divcontainer\_III\_text) és a képekre külön kerülnek alkalmazásra.

**5. Bejelentkezés szekció**

* A divcontainer V osztályú szekció tartalmaz egy gombot, amely a bejelentkezési oldalra navigál.
* A gomb fordítási kulccsal jelenik meg, és az aktuális nyelv (currentLang) alapján állítja be a navigációs útvonalat.
* Az animációk a gombra és a környező elemekre (divcontainer\_V\_div\_block, divcontainer\_V\_div) kerülnek alkalmazásra.

**6. Kapcsolat szekció**

* A divcontainer VI osztályú szekció tartalmazza a kapcsolatfelvételi információkat, például telefonszámot, e-mail címet és címet.
* A szekció tartalmaz egy logót és közösségi média ikonokat is.
* Az ikonok linkekkel vannak ellátva, amelyek külső oldalakra mutatnak.

**Reszponzív elemek**

* A sablon elemei reszponzívak, és különböző képernyőméretekhez igazodnak.
* A képek és szövegek megfelelően méreteződnek, hogy mobil eszközökön is jól jelenjenek meg.

**Animációk**

* Az animációk a #child referencia változóval ellátott elemekhez kapcsolódnak.
* Az IntersectionObserver API figyeli az elemek megjelenését a nézetben, és az osztályok alapján különböző animációkat alkalmaz.

**Fordítási kulcsok**

A fordítási kulcsokhoz szükséges, hogy a megfelelő nyelvi fájlok (pl. en.json, hu.json) definiálva legyenek az alkalmazásban.

**Közösségi média ikonok bővítése**

* Új ikonokat lehet hozzáadni a divcontainer VI szekcióhoz, például LinkedIn vagy YouTube linkeket.

## LoginComponent

A **LoginComponent** felelős a felhasználók bejelentkezési folyamatának kezeléséért. A komponens ellenőrzi a felhasználó által megadott adatokat, hitelesíti azokat a szerverrel, és sikeres bejelentkezés esetén elmenti a szükséges adatokat a [localStorage](about:blank)-ba. Emellett támogatja a többnyelvűséget az @ngx-translate/core könyvtár segítségével.

**Főbb funkciók**

**Bejelentkezés**

* Ellenőrzi a felhasználó által megadott e-mail címet és jelszót.
* Hibaüzeneteket jelenít meg, ha az adatok hiányosak vagy helytelenek.
* Sikeres hitelesítés esetén elmenti a felhasználói tokent és e-mail címet a [localStorage](about:blank)-ba.
* Átirányítja a felhasználót a kezdőoldalra vagy egy megadott URL-re.

**Felhasználói élmény javítása**

* Automatikusan kitölti az e-mail mezőt, ha korábban elmentett e-mail cím található a [localStorage](about:blank)-ban.
* Betöltés közben egy [loading](about:blank) állapotot kezel, amely vizuális visszajelzést ad a felhasználónak.

**Konstruktor**

A konstruktor figyeli az URL paramétereket az [ActivatedRoute](about:blank) segítségével. Ha a [lang](about:blank) paraméter elérhető, frissíti az aktuális nyelvet, és meghívja a[TranslateService](about:blank) megfelelő metódusát a nyelv beállításához.

**ngOnInit**

Az [ngOnInit](about:blank) metódus inicializálja a komponenst, és ellenőrzi, hogy van-e elmentett e-mail cím a [localStorage](about:blank)-ban. Ha talál, automatikusan kitölti az e-mail mezőt.

**Funkciók**

* Ellenőrzi a [localStorage](about:blank)-ban tárolt userEmail értéket.
* Ha található mentett e-mail cím, azt beállítja az [email](about:blank) mező értékeként.

**login**

A [login](about:blank) metódus felelős a bejelentkezési folyamat kezeléséért. Ellenőrzi a felhasználó által megadott adatokat, és hitelesíti azokat a szerverrel.

**Funkciók**

1. **Hibakezelés**:
   * Ellenőrzi, hogy az e-mail és jelszó mezők nincsenek-e üresen.
   * Ha bármelyik mező üres, a megfelelő hibaüzenetet állítja be ([emailError](about:blank), [passwordError](about:blank)).
2. **API hívás**:
   * Ha az adatok helyesek, HTTP POST kérést küld a szervernek a hitelesítéshez.
   * A kérés tartalmazza az e-mail címet és a jelszót.
3. **Sikeres bejelentkezés**:
   * Elmenti a szerver által visszaadott tokent a [localStorage](about:blank)-ba.
   * Elmenti az e-mail címet a [localStorage](about:blank)-ba.
   * Átirányítja a felhasználót a kezdőoldalra vagy egy megadott URL-re.
4. **Hibás bejelentkezés**:
   * Ha a hitelesítés sikertelen, beállítja a [loginError](about:blank) értékét, és leállítja a betöltési állapotot.

**resetErrors**

A [resetErrors](about:blank) metódus alaphelyzetbe állítja a hibaüzeneteket.

**Funkciók**

* Visszaállítja az [emailError](about:blank), passwordError és [loginError](about:blank) értékét false-ra.

**username**

* A felhasználónevet tárolja. Jelenleg nem használ konkrét funkciót.

**email**

* A felhasználó által megadott e-mail címet tárolja.

**password**

* A felhasználó által megadott jelszót tárolja.

**emailError**

* Logikai érték, amely jelzi, hogy az e-mail mező üres vagy helytelen.

**passwordError**

* Logikai érték, amely jelzi, hogy a jelszó mező üres vagy helytelen.

**loginError**

* Logikai érték, amely jelzi, hogy a bejelentkezés sikertelen volt.

**loading**

* Logikai érték, amely jelzi, hogy a bejelentkezési folyamat folyamatban van.

**currentLang**

* Az aktuális nyelvet tárolja. Alapértelmezett értéke en (angol).

**Fordítási kulcsok**

A komponensben használt szövegek fordítási kulcsokkal jelennek meg, például:

* "LOGIN.EMAIL" | translate
* "LOGIN.PASSWORD" | translate
* "LOGIN.BUTTON" | translate

Ezek a kulcsok az @ngx-translate/core könyvtár konfigurációjában definiált fordítási fájlokból kerülnek betöltésre.

**API integráció**

A komponens az [ApiService](about:blank) szolgáltatást használja a bejelentkezési API hívásokhoz. Az [ApiService](about:blank)-ben definiált [loginUser](about:blank) metódus végzi a hitelesítést.

## LoginComponentHTML

A **LoginComponent** HTML sablonja a bejelentkezési oldal vizuális megjelenítéséért felel. A sablon tartalmazza a bejelentkezési űrlapot, a hibaüzeneteket, a betöltési állapotot, valamint egy regisztrációs linket. A többnyelvű támogatás az @ngx-translate/core könyvtár segítségével valósul meg.

**Főbb szekciók**

**1. Bejelentkezési fejléc**

* A fejléc (<h1>) tartalmazza a "Login" szöveget, amely fordítási kulccsal jelenik meg.
* A translate pipe biztosítja a többnyelvű támogatást.

**2. Beviteli mezők**

* Az input-div osztályú szekció tartalmazza az e-mail és jelszó mezőket.
* Az e-mail mező:
  + type="email" attribútummal rendelkezik a formátum ellenőrzéséhez.
  + A [(ngModel)] adatkö binding segítségével kapcsolódik az email változóhoz.
* A jelszó mező:
  + type="password" attribútummal rendelkezik a biztonságos bevitelhez.
  + A maxlength="10" attribútum korlátozza a jelszó hosszát.
  + A [(ngModel)] adatkö binding segítségével kapcsolódik a password változóhoz.

**3. Hibaüzenetek**

* A hibaüzenetek feltételesen jelennek meg az \*ngIf direktíva segítségével:
  + emailError: Ha az e-mail mező üres.
  + passwordError: Ha a jelszó mező üres.
  + loginError: Ha a bejelentkezés sikertelen.
* A hibaüzenetek szövegei fordítási kulcsokkal jelennek meg.

**4. Bejelentkezési gomb**

* A login-btn-div osztályú szekció tartalmazza a bejelentkezési gombot.
* A gomb:
  + A login() metódust hívja meg a click eseményre.
  + A szöveg fordítási kulccsal jelenik meg.

**5. Regisztrációs link**

* A register-link osztályú szekció tartalmaz egy linket a regisztrációs oldalra.
* A link:
  + Az @angular/router routerLink direktíváját használja a navigációhoz.
  + A fordítási kulcsok segítségével többnyelvű szöveget jelenít meg.

**6. Betöltési állapot**

* A Loading osztályú szekció tartalmazza a betöltési animációt.
* Az app-loading komponens jelenik meg, ha a loading változó értéke true.

**Reszponzív elemek**

* A sablon elemei reszponzívak, és különböző képernyőméretekhez igazodnak.
* A beviteli mezők és a gombok megfelelően méreteződnek, hogy mobil eszközökön is jól jelenjenek meg.

**Animációk és betöltési állapot**

* A betöltési állapotot az app-loading komponens jeleníti meg.
* Az \*ngIf direktíva biztosítja, hogy a betöltési animáció csak akkor jelenjen meg, ha a loading változó értéke true.

**Fordítási kulcsok**

A fordítási kulcsokhoz szükséges, hogy a megfelelő nyelvi fájlok (pl. en.json, hu.json) definiálva legyenek az alkalmazásban.

## RegisterComponent

A **RegisterComponent** felelős az új felhasználók regisztrációs folyamatának kezeléséért. A komponens több lépésben ellenőrzi a felhasználó által megadott adatokat, és a sikeres regisztráció után átirányítja a felhasználót a bejelentkezési oldalra. A többnyelvű támogatás az @ngx-translate/core könyvtár segítségével valósul meg.

**Főbb funkciók**

**Többlépéses regisztráció**

* A regisztráció három lépésben történik:
  1. Felhasználónév megadása.
  2. E-mail cím megadása.
  3. Jelszó és jelszó megerősítés megadása.
* Minden lépésnél ellenőrzi a megadott adatok helyességét.

**Validáció**

* Ellenőrzi, hogy a felhasználónév, e-mail cím és jelszó mezők nincsenek-e üresen.
* Az e-mail cím formátumát reguláris kifejezéssel ellenőrzi.
* A jelszó hosszát és formátumát validálja:
  + Minimum 6, maximum 10 karakter.
  + Nem tartalmazhat tiltott írásjeleket.
  + A jelszónak és a megerősítő jelszónak egyeznie kell.

**Regisztrációs API hívás**

* A regisztrációs adatok HTTP POST kéréssel kerülnek elküldésre a szervernek.
* Sikeres regisztráció esetén a felhasználó átirányításra kerül a bejelentkezési oldalra.

**Konstruktor**

A konstruktor figyeli az URL paramétereket az [ActivatedRoute](about:blank) segítségével. Ha a [lang](about:blank) paraméter elérhető, frissíti az aktuális nyelvet, és meghívja a [TranslateService](about:blank) megfelelő metódusát a nyelv beállításához.

**ngOnInit**

Az [ngOnInit](about:blank) metódus inicializálja a komponenst, és átirányítja a felhasználót a regisztrációs oldalra az aktuális nyelv alapján.

**Funkciók**

* Navigáció az aktuális nyelvnek megfelelő regisztrációs oldalra.

**nextStep**

A [nextStep](about:blank) metódus kezeli a regisztrációs folyamat lépéseit. Minden lépésnél ellenőrzi a megadott adatok helyességét, és ha minden adat megfelelő, továbblép a következő lépésre.

**Funkciók**

1. **Felhasználónév ellenőrzése**:
   * Ellenőrzi, hogy a felhasználónév mező nem üres-e.
   * Ha üres, beállítja a [usernameError](about:blank) értékét.
2. **E-mail cím ellenőrzése**:
   * Ellenőrzi, hogy az e-mail mező nem üres-e.
   * Az e-mail formátumát reguláris kifejezéssel ellenőrzi.
   * Ha az e-mail formátuma hibás, beállítja az [emailFormatError](about:blank) értékét.
3. **Jelszó ellenőrzése**:
   * Ellenőrzi, hogy a jelszó mező nem üres-e.
   * Ellenőrzi a jelszó hosszát (minimum 6, maximum 10 karakter).
   * Ellenőrzi, hogy a jelszó és a megerősítő jelszó egyezik-e.
   * Ellenőrzi, hogy a jelszó nem tartalmaz tiltott írásjeleket.
4. **Regisztráció indítása**:
   * Ha minden adat helyes, meghívja a [registerUser](about:blank) metódust.

**previousStep**

A [previousStep](about:blank) metódus visszalép az előző lépésre, ha a felhasználó a regisztrációs folyamat során vissza szeretne lépni.

**resetErrors**

A [resetErrors](about:blank) metódus alaphelyzetbe állítja az összes hibaüzenetet.

**Funkciók**

* Visszaállítja az összes hibaüzenet ([usernameError](about:blank), [emailError](about:blank), stb.) értékét false-ra.

**isEmailValid**

**Leírás**

Az [isEmailValid](about:blank) metódus ellenőrzi, hogy az e-mail cím formátuma helyes-e.

**Funkciók**

* Reguláris kifejezéssel ellenőrzi az e-mail cím formátumát.

**containsPunctuation**

A [containsPunctuation](about:blank) metódus ellenőrzi, hogy a jelszó tartalmaz-e tiltott írásjeleket.

* Reguláris kifejezéssel ellenőrzi a jelszóban található írásjeleket.

**registerUser**

A [registerUser](about:blank) metódus HTTP POST kérést küld a szervernek a regisztrációs adatokkal.

**Funkciók**

1. **Adatok elküldése**:
   * A felhasználónév, e-mail cím, jelszó és jelszó megerősítés kerül elküldésre.
2. **Sikeres regisztráció**:
   * Átirányítja a felhasználót a bejelentkezési oldalra.
3. **Hibás regisztráció**:
   * Beállítja a [registerError](about:blank) értékét, és visszaállítja a betöltési állapotot.

**Tulajdonságok**

**currentStep**

* A regisztráció aktuális lépését tárolja.

**username, email, password, confirmPassword**

* A felhasználó által megadott adatokat tárolják.

**usernameError, emailError, emailFormatError, stb.**

* Logikai értékek, amelyek jelzik, hogy az adott mező hibás vagy hiányos.

**loading, unloading**

* A betöltési állapotot kezelik.

**currentLang**

* Az aktuális nyelvet tárolja.

## RegisterComponentHTML

A **RegisterComponent** HTML sablonja egy többlépéses regisztrációs folyamatot valósít meg. A sablon három lépésben gyűjti be a felhasználó adatait: felhasználónév, e-mail cím, valamint jelszó és jelszó megerősítés. A többnyelvű támogatás az @ngx-translate/core könyvtár segítségével valósul meg, és a felhasználói élményt hibaüzenetek, animációk és betöltési állapot javítják.

**Főbb szekciók**

**1. Regisztrációs fejléc**

* A fejléc (<h1>) tartalmazza a "Register" szöveget, amely fordítási kulccsal jelenik meg.
* A translate pipe biztosítja a többnyelvű támogatást.

**2. Beviteli mezők**

A beviteli mezők három lépésben jelennek meg, a currentStep változó értékétől függően.

**Felhasználónév mező (1. lépés)**

* Az input-div-1 osztályú szekció tartalmazza a felhasználónév mezőt.
* A mező:
  + type="text" attribútummal rendelkezik.
  + A [(ngModel)] adatkö binding segítségével kapcsolódik a username változóhoz.
  + A maxlength="20" attribútum korlátozza a felhasználónév hosszát.
* Hibaüzenet:
  + Ha a mező üres, a usernameError értéke true, és megjelenik a hibaüzenet.

**E-mail mező (2. lépés)**

* Az input-div-2 osztályú szekció tartalmazza az e-mail mezőt.
* A mező:
  + type="email" attribútummal rendelkezik a formátum ellenőrzéséhez.
  + A [(ngModel)] adatkö binding segítségével kapcsolódik az email változóhoz.
* Hibaüzenetek:
  + Ha a mező üres, a emailError értéke true.
  + Ha az e-mail formátuma hibás, a emailFormatError értéke true.

**Jelszó és jelszó megerősítés mezők (3. lépés)**

* Az input-div-3 osztályú szekció tartalmazza a jelszó és jelszó megerősítés mezőket.
* A mezők:
  + type="password" attribútummal rendelkeznek a biztonságos bevitelhez.
  + A [(ngModel)] adatkö binding segítségével kapcsolódnak a password és confirmPassword változókhoz.
  + A maxlength="10" attribútum korlátozza a jelszó hosszát.
* Hibaüzenetek:
  + Ha a mezők üresek, a passwordnullError értéke true.
  + Ha a jelszavak nem egyeznek, a passwordError értéke true.
  + Ha a jelszó tiltott írásjeleket tartalmaz, a passwordFormatError értéke true.
  + Ha a jelszó túl rövid, a passwordMinError értéke true.
  + Ha a jelszó túl hosszú, a passwordMaxError értéke true.
  + Ha a regisztráció sikertelen (pl. az e-mail cím már létezik), a registerError értéke true.

**3. Betöltési állapot**

* A Loading osztályú szekció tartalmazza a betöltési animációt.
* Az app-loading komponens jelenik meg, ha a loading változó értéke true.

**4. Navigációs gombok**

* A register-btn-div osztályú szekció tartalmazza a "Back" és "Next" gombokat.
* A gombok:
  + A previousStep() és nextStep() metódusokat hívják meg a click eseményre.
  + A szövegek fordítási kulcsokkal jelennek meg.

Ezek a kulcsok az @ngx-translate/core könyvtár konfigurációjában definiált fordítási fájlokból kerülnek betöltésre.

**Reszponzív elemek**

* A sablon elemei reszponzívak, és különböző képernyőméretekhez igazodnak.
* Az animációk (fade-in osztály) biztosítják a vizuális átmenetet a lépések között.

**Animációk és betöltési állapot**

* Az \*ngIf direktíva biztosítja, hogy a megfelelő mezők és gombok csak a megfelelő lépésnél jelenjenek meg.
* A betöltési animáció (app-loading komponens) csak akkor jelenik meg, ha a loading változó értéke true.

**CSS osztályok**

A sablonban használt osztályokhoz szükséges, hogy a megfelelő CSS stílusok definiálva legyenek a register.component.css fájlban. Példák:

* body
* register
* input-div
* input-div-1, input-div-2, input-div-3
* error
* register-btn-div
* Loading

## WebshopPageComponent

A **WebshopPageComponent** felelős a webshop termékeinek megjelenítéséért, szerkesztéséért és törléséért. A komponens a termékek betöltését az API-n keresztül végzi, és lehetőséget biztosít a termékek módosítására egy szerkesztési párbeszédablak segítségével. A felhasználói szerepkör alapján különböző funkciók érhetők el.

**Főbb funkciók**

**Termékek betöltése**

* Az API-n keresztül lekéri a termékek listáját.
* A termékek adatai a [products](about:blank) tömbben kerülnek tárolásra.

**Termék törlése**

* Egy termék törlésére szolgáló funkció, amely megerősítést kér a felhasználótól.
* A törlés után a termék eltávolításra kerül a helyi listából.

**Termék szerkesztése**

* Egy termék szerkesztésére szolgáló funkció, amely egy párbeszédablakot nyit meg.
* A párbeszédablakban módosított adatokat az API-n keresztül frissíti, és a helyi listában is frissíti a terméket.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például a termékek kezeléséhez, a navigációhoz és a párbeszédablakok megnyitásához.

**Paraméterek**

* [productService](about:blank): A termékek kezeléséhez szükséges szolgáltatás.
* [router](about:blank): Navigációs szolgáltatás az útvonalak kezeléséhez.
* [userService](about:blank): A felhasználói szerepkör kezeléséhez szükséges szolgáltatás.
* [dialog](about:blank): A párbeszédablakok megnyitásához szükséges Angular Material szolgáltatás.

**ngOnInit**

Az [ngOnInit](about:blank) metódus inicializálja a komponenst, és meghívja a termékek betöltéséért felelős [loadProducts](about:blank) metódust.

**loadProducts**

A [loadProducts](about:blank) metódus az API-n keresztül lekéri a termékek listáját, és eltárolja azokat a [products](about:blank) tömbben.

**Funkciók**

1. **API hívás**:
   * A [productService.getProducts()](about:blank) metódus segítségével lekéri a termékeket.
2. **Sikeres válasz**:
   * A termékek listája a [products](about:blank) tömbben kerül tárolásra.
3. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**deleteProduct**

A [deleteProduct](about:blank) metódus egy termék törlésére szolgál. A törlés előtt megerősítést kér a felhasználótól.

**Funkciók**

1. **Megerősítés**:
   * A [confirm](about:blank) függvény segítségével megerősítést kér a felhasználótól.
2. **API hívás**:
   * A [productService.deleteProduct()](about:blank) metódus segítségével törli a terméket az API-n keresztül.
3. **Helyi lista frissítése**:
   * A törölt termék eltávolításra kerül a [products](about:blank) tömbből.
4. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**editProduct**

Az [editProduct](about:blank) metódus egy termék szerkesztésére szolgál. Egy párbeszédablakot nyit meg, amelyben a felhasználó módosíthatja a termék adatait.

**Funkciók**

1. **Termék keresése**:
   * A [products](about:blank) tömbben megkeresi a szerkesztendő terméket.
2. **Párbeszédablak megnyitása**:
   * Az [EditProductDialogComponent](about:blank) párbeszédablakot nyitja meg, és átadja a termék adatait.
3. **Frissített adatok kezelése**:
   * A párbeszédablak bezárása után ellenőrzi, hogy történt-e módosítás.
   * Az API-n keresztül frissíti a terméket a [productService.updateProduct()](about:blank) metódus segítségével.
   * A helyi listában is frissíti a termék adatait.
4. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**Tulajdonságok**

**userRole**

* A felhasználó szerepkörét tárolja. Alapértelmezett értéke null.

**products**

* A termékek listáját tárolja.
* Új párbeszédablakok, például termékek részletes megtekintése, hozzáadhatók a komponenshez.

## WebshopPageComponentHTML

A **WebshopPageComponent** HTML sablonja a webshop termékeinek megjelenítéséért felel. A sablon egy rácsos elrendezést használ, amelyben a termékek dinamikusan jelennek meg az API-ból betöltött adatok alapján. Minden termék tartalmaz egy képet, nevet és árat, valamint egy linket a termék részletes oldalára.

**Főbb szekciók**

**1. Termékek rácsos elrendezése**

* A product-grid osztályú konténer biztosítja a rácsos elrendezést.
* Az Angular \*ngFor direktívája iterál a products tömbön, és minden termékhez létrehoz egy product-item elemet.

**2. Termékek megjelenítése**

* Minden termék egy product-item osztályú div-ben jelenik meg.
* A termékek tartalmazzák:
  + **Kép**:
    - Az img elem dinamikusan tölti be a termék képét az imageURL alapján.
    - Az \*ngIf direktíva biztosítja, hogy a kép csak akkor jelenjen meg, ha az imageURL elérhető.
    - Az alt attribútum a termék nevét jeleníti meg alternatív szövegként.
  + **Név**:
    - A termék neve a name mezőből kerül megjelenítésre.
  + **Ár**:
    - Az ár a price mezőből kerül megjelenítésre, és az Angular currency pipe segítségével formázódik.

**3. Termék részletek link**

* Az a elem a termék részletes oldalára navigál a routerLink direktíva segítségével.
* A routerLink dinamikusan generálja az URL-t a termék id mezője alapján.

**Reszponzív elemek**

* A product-grid és product-item osztályok biztosítják a reszponzív elrendezést.
* A termékek rácsos elrendezése automatikusan igazodik a képernyő méretéhez.

**Angular direktívák**

* **\*ngFor**:
  + Iterál a products tömbön, és minden termékhez létrehoz egy HTML elemet.
* **\*ngIf**:
  + Ellenőrzi, hogy a termék rendelkezik-e képpel (imageURL), és csak akkor jeleníti meg a képet.

**Reszponzív kialakítás**

* A product-grid osztályhoz további CSS szabályok adhatók a különböző képernyőméretekhez való jobb igazodás érdekében.

**Fordítási kulcsok hozzáadása**

* A statikus szövegek (pl. "Price") fordítási kulcsokkal helyettesíthetők az @ngx-translate/core könyvtár segítségével.

## AddProductComponent

Az AddProductComponent felelős új termékek hozzáadásáért, meglévő termékek szerkesztéséért és megtekintéséért. A komponens lehetőséget biztosít a termékek adatainak feltöltésére, képek csatolására, valamint a termékek listájának betöltésére és frissítésére. A párbeszédablakok segítségével a termékek részletei és szerkesztési funkciói is elérhetők.

Főbb funkciók

Termék hozzáadása

* A felhasználó megadhatja a termék nevét, leírását, árát és feltölthet egy képet.
* Az adatok egy [FormData](about:blank) objektumba kerülnek, amelyet az API-n keresztül küld el a szervernek.
* Sikeres hozzáadás után a termékek listája frissül.

**Termékek betöltése**

* Az API-n keresztül lekéri a termékek listáját, és eltárolja azokat a [products](about:blank) tömbben.

**Termék szerkesztése**

* Egy termék szerkesztésére szolgáló funkció, amely egy párbeszédablakot nyit meg.
* A párbeszédablakban módosított adatokat az API-n keresztül frissíti, és a helyi listában is frissíti a terméket.

**Termék részleteinek megtekintése**

* Egy termék részleteit megjelenítő párbeszédablakot nyit meg.
* A párbeszédablakban lehetőség van a termék frissítésére vagy törlésére.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például a termékek kezeléséhez, a navigációhoz és a párbeszédablakok megnyitásához.

**Paraméterek**

* [productService](about:blank): A termékek kezeléséhez szükséges szolgáltatás.
* [router](about:blank): Navigációs szolgáltatás az útvonalak kezeléséhez.
* [userService](about:blank): A felhasználói szerepkör kezeléséhez szükséges szolgáltatás.
* [dialog](about:blank): A párbeszédablakok megnyitásához szükséges Angular Material szolgáltatás.

**ngOnInit**

Az [ngOnInit](about:blank) metódus inicializálja a komponenst, és meghívja a termékek betöltéséért felelős [loadProducts](about:blank) metódust.

**onFileSelected**

Az [onFileSelected](about:blank) metódus kezeli a fájl kiválasztását a termék képének feltöltéséhez.

**Funkciók**

* Ellenőrzi, hogy a felhasználó kiválasztott-e egy fájlt.
* A kiválasztott fájl a [selectedFile](about:blank) változóban kerül tárolásra.

**onSubmit**

Az [onSubmit](about:blank) metódus kezeli az új termék hozzáadását.

1. Adatok előkészítése:
   * Az űrlap adatai egy [FormData](about:blank) objektumba kerülnek.
   * A termék neve, leírása, ára és a kiválasztott kép kerül feltöltésre.
2. API hívás:
   * Az [addProduct](about:blank) metódus segítségével elküldi az adatokat az API-nak.
3. Sikeres hozzáadás:
   * A termékek listája frissül.
   * Az űrlap mezői alaphelyzetbe állnak.
   * Egy sikerüzenet jelenik meg, amely 3 másodperc után eltűnik.
4. Hibakezelés:

Hiba esetén a konzolban megjelenik a hibaüzenet**loadProducts**

A [loadProducts](about:blank) metódus az API-n keresztül lekéri a termékek listáját, és eltárolja azokat a [products](about:blank) tömbben.

**Funkciók**

1. API hívás:
   * A [productService.getProducts()](about:blank) metódus segítségével lekéri a termékeket.
2. Sikeres válasz:
   * A termékek listája a [products](about:blank) tömbben kerül tárolásra.
3. Hibakezelés:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**editProduct**

Az [editProduct](about:blank) metódus egy termék szerkesztésére szolgál. Egy párbeszédablakot nyit meg, amelyben a felhasználó módosíthatja a termék adatait.

1. Termék keresése:
   * A [products](about:blank) tömbben megkeresi a szerkesztendő terméket.

**Frissített adatok kezelése:**

* + Az API-n keresztül frissíti a terméket a [productService.updateProduct()](about:blank) metódus segítségével.
  + A helyi listában is frissíti a termék adatait.

1. **Hibakezelés:**
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**openProductDialog**

Az [openProductDialog](about:blank) metódus egy termék részleteit megjelenítő párbeszédablakot nyit meg.

**Funkciók**

1. Párbeszédablak megnyitása:
   * A [ProductDetailsDialogComponent](about:blank) párbeszédablakot nyitja meg, és átadja a termék adatait.
2. Frissített adatok kezelése:
   * Ha a termék frissült vagy törlésre került, a termékek listája újratöltődik.

**addproducForm**

* Logikai érték, amely jelzi, hogy a termék hozzáadási űrlap meg van-e nyitva.

userRole

* A felhasználó szerepkörét tárolja. Alapértelmezett értéke null.

products

* A termékek listáját tárolja.

product

* Az új termék adatait tárolja.

selectedFile

* A kiválasztott fájlt tárolja a kép feltöltéséhez.

successMessage

* Logikai érték, amely jelzi, hogy a termék sikeresen hozzáadásra került.
* A [userRole](about:blank) alapján további jogosultságokat lehet implementálni, például csak adminisztrátorok számára elérhető funkciókat.

## AddProductComponentHTML

Az **AddProductComponent** HTML sablonja két fő funkciót valósít meg: a termékek listázását és új termékek hozzáadását. A sablon lehetőséget biztosít a termékek megtekintésére, valamint egy űrlap segítségével új termékek hozzáadására, beleértve a képek feltöltését is. A felhasználói élményt fordítási kulcsok és sikerüzenetek javítják.

**Főbb szekciók**

**1. Termékek listázása**

* A product-grid osztályú konténer biztosítja a termékek rácsos elrendezését.
* Az Angular \*ngFor direktívája iterál a products tömbön, és minden termékhez létrehoz egy product-item elemet.
* A termékek tartalmazzák:
  + **Kép**:
    - Az img elem dinamikusan tölti be a termék képét az imageURL alapján.
    - Az \*ngIf direktíva biztosítja, hogy a kép csak akkor jelenjen meg, ha az imageURL elérhető.
  + **Név**:
    - A termék neve a name mezőből kerül megjelenítésre.
  + **Ár**:
    - Az ár a price mezőből kerül megjelenítésre, és "Ft" formátumban jelenik meg.
* A termékek kattinthatók, és a openProductDialog(product) metódust hívják meg.

**2. Új termék hozzáadása gomb**

* Az add-product-btn osztályú gomb lehetővé teszi az új termék hozzáadási űrlap megnyitását.
* A gomb szövege fordítási kulccsal jelenik meg: "add product" | translate.

**3. Új termék hozzáadási űrlap**

* Az addproduc osztályú konténer tartalmazza az új termék hozzáadására szolgáló űrlapot.
* Az űrlap csak akkor jelenik meg, ha az addproducForm értéke true.
* Az űrlap mezői:
  + **Név**:
    - type="text" mező, amely a termék nevét tárolja.
    - A [(ngModel)] adatkö binding segítségével kapcsolódik a product.name változóhoz.
  + **Leírás**:
    - textarea mező, amely a termék leírását tárolja.
    - A [(ngModel)] adatkö binding segítségével kapcsolódik a product.description változóhoz.
  + **Ár**:
    - type="number" mező, amely a termék árát tárolja.
    - A [(ngModel)] adatkö binding segítségével kapcsolódik a product.price változóhoz.
  + **Kép**:
    - type="file" mező, amely lehetővé teszi a kép feltöltését.
    - Az onFileSelected($event) metódust hívja meg a fájl kiválasztásakor.
* **Gombok**:
  + **Hozzáadás**:
    - Az űrlap elküldésére szolgál, és az onSubmit() metódust hívja meg.
  + **Mégse**:
    - Bezárja az űrlapot, és visszaállítja az addproducForm értékét false-ra.

**4. Sikerüzenet**

* Az success-message osztályú div jeleníti meg a sikeres termékhozzáadás üzenetét.
* Az üzenet csak akkor jelenik meg, ha a successMessage értéke true.

**Reszponzív elemek**

* A product-grid és product-item osztályok biztosítják a reszponzív elrendezést.
* Az űrlap és a termékek listája külön szekcióban jelenik meg, és a képernyő méretéhez igazodik.

**Használat**

**Angular direktívák**

* **\*ngFor**:
  + Iterál a products tömbön, és minden termékhez létrehoz egy HTML elemet.
* **\*ngIf**:
  + Ellenőrzi, hogy az addproducForm vagy a successMessage értéke true, és ennek megfelelően jeleníti meg az elemeket.

**Reszponzív kialakítás**

* További CSS szabályok adhatók a különböző képernyőméretekhez való jobb igazodás érdekében.

## CartPageComponent

A **CartPageComponent** felelős a vásárlói kosár kezeléséért. A komponens betölti a kosár tartalmát a backendből, lehetőséget biztosít a termékek hozzáadására, eltávolítására, mennyiségük csökkentésére, valamint a teljes ár kiszámítására. A többnyelvű támogatás az @ngx-translate/core könyvtár segítségével valósul meg.

**Kosár betöltése**

* Az API-n keresztül lekéri a kosár tartalmát, és eltárolja azokat a [cart](about:blank) tömbben.
* A termékek mennyisége és ára alapján kiszámítja a teljes árat.

**Termék hozzáadása a kosárhoz**

* Egy termék hozzáadására szolgáló funkció, amely az API-n keresztül frissíti a kosár tartalmát.

**Termék eltávolítása a kosárból**

* Egy termék eltávolítására szolgáló funkció, amely az API-n keresztül frissíti a kosár tartalmát.

**Termék mennyiségének csökkentése**

* Egy termék mennyiségének csökkentésére szolgáló funkció, amely az API-n keresztül frissíti a kosár tartalmát.

**Teljes ár kiszámítása**

* A kosárban lévő termékek mennyisége és ára alapján kiszámítja a teljes árat.

**Nyelvi támogatás**

* Az URL paraméterekből (lang) olvassa ki az aktuális nyelvet.
* Az @ngx-translate/core könyvtár segítségével állítja be a nyelvet.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például a kosár kezeléséhez, a fordításokhoz és az útvonalak figyeléséhez.

**Paraméterek**

* [shoppingCartService](about:blank): A kosár kezeléséhez szükséges szolgáltatás.
* [translate](about:blank): A fordítások kezeléséhez szükséges szolgáltatás.
* [route](about:blank): Az aktuális útvonal figyeléséhez szükséges szolgáltatás.

**ngOnInit**

Az [ngOnInit](about:blank) metódus inicializálja a komponenst, és figyeli az URL-ben található nyelvi paramétereket. A nyelv beállítása után betölti a kosár tartalmát.

**Funkciók**

* Az URL paraméterekből kiolvassa a lang értékét, és beállítja az aktuális nyelvet.
* Meghívja a [loadCart](about:blank) metódust a kosár tartalmának betöltésére.

**loadCart**

A [loadCart](about:blank) metódus az API-n keresztül lekéri a kosár tartalmát, és eltárolja azokat a [cart](about:blank) tömbben.

**Funkciók**

1. **API hívás**:
   * A [shoppingCartService.getCartFromBackend()](about:blank) metódus segítségével lekéri a kosár tartalmát.
2. **Sikeres válasz**:
   * A kosár adatai a [cart](about:blank) tömbben kerülnek tárolásra.
   * Meghívja a [calculateTotalPrice](about:blank) metódust a teljes ár kiszámítására.
3. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**addToCart**

Az [addToCart](about:blank) metódus egy termék hozzáadására szolgál a kosárhoz.

**Funkciók**

1. **API hívás**:
   * A [shoppingCartService.addCartItem()](about:blank) metódus segítségével hozzáadja a terméket a kosárhoz.
2. **Sikeres válasz**:
   * A kosár tartalma frissül a [loadCart](about:blank) metódus meghívásával.
3. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**removeFromCart**

A [removeFromCart](about:blank) metódus egy termék eltávolítására szolgál a kosárból.

**Funkciók**

1. **API hívás**:
   * A [shoppingCartService.removeCartItem()](about:blank) metódus segítségével eltávolítja a terméket a kosárból.
2. **Sikeres válasz**:
   * A kosár tartalma frissül a [loadCart](about:blank) metódus meghívásával.
3. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**decreaseFromCart**

A [decreaseFromCart](about:blank) metódus egy termék mennyiségének csökkentésére szolgál a kosárban.

**Funkciók**

1. **API hívás**:
   * A [shoppingCartService.decreaseCartItem()](about:blank) metódus segítségével csökkenti a termék mennyiségét.
2. **Sikeres válasz**:
   * A kosár tartalma frissül a [loadCart](about:blank) metódus meghívásával.
3. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**calculateTotalPrice**

A [calculateTotalPrice](about:blank) metódus kiszámítja a kosárban lévő termékek teljes árát.

**Funkciók**

* A [cart](about:blank) tömbön iterálva összegzi a termékek árát és mennyiségét.

**Tulajdonságok**

**cart**

* A kosár tartalmát tárolja, amely termékekből és azok mennyiségéből áll.

**totalPrice**

* A kosárban lévő termékek teljes árát tárolja.

## CartPageComponentHTML

A **CartPageComponent** HTML sablonja a vásárlói kosár tartalmának megjelenítéséért felel. A sablon dinamikusan jeleníti meg a kosárban lévő termékeket, azok mennyiségét, árát, valamint a teljes összeget. Emellett lehetőséget biztosít a termékek mennyiségének csökkentésére, eltávolítására, és a fizetési folyamat elindítására.

**Főbb szekciók**

**1. Kosár címe**

* A cart-title osztályú fejléc tartalmazza a kosár címét.
* A szöveg fordítási kulccsal jelenik meg: "CART.TITLE" | translate.

**2. Üres kosár üzenet**

* Az empty-cart osztályú div jelenik meg, ha a kosár üres (cart.length === 0).
* Az üzenet fordítási kulccsal jelenik meg: "CART.EMPTY\_MESSAGE" | translate.

**3. Kosár tartalma**

* Az cart-container osztályú div jelenik meg, ha a kosár nem üres (cart.length > 0).
* A kosár tartalma egy listában (ul.product-list) jelenik meg, ahol minden termék egy li.product-item elemként jelenik meg.

**Termékek megjelenítése**

* **Kép**:
  + Az img elem dinamikusan tölti be a termék képét az imageURL alapján.
  + Az \*ngIf direktíva biztosítja, hogy a kép csak akkor jelenjen meg, ha az imageURL elérhető.
* **Név**:
  + A termék neve a product.name mezőből kerül megjelenítésre.
* **Ár**:
  + Az ár a product.price mezőből kerül megjelenítésre, és az Angular number pipe segítségével formázódik.
* **Mennyiség**:
  + A termék mennyisége a quantity mezőből kerül megjelenítésre.
  + A szöveg fordítási kulccsal jelenik meg: "CART.QUANTITY" | translate.

**Termék műveletek**

* **Mennyiség csökkentése**:
  + A decreaseFromCart(item.product.id) metódust hívja meg.
  + A gomb szövege fordítási kulccsal jelenik meg: "CART.DECREASE" | translate.
* **Eltávolítás**:
  + A removeFromCart(item.product.id) metódust hívja meg.
  + A gomb szövege fordítási kulccsal jelenik meg: "CART.REMOVE" | translate.

**4. Teljes ár**

* Az total-price osztályú div jeleníti meg a kosár teljes árát.
* **Címke**:
  + A szöveg fordítási kulccsal jelenik meg: "CART.TOTAL" | translate.
* **Összeg**:
  + A teljes ár a totalPrice változóból kerül megjelenítésre, és az Angular number pipe segítségével formázódik.

**5. Fizetés gomb**

* Az delivery-button osztályú div tartalmazza a fizetési folyamat elindítására szolgáló linket.
* A link a delivery oldalra navigál a routerLink direktíva segítségével.
* A szöveg fordítási kulccsal jelenik meg: "CART.CHECKOUT" | translate.

**Fordítási kulcsok**

A sablonban használt fordítási kulcsok:

* "CART.TITLE"
* "CART.EMPTY\_MESSAGE"
* "CART.QUANTITY"
* "CART.DECREASE"
* "CART.REMOVE"
* "CART.TOTAL"
* "CART.CHECKOUT"

Ezek a kulcsok az @ngx-translate/core könyvtár konfigurációjában definiált fordítási fájlokból kerülnek betöltésre.

**Reszponzív elemek**

* A cart-container és product-list osztályok biztosítják a reszponzív elrendezést.
* A termékek listája és a teljes ár megjelenítése automatikusan igazodik a képernyő méretéhez.

**Angular direktívák**

* **\*ngIf**:
  + Ellenőrzi, hogy a kosár üres-e vagy sem, és ennek megfelelően jeleníti meg az elemeket.
* **\*ngFor**:
  + Iterál a cart tömbön, és minden termékhez létrehoz egy HTML elemet.

## DeliveryPageComponent

A **DeliveryPageComponent** felelős a szállítási adatok kezeléséért. A komponens lehetőséget biztosít a felhasználók számára, hogy megadják szállítási adataikat, amelyeket az API-n keresztül ment a backendbe. Emellett támogatja a többnyelvűséget az @ngx-translate/core könyvtár segítségével, és automatikusan kitölti az e-mail mezőt, ha az elérhető a [localStorage](about:blank)-ban.

**Főbb funkciók**

**Szállítási adatok megadása**

* A felhasználók megadhatják nevüket, e-mail címüket, címüket, városukat, házszámukat, irányítószámukat és telefonszámukat.
* Az űrlap validációt végez a mezők helyességének ellenőrzésére.

**Szállítási adatok mentése**

* Az űrlap elküldésekor az adatok az API-n keresztül mentésre kerülnek.
* Sikeres mentés után a felhasználó visszairányításra kerül a kezdőoldalra.

**Nyelvi támogatás**

* Az URL paraméterekből (lang) olvassa ki az aktuális nyelvet.
* Az @ngx-translate/core könyvtár segítségével állítja be a nyelvet.

**Automatikus e-mail kitöltés**

* Ha a felhasználó e-mail címe elérhető a [localStorage](about:blank)-ban, az automatikusan kitöltésre kerül az e-mail mezőben.

**Input validáció**

* A [validateInput](about:blank) metódus biztosítja, hogy csak betűk, számok és szóközök kerüljenek a mezőkbe.
* A [validateNumberInput](about:blank) metódus biztosítja, hogy csak számok kerüljenek a numerikus mezőkbe, és korlátozza az irányítószámot 4 számjegyre, a telefonszámot pedig 11 számjegyre.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például a fordítások kezeléséhez, az útvonalak figyeléséhez és az API hívásokhoz.

**Paraméterek**

* [router](about:blank): Navigációs szolgáltatás az útvonalak kezeléséhez.
* [http](about:blank): HTTP kliens az API hívásokhoz.
* [translate](about:blank): A fordítások kezeléséhez szükséges szolgáltatás.
* [route](about:blank): Az aktuális útvonal figyeléséhez szükséges szolgáltatás.

**ngOnInit**

Az [ngOnInit](about:blank) metódus inicializálja a komponenst, és figyeli az URL-ben található nyelvi paramétereket. Emellett automatikusan kitölti az e-mail mezőt, ha az elérhető a [localStorage](about:blank)-ban.

**Funkciók**

* Az URL paraméterekből kiolvassa a lang értékét, és beállítja az aktuális nyelvet.
* Ellenőrzi, hogy van-e elmentett e-mail cím a [localStorage](about:blank)-ban, és ha igen, kitölti az e-mail mezőt.

**onSubmit**

Az [onSubmit](about:blank) metódus kezeli az űrlap elküldését.

**Funkciók**

1. **Validáció**:
   * Ellenőrzi, hogy az űrlap érvényes-e.
2. **API hívás**:
   * Az űrlap adatait elküldi az API-nak a [http.post](about:blank) metódus segítségével.
3. **Sikeres mentés**:
   * A sikeres mentés után a felhasználó visszairányításra kerül a kezdőoldalra.
4. **Hibakezelés**:
   * Hiba esetén a konzolban megjelenik a hibaüzenet.

**validateInput**

A validateInput metódus biztosítja, hogy csak betűk, számok és szóközök kerüljenek a mezőkbe.

**Funkciók**

* A nem megengedett karaktereket eltávolítja a mező értékéből.

**validateNumberInput**

A [validateNumberInput](about:blank) metódus biztosítja, hogy csak számok kerüljenek a numerikus mezőkbe.

**Funkciók**

* Csak számokat engedélyez.
* Korlátozza az irányítószámot 4 számjegyre és a telefonszámot 11 számjegyre.

**submitDeliveryInfo**

A [submitDeliveryInfo](about:blank) metódus a szállítási adatok mentését kezeli.

**Funkciók**

* A szállítási adatok megjelenítése a konzolban.
* Egy értesítés jelenik meg a sikeres mentésről.

**Tulajdonságok**

**deliveryInfo**

* A szállítási adatokat tárolja, beleértve a nevet, e-mail címet, címet, várost, házszámot, irányítószámot és telefonszámot.

**formSubmitted**

* Logikai érték, amely jelzi, hogy az űrlap el lett-e küldve.

**currentLang**

* Az aktuális nyelvet tárolja. Alapértelmezett értéken.

## DeliveryPageComponentHTML

A **DeliveryPageComponent** HTML sablonja egy szállítási adatok megadására szolgáló űrlapot tartalmaz. Az űrlap validációval biztosítja, hogy a felhasználó helyes adatokat adjon meg, és lehetőséget biztosít az adatok mentésére. A sablon többnyelvű támogatást nyújt az @ngx-translate/core könyvtár segítségével.

**Főbb szekciók**

**1. Cím**

* A fejléc (<h2>) tartalmazza a szállítási adatok címét.
* A szöveg fordítási kulccsal jelenik meg: "Delivery Information" | translate.

**2. Űrlap**

Az űrlap az Angular ngForm direktíváját használja, és az onSubmit metódust hívja meg az elküldéskor.

**Mezők**

1. **Teljes név**:
   * id="fullName" mező, amely a teljes nevet tárolja.
   * A pattern attribútum biztosítja, hogy csak betűk és szóközök kerüljenek a mezőbe.
   * A validateInput metódus további validációt végez.
   * Hibaüzenet jelenik meg, ha a mező üres vagy érvénytelen: "Full Name is required!" | translate.
2. **E-mail**:
   * id="email" mező, amely az e-mail címet tárolja.
   * A [(ngModel)] adatkö binding segítségével kapcsolódik a deliveryInfo.email változóhoz.
   * Hibaüzenet jelenik meg, ha a mező üres vagy érvénytelen: "Email is required!" | translate.
3. **Cím**:
   * id="address" mező, amely a címet tárolja.
   * A pattern attribútum biztosítja, hogy csak betűk és szóközök kerüljenek a mezőbe.
   * Hibaüzenet jelenik meg, ha a mező üres vagy érvénytelen: "Address is required!" | translate.
4. **Házszám**:
   * id="houseNumber" mező, amely a házszámot tárolja.
   * A validateInput metódus biztosítja, hogy csak érvényes karakterek kerüljenek a mezőbe.
   * Hibaüzenet jelenik meg, ha a mező üres: "House Number is required!" | translate.
5. **Város**:
   * id="city" mező, amely a várost tárolja.
   * A pattern attribútum biztosítja, hogy csak betűk és szóközök kerüljenek a mezőbe.
   * Hibaüzenet jelenik meg, ha a mező üres vagy érvénytelen: "City is required!" | translate.
6. **Irányítószám**:
   * id="postalCode" mező, amely az irányítószámot tárolja.
   * A validateNumberInput metódus biztosítja, hogy csak számok kerüljenek a mezőbe, és korlátozza a hosszát 4 számjegyre.
   * Hibaüzenet jelenik meg, ha a mező üres: "Postal Code is required!" | translate.
7. **Telefonszám**:
   * id="phone" mező, amely a telefonszámot tárolja.
   * A validateNumberInput metódus biztosítja, hogy csak számok kerüljenek a mezőbe, és korlátozza a hosszát 11 számjegyre.
   * Hibaüzenet jelenik meg, ha a mező üres: "Phone Number is required!" | translate.

**3. Mentés gomb**

* A mentés gomb (<button type="submit">) elküldi az űrlapot.
* A szöveg fordítási kulccsal jelenik meg: "Save" | translate.

**4. Sikerüzenet**

* A sikerüzenet (<div class="success-message">) akkor jelenik meg, ha az űrlap sikeresen el lett küldve.
* A szöveg fordítási kulccsal jelenik meg: "Success! Your data has been saved." | translate.

**Reszponzív elemek**

* A form-group osztályok biztosítják az űrlap mezőinek megfelelő elrendezését.
* Az űrlap reszponzív kialakítású, és különböző képernyőméretekhez igazodik.

**Angular direktívák**

* **ngForm**:
  + Az űrlap validációját és kezelését biztosítja.
* **ngModel**:
  + Az űrlap mezőinek adatkö bindingját biztosítja.
* **\*ngIf**:
  + Feltételesen jeleníti meg a hibaüzeneteket és a sikerüzenetet.

## EditProductDialogComponent

Az **EditProductDialogComponent** egy párbeszédablakot biztosít a termékek adatainak szerkesztésére. A komponens egy Angular [FormGroup](about:blank)-ot használ az űrlap kezelésére, amely validációval biztosítja a helyes adatok megadását. A párbeszédablak támogatja a többnyelvűséget az @ngx-translate/core könyvtár segítségével.

**Főbb funkciók**

**Termékadatok szerkesztése**

* Az űrlap lehetővé teszi a termék nevének, árának és leírásának szerkesztését.
* Az adatok validációval ellenőrzöttek:
  + A név mező kötelező.
  + Az ár mező kötelező, és nem lehet negatív.

**Adatok visszaadása**

* Az űrlap elküldésekor a szerkesztett adatok visszakerülnek a hívó komponenshez.

**Konstruktor**

A konstruktor inicializálja az űrlapot és beállítja az alapértelmezett értékeket a kapott adatok alapján.

**Paraméterek**

* [fb](about:blank): Az Angular [FormBuilder](about:blank) szolgáltatás az űrlap létrehozásához.
* [dialogRef](about:blank): A párbeszédablak referencia, amely lehetővé teszi a párbeszédablak bezárását és adatok visszaadását.
* [translate](about:blank): A fordítások kezeléséhez szükséges szolgáltatás.
* [data](about:blank): A párbeszédablaknak átadott adatok, amelyek tartalmazzák a szerkesztendő termék adatait.

**editForm**

* Az űrlapot reprezentáló [FormGroup](about:blank).
* Mezők:
  + [name](about:blank): A termék neve. Kötelező mező.
  + [price](about:blank): A termék ára. Kötelező mező, amely nem lehet negatív.
  + [description](about:blank): A termék leírása. Nem kötelező mező.

**currentLang**

* Az aktuális nyelvet tárolja. Alapértelmezett értéke en.

**Metódusok**

**ngOnInit**

* Beállítja az aktuális nyelvet az @ngx-translate/core segítségével.

**save**

* Ellenőrzi, hogy az űrlap érvényes-e.
* Ha érvényes, a szerkesztett adatokat visszaadja a hívó komponensnek a [dialogRef.close()](about:blank) metódus segítségével.

**close**

* Bezárja a párbeszédablakot anélkül, hogy adatokat adna vissza.

**Validáció**

**Név mező**

* Kötelező ([Validators.required](about:blank)).

**Ár mező**

* Kötelező ([Validators.required](about:blank)).
* Nem lehet negatív ([Validators.min(0)](about:blank)).

**Leírás mező**

* Nem kötelező.

## EditProductDialogComponentHTML

Az **EditProductDialogComponent** HTML sablonja egy párbeszédablakot biztosít a termékek adatainak szerkesztésére. Az Angular Material elemek segítségével modern és felhasználóbarát felületet nyújt. A sablon támogatja a többnyelvűséget az @ngx-translate/core könyvtár segítségével.

**Főbb szekciók**

**1. Cím**

* A párbeszédablak címe a mat-dialog-title direktívával jelenik meg.
* A szöveg fordítási kulccsal jelenik meg: "EDIT\_PRODUCT.TITLE" | translate.

**2. Űrlap**

Az űrlap az Angular FormGroup-ot használja, amely az editForm változóhoz van kötve. Az űrlap elküldésekor a save() metódus hívódik meg.

**Mezők**

1. **Név**:
   * mat-form-field elem tartalmazza.
   * A mat-label címke fordítási kulccsal jelenik meg: "EDIT\_PRODUCT.NAME" | translate.
   * Az input mező az editForm name mezőjéhez van kötve a formControlName direktíva segítségével.
2. **Ár**:
   * mat-form-field elem tartalmazza.
   * A mat-label címke fordítási kulccsal jelenik meg: "EDIT\_PRODUCT.PRICE" | translate.
   * Az input mező type="number" attribútummal rendelkezik, és az editForm price mezőjéhez van kötve.
3. **Leírás**:
   * mat-form-field elem tartalmazza.
   * A mat-label címke fordítási kulccsal jelenik meg: "EDIT\_PRODUCT.DESCRIPTION" | translate.
   * A textarea mező az editForm description mezőjéhez van kötve.

**3. Műveleti gombok**

* A gombok a mat-dialog-actions direktívával vannak csoportosítva.
* **Mégse gomb**:
  + A mat-button stílusú gomb bezárja a párbeszédablakot a close() metódus meghívásával.
  + A szöveg fordítási kulccsal jelenik meg: "EDIT\_PRODUCT.CANCEL" | translate.
* **Mentés gomb**:
  + A mat-raised-button stílusú gomb elküldi az űrlapot.
  + A gomb színe color="primary".
  + A szöveg fordítási kulccsal jelenik meg: "EDIT\_PRODUCT.SAVE" | translate.

**Fordítási kulcsok**

A sablonban használt fordítási kulcsok:

* "EDIT\_PRODUCT.TITLE"
* "EDIT\_PRODUCT.NAME"
* "EDIT\_PRODUCT.PRICE"
* "EDIT\_PRODUCT.DESCRIPTION"
* "EDIT\_PRODUCT.CANCEL"
* "EDIT\_PRODUCT.SAVE"

Ezek a kulcsok az @ngx-translate/core könyvtár konfigurációjában definiált fordítási fájlokból kerülnek betöltésre.

**Reszponzív elemek**

* Az Angular Material mat-form-field elemei automatikusan reszponzívak, és különböző képernyőméretekhez igazodnak.
* A párbeszédablak mérete az Angular Material konfigurációjában állítható be.

**Angular direktívák**

* **formGroup**:
  + Az űrlap az editForm változóhoz van kötve.
* **formControlName**:
  + Az űrlap mezői az editForm megfelelő mezőihez vannak kötve.

**Angular Material elemek**

* **mat-form-field**:
  + Az űrlap mezőinek megjelenítésére szolgál.
* **mat-label**:
  + A mezők címkéit jeleníti meg.
* **mat-dialog-actions**:
  + A műveleti gombok csoportosítására szolgál.
* **mat-button** és **mat-raised-button**:
  + A gombok stílusát biztosítják.

## NavbarComponent

A **NavbarComponent** egy navigációs sávot biztosít az alkalmazás számára. A komponens támogatja a többnyelvűséget, a menü és nyelvváltó legördülő menük kezelését, valamint az útvonalak közötti navigációt. A felhasználói élményt reszponzív kialakítás és dinamikus eseménykezelés javítja.

**Főbb funkciók**

**Menü megnyitása és bezárása**

* A [menuOpen](about:blank) változó vezérli a navigációs menü állapotát.
* A [toggleMenu()](about:blank) metódus váltja a menü nyitott és zárt állapotát.

**Nyelvváltó legördülő menü**

* A [dropdownOpen](about:blank) változó vezérli a nyelvváltó legördülő menü állapotát.
* A [toggleDropdown()](about:blank) metódus váltja a legördülő menü nyitott és zárt állapotát.

**Nyelvváltás**

* A [switchLang(lang: string)](about:blank) metódus váltja az aktuális nyelvet.
* A nyelvváltás frissíti az URL-t, a fordítási szolgáltatást ([TranslateService](about:blank)), és elmenti a választott nyelvet a [localStorage](about:blank)-ba.

**Webshop oldalra navigálás**

* A [switchtoWebShop()](about:blank) metódus navigál a webshop oldalra az aktuális nyelv alapján.

**Kattintás kívülre eseménykezelés**

* Az @HostListener('document:click') dekorátor figyeli a dokumentumon belüli kattintásokat.
* Ha a kattintás a nyelvváltó vagy a menü elemein kívül történik, a legördülő menü és a navigációs menü bezáródik.

**Nyelv inicializálása**

* Az [ngOnInit()](about:blank) metódus ellenőrzi, hogy van-e elmentett nyelv a [localStorage](about:blank)-ban.
* Ha nincs, az alapértelmezett nyelv (en) kerül beállításra, és a felhasználó a kezdőoldalra (home) navigál.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például az útvonalak figyeléséhez, a fordítások kezeléséhez és a navigációhoz.

**Paraméterek**

* [route](about:blank): Az aktuális útvonal figyeléséhez szükséges szolgáltatás.
* [router](about:blank): Navigációs szolgáltatás az útvonalak kezeléséhez.
* [translate](about:blank): A fordítások kezeléséhez szükséges szolgáltatás.

**Tulajdonságok**

**dropdownOpen**

* Logikai érték, amely jelzi, hogy a nyelvváltó legördülő menü nyitva van-e.

**menuOpen**

* Logikai érték, amely jelzi, hogy a navigációs menü nyitva van-e.

**currentLang**

* Az aktuális nyelvet tárolja. Alapértelmezett értéke en.

**Metódusok**

**toggleMenu**

* Váltja a [menuOpen](about:blank) állapotát, ezzel nyitja vagy zárja a navigációs menüt.

**toggleDropdown**

* Váltja a [dropdownOpen](about:blank) állapotát, ezzel nyitja vagy zárja a nyelvváltó legördülő menüt.

**ngOnInit**

* Ellenőrzi, hogy van-e elmentett nyelv a [localStorage](about:blank)-ban.
* Ha van, beállítja az aktuális nyelvet.
* Ha nincs, az alapértelmezett nyelvet (en) állítja be, és a kezdőoldalra navigál.

**switchLang**

* Frissíti az aktuális nyelvet, az URL-t, és elmenti a választott nyelvet a [localStorage](about:blank)-ba.

**switchtoWebShop**

* Navigál a webshop oldalra az aktuális nyelv alapján.

**onClickOutside**

* Bezárja a nyelvváltó legördülő menüt és a navigációs menüt, ha a kattintás a komponensen kívül történik.

## NavbarComponentHTML

A **NavbarComponent** HTML sablonja egy reszponzív navigációs sávot biztosít az alkalmazás számára. A navigációs sáv tartalmazza az alkalmazás logóját, menüpontokat, felhasználói és kosár ikonokat, valamint egy nyelvváltó legördülő menüt. A sablon többnyelvű támogatást nyújt az @ngx-translate/core könyvtár segítségével.

**Főbb szekciók**

**1. Logó**

* Az alkalmazás logója és neve a logo osztályú <a> elemben található.
* A logó kép (logo.png) az assets/images mappából kerül betöltésre.
* A logó a kezdőoldalra (home) navigál az aktuális nyelv alapján.

**2. Menü gomb**

* A menu-toggle osztályú <button> elem egy hamburger menü ikont jelenít meg.
* A gomb a toggleMenu() metódust hívja meg, amely nyitja vagy zárja a navigációs menüt

**3. Navigációs menüpontok**

* A nav-links osztályú <div> tartalmazza a navigációs menüpontokat.
* A menuOpen változó alapján a open osztály hozzáadódik, amely a menü megjelenítését vezérli.
* Menüpontok:
  + **Home**: A kezdőoldalra navigál.
  + **Service**: A szolgáltatások oldalra navigál.
  + **Árlista**: Az árlista oldalra navigál.
  + **Web Shop**: A webshop oldalra navigál.
* A menüpontok szövege fordítási kulccsal jelenik meg, például: "Home" | translate.

**4. Felhasználói ikon**

* A user\_a osztályú <a> elem tartalmazza a felhasználói ikont.
* Az ikon (user.svg) az assets/images mappából kerül betöltésre.
* Az ikon a felhasználói oldalra (user) navigál az aktuális nyelv alapján.

**5. Kosár ikon**

* A cart\_a osztályú <a> elem tartalmazza a kosár ikont.
* Az ikon (cart2.svg) az assets/images mappából kerül betöltésre.
* Az ikon a kosár oldalra (cart) navigál az aktuális nyelv alapján.

**6. Nyelvváltó legördülő menü**

* A language-switcher osztályú <div> tartalmazza a nyelvváltó gombot és a legördülő menüt.
* A nyelvváltó gomb (language\_btn) az aktuális nyelv zászlóját jeleníti meg, amely az assets/images mappából kerül betöltésre.
* A legördülő menü (dropdown) csak akkor jelenik meg, ha a dropdownOpen változó értéke true.
* Elérhető nyelvek:
  + **English**: Az switchLang('en') metódust hívja meg.
  + **Magyar**: Az switchLang('hu') metódust hívja meg.
  + **中文**: Az switchLang('zh') metódust hívja meg.

**Reszponzív elemek**

* A menu-toggle gomb és a nav-links osztály biztosítja a reszponzív kialakítást.
* A menü mobil nézetben a menuOpen változó alapján jelenik meg vagy rejtőzik el.

**Angular direktívák**

* **[routerLink]**:
  + Az útvonalak közötti navigációt biztosítja az aktuális nyelv alapján.
* **routerLinkActive**:
  + Az aktív menüpont kiemelésére szolgál.
* **\*ngIf**:
  + A nyelvváltó legördülő menü megjelenítését vezérli.

## PricesComponent

A **PricesComponent** felelős az árlista megjelenítéséért az alkalmazásban. A komponens figyeli az aktuális nyelvet az URL paraméterekből, és ennek megfelelően tölti be az árlistát. A többnyelvű támogatás biztosítja, hogy az árlista a felhasználó által választott nyelven jelenjen meg.

**Főbb funkciók**

**Nyelv figyelése**

* Az [ngOnInit()](about:blank) metódus figyeli az URL-ben található lang paramétert.
* Ha a nyelv paraméter elérhető, beállítja az aktuális nyelvet ([currentLang](about:blank)).
* Ha nincs nyelv paraméter, az alapértelmezett nyelv (en) kerül beállításra.

**Árlista betöltése**

* A [loadPriceList()](about:blank) metódus felelős az árlista betöltéséért az aktuális nyelv alapján.
* A metódus jelenleg csak egy konzolüzenetet jelenít meg, de bővíthető API-hívásokkal vagy más adatforrásokkal.

**Konstruktor**

A konstruktor inicializálja az [ActivatedRoute](about:blank) szolgáltatást, amely lehetővé teszi az URL paraméterek figyelését.

**Paraméterek**

* [route](about:blank): Az aktuális útvonal figyeléséhez szükséges szolgáltatás.

**Metódusok**

**ngOnInit**

* Az Angular életciklus metódusa, amely a komponens inicializálásakor fut le.
* Figyeli az URL paramétereket, és beállítja az aktuális nyelvet.
* Meghívja a [loadPriceList()](about:blank) metódust az árlista betöltésére.

**loadPriceList**

* Az aktuális nyelv alapján tölti be az árlistát.
* Jelenleg csak egy konzolüzenetet jelenít meg, amely tartalmazza az aktuális nyelvet.

## PricesComponentHTML

A **PricesComponent** HTML sablonja egy árlistát jelenít meg táblázatos formában. A sablon többnyelvű támogatást nyújt az @ngx-translate/core könyvtár segítségével, így az árlista a felhasználó által választott nyelven jelenik meg. Az árlista tartalmazza a szolgáltatások nevét és azok árait.

**1. Cím**

* A <h1> elem tartalmazza az árlista címét.
* A szöveg fordítási kulccsal jelenik meg: "Price List" | translate.

**2. Táblázat**

A táblázat két fő részből áll: a fejlécből (<thead>) és a tartalomból (<tbody>).

**Fejléc**

* A <thead> elem tartalmazza a táblázat oszlopainak címkéit:
  + **Service**: A szolgáltatás neve. Fordítási kulccsal jelenik meg: "Service" | translate.
  + **Price**: A szolgáltatás ára. Fordítási kulccsal jelenik meg: "Price" | translate.

**Tartalom**

* A <tbody> elem tartalmazza a szolgáltatások listáját és azok árait.
* Minden szolgáltatás egy <tr> sorban jelenik meg, amely két <td> cellát tartalmaz:
  + Az első cella a szolgáltatás nevét tartalmazza, például: "Haircut" | translate.
  + A második cella a szolgáltatás árát tartalmazza, például: 4 500 Ft.

**Példa szolgáltatások és árak**

* **Haircut**: 4 500 Ft
* **Beard Trim**: 3 000 Ft
* **Haircut + Beard Trim**: 7 000 Ft
* **Kids Haircut**: 3 500 Ft
* **Shaving**: 2 500 Ft

**Reszponzív elemek**

* A táblázat reszponzív kialakítású lehet, ha a CSS fájlban megfelelő stílusokat definiálnak.
* A container osztály biztosítja az árlista középre igazítását és megfelelő elrendezését.

## ProductDetailsDialogComponent

A **ProductDetailsDialogComponent** egy párbeszédablakot biztosít a termékek részletes adatainak megtekintésére, szerkesztésére és törlésére. A komponens az Angular Material párbeszédablakát használja, és támogatja a többnyelvűséget az @ngx-translate/core könyvtár segítségével. A termékadatok módosítása és törlése API-hívásokkal történik.

**Főbb funkciók**

**Termékadatok megjelenítése**

* A párbeszédablak a termék adatait jeleníti meg, amelyeket a [MAT\_DIALOG\_DATA](about:blank) segítségével kap meg.
* A termék adatai klónozásra kerülnek, hogy elkerüljék az eredeti adatok közvetlen módosítását.

**Termék frissítése**

* A [updateProduct()](about:blank) metódus frissíti a termék adatait az API-n keresztül.
* Sikeres frissítés esetén a párbeszédablak bezárul, és visszaadja az [updatedProduct](about:blank) jelzőt.

**Termék törlése**

* A [deleteProduct()](about:blank) metódus törli a terméket az API-n keresztül.
* Sikeres törlés esetén a párbeszédablak bezárul, és visszaadja a [deletedProduct](about:blank) jelzőt.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például az API-hívásokhoz, fordításokhoz és útvonalak figyeléséhez.

**Paraméterek**

* [data](about:blank): A párbeszédablaknak átadott adatok, amelyek tartalmazzák a termék adatait.
* [dialogRef](about:blank): A párbeszédablak referencia, amely lehetővé teszi a párbeszédablak bezárását és adatok visszaadását.
* [productService](about:blank): A termékek kezeléséhez szükséges szolgáltatás.
* [translate](about:blank): A fordítások kezeléséhez szükséges szolgáltatás.
* [http](about:blank): HTTP kliens az API-hívásokhoz.
* [route](about:blank): Az aktuális útvonal figyeléséhez szükséges szolgáltatás.
* [router](about:blank): Navigációs szolgáltatás az útvonalak kezeléséhez.

**product**

* A termék adatait tárolja, amelyeket a párbeszédablak megjelenít.

**selectedFile**

* A kiválasztott fájlt tárolja a termék képének frissítéséhez (jelenleg nem implementált).

**currentLang**

* Az aktuális nyelvet tárolja. Alapértelmezett értéke: zh.

**Metódusok**

**ngOnInit**

* Figyeli az URL paramétereket, és beállítja az aktuális nyelvet.
* A fordítási szolgáltatás ([TranslateService](about:blank)) segítségével állítja be a nyelvet.

**updateProduct**

* Frissíti a termék adatait az API-n keresztül.
* Sikeres frissítés esetén bezárja a párbeszédablakot, és visszaadja az [updatedProduct](about:blank) jelzőt.
* Hiba esetén a fordítási kulcs alapján hibaüzenetet jelenít meg a konzolban.

**deleteProduct**

* Törli a terméket az API-n keresztül.
* Sikeres törlés esetén bezárja a párbeszédablakot, és visszaadja a [deletedProduct](about:blank) jelzőt.
* Hiba esetén a fordítási kulcs alapján hibaüzenetet jelenít meg a konzolban.

**Fordítási kulcsok**

A sablonban és a metódusokban használt fordítási kulcsok:

* "PRODUCT\_DETAILS.UPDATE\_SUCCESS"
* "PRODUCT\_DETAILS.UPDATE\_ERROR"
* "PRODUCT\_DETAILS.DELETE\_SUCCESS"
* "PRODUCT\_DETAILS.DELETE\_ERROR"

## ProductDetailsDialogComponentHTML

A **ProductDetailsDialogComponent** HTML sablonja egy párbeszédablakot biztosít a termékek részletes adatainak megtekintésére, szerkesztésére és törlésére. Az Angular Material elemek segítségével modern és felhasználóbarát felületet nyújt. A sablon többnyelvű támogatást nyújt az @ngx-translate/core könyvtár segítségével.

**1. Cím**

* A párbeszédablak címe a mat-dialog-title direktívával jelenik meg.
* A szöveg fordítási kulccsal jelenik meg: "PRODUCT\_DETAILS.TITLE" | translate.

**2. Tartalom**

A mat-dialog-content osztályú <div> tartalmazza a termék adatait megjelenítő és szerkesztő mezőket.

**Mezők**

1. **Név**:
   * A termék nevét tartalmazza.
   * Az input mező az [(ngModel)] adatkö binding segítségével kapcsolódik a product.name változóhoz.
   * A mező kötelező (required).
2. **Leírás**:
   * A termék leírását tartalmazza.
   * A textarea mező az [(ngModel)] adatkö binding segítségével kapcsolódik a product.description változóhoz.
   * A mező maximális hossza 70 karakter (maxlength="70").
3. **Ár**:
   * A termék árát tartalmazza.
   * Az input mező type="number" attribútummal rendelkezik, és az [(ngModel)] adatkö binding segítségével kapcsolódik a product.price változóhoz.

**3. Műveleti gombok**

A mat-dialog-actions osztályú <div> tartalmazza a párbeszédablak műveleti gombjait.

**Gombok**

1. **Frissítés gomb**:
   * A updateProduct() metódust hívja meg.
   * A szöveg fordítási kulccsal jelenik meg: "PRODUCT\_DETAILS.UPDATE" | translate.
2. **Törlés gomb**:
   * A deleteProduct() metódust hívja meg.
   * A szöveg fordítási kulccsal jelenik meg: "PRODUCT\_DETAILS.DELETE" | translate.
3. **Bezárás gomb**:
   * A párbeszédablak bezárására szolgál a mat-dialog-close direktíva segítségével.
   * A szöveg fordítási kulccsal jelenik meg: "PRODUCT\_DETAILS.CLOSE" | translate.

**Fordítási kulcsok**

A sablonban használt fordítási kulcsok:

* "PRODUCT\_DETAILS.TITLE"
* "PRODUCT\_DETAILS.NAME"
* "PRODUCT\_DETAILS.DESCRIPTION"
* "PRODUCT\_DETAILS.PRICE"
* "PRODUCT\_DETAILS.UPDATE"
* "PRODUCT\_DETAILS.DELETE"
* "PRODUCT\_DETAILS.CLOSE"

**Reszponzív elemek**

* Az Angular Material elemek automatikusan reszponzívak, és különböző képernyőméretekhez igazodnak.
* A buttons osztály biztosítja a gombok megfelelő elrendezését.

**Angular direktívák**

* **[(ngModel)]**:
  + Az adatkö bindingot biztosítja a termék adatai és a HTML mezők között.
* **mat-dialog-title**:
  + A párbeszédablak címének megjelenítésére szolgál.
* **mat-dialog-content**:
  + A párbeszédablak tartalmának megjelenítésére szolgál.
* **mat-dialog-actions**:
  + A párbeszédablak műveleti gombjainak csoportosítására szolgál.
* **mat-dialog-close**:
  + A párbeszédablak bezárására szolgál.

## ServiceComponent

A **ServiceComponent** egy foglalási rendszer része, amely lehetővé teszi a felhasználók számára, hogy időpontot foglaljanak különböző szolgáltatásokra. A komponens figyeli a foglalásokat, ellenőrzi az elérhetőséget, és API-hívások segítségével tárolja az új foglalásokat. A többnyelvű támogatás az @ngx-translate/core könyvtár segítségével valósul meg.

**Főbb funkciók**

**Időpontfoglalás**

* A felhasználók kiválaszthatják a dátumot, időpontot, nemet és megadhatják az e-mail címüket.
* Az időpontfoglalás az API-n keresztül kerül tárolásra.

**Foglalások ellenőrzése**

* A komponens ellenőrzi, hogy egy adott dátum és időpont foglalt-e.
* A foglalások listája betöltésre kerül az API-ból.

**Elérhető időpontok generálása**

* A [generateAvailableTimes()](about:blank) metódus 8:00 és 18:00 óra közötti időpontokat generál.

**Nyelvi támogatás**

* Az aktuális nyelv ([currentLang](about:blank)) az URL paraméterekből kerül beállításra.
* A fordítások az @ngx-translate/core segítségével történnek.

**Dátumválasztó szűrő**

* A dátumválasztó csak a mai napot követő hétköznapokat engedélyezi.

**Konstruktor**

A konstruktor inicializálja a szükséges szolgáltatásokat, például az API-hívásokhoz, fordításokhoz és útvonalak figyeléséhez.

**Paraméterek**

* [http](about:blank): HTTP kliens az API-hívásokhoz.
* [bookingService](about:blank): A foglalások kezeléséhez szükséges szolgáltatás.
* [router](about:blank): Navigációs szolgáltatás az útvonalak kezeléséhez.
* [route](about:blank): Az aktuális útvonal figyeléséhez szükséges szolgáltatás.
* [translate](about:blank): A fordítások kezeléséhez szükséges szolgáltatás.

**selectedDate**

* A felhasználó által kiválasztott dátumot tárolja.

**selectedTime**

* A felhasználó által kiválasztott időpontot tárolja.

**bookedDates**

* A már lefoglalt időpontokat tartalmazza.

**availableTimes**

* Az elérhető időpontokat tartalmazza.

**showBookedMessage**

* Logikai érték, amely jelzi, hogy a foglalás sikeres volt-e.

**selectedGender**

* A felhasználó által kiválasztott nemet tárolja.

**isDayFullyBooked**

* Logikai érték, amely jelzi, hogy egy adott nap teljesen foglalt-e.

**currentLang**

* Az aktuális nyelvet tárolja. Alapértelmezett értéke: en.

**email**

* A felhasználó e-mail címét tárolja.

**showAvailableMessage**

* Logikai érték, amely jelzi, hogy az időpont elérhető-e.

**isTimeBooked**

* Logikai érték, amely jelzi, hogy az időpont foglalt-e.

**myFilter**

* A dátumválasztó szűrője, amely csak a hétköznapokat engedélyezi.

**Metódusok**

**ngOnInit**

* Figyeli az URL paramétereket, és beállítja az aktuális nyelvet.
* Betölti a már lefoglalt időpontokat.
* Generálja az elérhető időpontokat.
* Automatikusan kitölti az e-mail mezőt, ha az elérhető a [localStorage](about:blank)-ban.

**generateAvailableTimes**

* 8:00 és 18:00 óra közötti időpontokat generál.

**onDateChange**

* Frissíti a kiválasztott dátumot.
* Ellenőrzi, hogy a nap teljesen foglalt-e.

**onApplyClick**

* Ellenőrzi, hogy minden mező ki van-e töltve.
* Ha igen, elmenti a foglalást az API-n keresztül.

**checkBooking**

* Ellenőrzi, hogy a kiválasztott dátum és időpont foglalt-e.

**loadBookedDates**

* Betölti a már lefoglalt időpontokat az API-ból.

**isDateBooked**

* Ellenőrzi, hogy egy adott dátum és időpont foglalt-e.

**onTimeSelect**

* Frissíti a kiválasztott időpontot.

**canBook**

* Ellenőrzi, hogy minden szükséges mező ki van-e töltve a foglaláshoz.

## ServiceComponentHTML

A **ServiceComponent** HTML sablonja egy foglalási űrlapot biztosít, amely lehetővé teszi a felhasználók számára, hogy időpontot foglaljanak különböző szolgáltatásokra. Az Angular Material elemek segítségével modern és felhasználóbarát felületet nyújt. A sablon többnyelvű támogatást nyújt az @ngx-translate/core könyvtár segítségével.

**1. Dátumválasztó**

* A dátumválasztó (mat-datepicker) lehetővé teszi a felhasználók számára, hogy kiválasszanak egy dátumot.
* A dátumválasztó szűrője (myFilter) csak a hétköznapokat engedélyezi.
* A kiválasztott dátum változását az onDateChange($event) metódus kezeli.

**Elemei:**

* **mat-label**: A dátumválasztó címkéje, fordítási kulccsal: "Choose a date" | translate.
* **mat-datepicker-toggle**: A dátumválasztó ikonja.
* **mat-datepicker-actions**: Gombok a dátumválasztó alkalmazásához vagy visszavonásához.

**2. Időpontválasztó**

* Az időpontválasztó (mat-select) lehetővé teszi a felhasználók számára, hogy kiválasszanak egy elérhető időpontot.
* Az időpontok listája az availableTimes tömbből generálódik.
* Az isDateBooked() metódus ellenőrzi, hogy az adott időpont foglalt-e.

**Elemei:**

* **mat-label**: Az időpontválasztó címkéje, fordítási kulccsal: "Choose a time" | translate.
* **mat-option**: Az elérhető időpontok listája.

**3. Nem kiválasztása**

* A nem kiválasztása (mat-select) lehetővé teszi a felhasználók számára, hogy megadják a nemüket.
* Az opciók: "Male", "Female", "Other", fordítási kulcsokkal.

**Elemei:**

* **mat-label**: A nem kiválasztásának címkéje, fordítási kulccsal: "Choose your gender" | translate.
* **mat-option**: A nemek listája.

**4. E-mail mező**

* Az e-mail mező (mat-input) lehetővé teszi a felhasználók számára, hogy megadják az e-mail címüket.
* Az [(ngModel)] adatkö binding segítségével kapcsolódik az email változóhoz.

**Elemei:**

* **mat-label**: Az e-mail mező címkéje, fordítási kulccsal: "Enter your email" | translate.
* **placeholder**: Példaérték az e-mail mezőben: "example@example.com".

**5. Foglalás gomb**

* A foglalás gomb (mat-raised-button) lehetővé teszi a felhasználók számára, hogy elküldjék a foglalási adatokat.
* A gomb csak akkor aktív, ha minden szükséges mező ki van töltve ([disabled]="!canBook()").

**Elemei:**

* **color="primary"**: A gomb színe.
* **(click)="onApplyClick()"**: A foglalási metódus meghívása.
* **Fordítási kulcs**: "Book" | translate.

**6. Értesítések**

* Az értesítések különböző állapotokat jelenítenek meg:
  + **Foglalás sikeres**: "NOTIFICATION.BOOKING\_SUCCESS" | translate.
  + **Időpont foglalt**: "NOTIFICATION.TIME\_BOOKED" | translate.
  + **Időpont elérhető**: "NOTIFICATION.TIME\_AVAILABLE" | translate.

**Elemei:**

* **\*ngIf**: Feltételes megjelenítés az értesítésekhez.
* **CSS osztályok**:
  + notification success: Sikeres foglalás.
  + notification booked: Foglalt időpont.
  + notification available: Elérhető időpont.

## CartService

A **CartService** egy Angular szolgáltatás, amely az alkalmazás kosár funkciójához kapcsolódó API-hívásokat kezeli. A szolgáltatás lehetővé teszi a kosár tartalmának lekérdezését, termékek hozzáadását a kosárhoz, valamint termékek eltávolítását a kosárból.

**1. Kosár tartalmának lekérdezése**

* A [getCart()](about:blank) metódus az API-tól lekéri a kosár aktuális tartalmát.
* Az API végpont: [**GET http://localhost:8000/api/cart**](about:blank)**.**

**2. Termék hozzáadása a kosárhoz**

* Az [addToCart(productId: number, quantity: number)](about:blank) metódus egy terméket ad a kosárhoz a megadott mennyiséggel.
* Az API végpont: [**POST http://localhost:8000/api/cart**](about:blank)**.**

**Termék eltávolítása a kosárból**

* A [removeFromCart(cartItemId: number)](about:blank) metódus eltávolít egy terméket a kosárból az adott kosár elem azonosítója alapján.
* Az API végpont: [**DELETE http://localhost:8000/api/cart/<cartItemId>**](about:blank)**.**

## ProductService

A **ProductService** egy Angular szolgáltatás, amely a termékek kezeléséhez szükséges funkciókat biztosítja. A szolgáltatás lehetővé teszi a termékek lekérdezését, egyedi termékek azonosító alapján történő elérését, valamint az összes termék listájának lekérését.

**Főbb funkciók**

**1. Termék lekérdezése azonosító alapján (aszinkron)**

* A [getProduct(productId: number)](about:blank) metódus egy terméket ad vissza az azonosítója alapján.
* Az eredmény egy [Observable<Product>](about:blank), amely egy példányterméket tartalmaz.

**2. Termék lekérdezése azonosító alapján (szinkron)**

* A [getProductById(id: number)](about:blank) metódus egy terméket ad vissza az azonosítója alapján.
* Az eredmény egy [Product](about:blank) típusú objektum.

**3. Összes termék lekérdezése**

* A [getAll()](about:blank) metódus az összes elérhető terméket visszaadja egy tömbben.
* Az eredmény egy [Product[]](about:blank) típusú tömb.

**Konstruktor**

A konstruktor jelenleg nem végez semmilyen inicializálást, de szükséges az Angular szolgáltatásként való regisztrációhoz.

**Metódusok**

**1. getProduct**

* **Leírás**: Egy terméket ad vissza az azonosítója alapján aszinkron módon.
* **Paraméterek**:
  + [productId](about:blank): A lekérdezni kívánt termék azonosítója.
* **Visszatérési érték**: [Observable<Product>](about:blank) - Egy terméket tartalmazó adatfolyam.

**2. getProductById**

* **Leírás**: Egy terméket ad vissza az azonosítója alapján szinkron módon.
* **Paraméterek**:
  + [id](about:blank): A lekérdezni kívánt termék azonosítója.
* **Visszatérési érték**: [Product](about:blank) - Egy termék objektum.

**3. getAll**

* **Leírás**: Az összes elérhető terméket visszaadja.
* **Visszatérési érték**: [Product[]](about:blank) - A termékek listája.

## Cartosztály

A **Cart** osztály az alkalmazás kosármodelljét reprezentálja. Ez az osztály tartalmazza a kosárban lévő tételek listáját, valamint egy számított tulajdonságot, amely a kosár teljes árát adja vissza.

**Tulajdonságok**

**1. items**

* **Típus**: [CartItem[]](about:blank)
* **Leírás**: A kosárban lévő tételek listája.
* **Alapértelmezett érték**: Üres tömb ([]).

**Számított tulajdonságok**

**1. totalPrice**

* **Típus**: number
* **Leírás**: A kosár teljes árát adja vissza.
* **Működés**:
  + Végigiterál a [items](about:blank) tömbön.
  + Összeadja az egyes tételek árát ([item.price](about:blank)).
  + Visszaadja az összesített értéket.

## ApiService

Az **ApiService** egy Angular szolgáltatás, amely az alkalmazás különböző funkcióihoz kapcsolódó API-hívásokat kezeli. A szolgáltatás biztosítja a felhasználók regisztrációját, bejelentkezését, jogosultságkezelését, foglalások kezelését, valamint a kosár funkcióit.

**1. Felhasználókezelés**

* **Regisztráció**: A [registerUser](about:blank) metódus új felhasználót regisztrál.
* **Bejelentkezés**: A [loginUser](about:blank) metódus hitelesíti a felhasználót.
* **Token ellenőrzése**: A [verifyToken](about:blank) metódus ellenőrzi a felhasználói token érvényességét.
* **Felhasználói adatok lekérdezése**: A [getUserInfo](about:blank) metódus visszaadja a felhasználó adatait.
* **Kijelentkezés**: A [logoutUser](about:blank) metódus kijelentkezteti a felhasználót.

**2. Adminisztráció**

* **Admin jogosultság hozzáadása**: Az [upgradeToAdmin](about:blank) metódus admin jogosultságot ad egy felhasználónak.
* **Admin jogosultság eltávolítása**: A [downgradeFromAdmin](about:blank) metódus eltávolítja az admin jogosultságot egy felhasználótól.

**3. Felhasználói adatok frissítése**

* **Felhasználónév frissítése**: Az [updateUserName](about:blank) metódus frissíti a felhasználó nevét.
* **Jelszó frissítése**: Az [updatePassword](about:blank) metódus frissíti a felhasználó jelszavát.
* **Régi jelszó ellenőrzése**: A [checkOldPassword](about:blank) metódus ellenőrzi a felhasználó régi jelszavát.

**4. Foglalások kezelése**

* **Foglalások lekérdezése**: A [getUserBookings](about:blank) és [getAllBookings](about:blank) metódusok visszaadják a felhasználó foglalásait.
* **Foglalás mentése**: A [storeBooking](about:blank) metódus új foglalást ment.
* **Foglalás törlése**: A [deleteBooking](about:blank) metódus töröl egy foglalást.

**5. Kosár funkciók**

* **Kosár elemek lekérdezése**: A [getCartItems](about:blank) metódus visszaadja a kosár tartalmát.
* **Kosár elem eltávolítása**: A [removeCartItem](about:blank) metódus eltávolít egy elemet a kosárból.

**Konstruktor**

A konstruktor inicializálja az [HttpClient](about:blank) szolgáltatást, amely az API-hívások végrehajtásához szükséges.

**Paraméterek**

* [http](about:blank): Az Angular [HttpClient](about:blank) szolgáltatása az HTTP-kérések kezeléséhez.

**Metódusok**

**Felhasználókezelés**

1. **registerUser**
   * **Leírás**: Új felhasználót regisztrál.
   * **Paraméterek**:
     + [lang](about:blank): A nyelv kódja.
     + [formData](about:blank): A regisztrációs adatok.
   * **Visszatérési érték**: [Observable<any>](about:blank).
2. **loginUser**
   * **Leírás**: Hitelesíti a felhasználót.
   * **Paraméterek**:
     + [lang](about:blank): A nyelv kódja.
     + [formData](about:blank): A bejelentkezési adatok.
   * **Visszatérési érték**: [Observable<any>](about:blank).
3. **verifyToken**
   * **Leírás**: Ellenőrzi a felhasználói token érvényességét.
   * **Paraméterek**:
     + [token](about:blank): A felhasználói token.
   * **Visszatérési érték**: [Observable<any>](about:blank).
4. **getUserInfo**
   * **Leírás**: Visszaadja a felhasználó adatait.
   * **Paraméterek**:
     + [token](about:blank): A felhasználói token.
   * **Visszatérési érték**: [Observable<any>](about:blank).
5. **logoutUser**
   * **Leírás**: Kijelentkezteti a felhasználót.
   * **Paraméterek**:
     + [lang](about:blank): A nyelv kódja.
   * **Visszatérési érték**: [Observable<any>](about:blank).

**Adminisztráció**

1. **upgradeToAdmin**
   * **Leírás**: Admin jogosultságot ad egy felhasználónak.
   * **Paraméterek**:
     + [useremail](about:blank): A felhasználó e-mail címe.
   * **Visszatérési érték**: [Observable<any>](about:blank).
2. **downgradeFromAdmin**
   * **Leírás**: Eltávolítja az admin jogosultságot egy felhasználótól.
   * **Paraméterek**:
     + [useremail](about:blank): A felhasználó e-mail címe.
   * **Visszatérési érték**: [Observable<any>](about:blank).

**Foglalások kezelése**

1. **getUserBookings**
   * **Leírás**: Visszaadja a felhasználó foglalásait.
   * **Paraméterek**:
     + [role](about:blank): A felhasználó szerepköre.
     + [email](about:blank): A felhasználó e-mail címe.
   * **Visszatérési érték**: [Observable<any>](about:blank).
2. **storeBooking**
   * **Leírás**: Új foglalást ment.
   * **Paraméterek**:
     + [bookingData](about:blank): A foglalás adatai.
   * **Visszatérési érték**: [Observable<any>](about:blank).
3. **deleteBooking**
   * **Leírás**: Töröl egy foglalást.
   * **Paraméterek**:
     + [bookingId](about:blank): A foglalás azonosítója.
   * **Visszatérési érték**: [Observable<any>](about:blank).

**Kosár funkciók**

1. **getCartItems**
   * **Leírás**: Visszaadja a kosár tartalmát.
   * **Visszatérési érték**: [Observable<any>](about:blank).
2. **removeCartItem**
   * **Leírás**: Eltávolít egy elemet a kosárból.
   * **Paraméterek**:
     + [productId](about:blank): A termék azonosítója.
   * **Visszatérési érték**: [Observable<any>](about:blank).

## AppRoutingModule

Az **AppRoutingModule** az Angular alkalmazás útvonalait kezeli. Az útvonalak lehetővé teszik a felhasználók számára, hogy különböző oldalakra navigáljanak az alkalmazáson belül. Az útvonalak támogatják a többnyelvűséget a :lang paraméter használatával, és bizonyos oldalakhoz hozzáférési védelem ([AuthGuard](about:blank)) is tartozik.

**1. Többnyelvűség támogatása**

* Az útvonalak tartalmazzák a :lang paramétert, amely a nyelvi kódot (en, hu, stb.) jelöli.
* Példa: /en/home, /hu/login.

**2. Hozzáférés védelem**

* Az [AuthGuard](about:blank) biztosítja, hogy bizonyos oldalak csak hitelesített felhasználók számára legyenek elérhetők.
* Példa: /en/user, /en/service.

**3. Alapértelmezett útvonal**

* Az alapértelmezett útvonal ('') átirányít a /en/home oldalra.

**Útvonalak**

**1. Kezdőlap**

* **Útvonal**: :lang/home
* **Komponens**: [HomeComponent](about:blank)

**2. Regisztráció**

* **Útvonal**: :lang/register
* **Komponens**: [RegisterComponent](about:blank)

**3. Bejelentkezés**

* **Útvonal**: :lang/login
* **Komponens**: [LoginComponent](about:blank)

**4. Felhasználói oldal**

* **Útvonal**: :lang/user
* **Komponens**: [UserComponent](about:blank)
* **Védelem**: [AuthGuard](about:blank)

**5. Szolgáltatások**

* **Útvonal**: :lang/service
* **Komponens**: [ServiceComponent](about:blank)
* **Védelem**: [AuthGuard](about:blank)

**6. Kiszállítás**

* **Útvonal**: :lang/delivery és delivery
* **Komponens**: [DeliveryPageComponent](about:blank)

**7. Webshop**

* **Útvonal**: :lang/webshop
* **Komponens**: [WebshopPageComponent](about:blank)
* **Védelem**: [AuthGuard](about:blank)

**8. Termékoldal**

* **Útvonal**: product/:id
* **Komponens**: [ProductPageComponent](about:blank)
* **Védelem**: [AuthGuard](about:blank)

**9. Kosár**

* **Útvonal**: :lang/cart
* **Komponens**: [CartPageComponent](about:blank)
* **Védelem**: [AuthGuard](about:blank)

**10. Termékkezelés**

* **Útvonal**: :lang/Product-Management
* **Komponens**: [AddProductComponent](about:blank)

**11. Árlista**

* **Útvonal**: :lang/prices
* **Komponens**: [PricesComponent](about:blank)

**12. Alapértelmezett átirányítás**

* **Útvonal**: ''
* **Átirányítás**: /en/home

## AppModule

Az **AppModule** az Angular alkalmazás fő modulja, amely az alkalmazás összes komponensét, szolgáltatását és külső könyvtárát regisztrálja. Ez a modul felelős az alkalmazás inicializálásáért és a szükséges modulok importálásáért.

**Főbb funkciók**

**1. Komponensek regisztrálása**

* Az alkalmazás összes komponense regisztrálva van a [declarations](about:blank) tömbben, például:
  + [HomeComponent](about:blank)
  + [NavbarComponent](about:blank)
  + [LoginComponent](about:blank)
  + [ServiceComponent](about:blank)
  + [PricesComponent](about:blank)

**2. Modulok importálása**

* Az Angular alapmoduljai és külső könyvtárak importálása, például:
  + [BrowserModule](about:blank)
  + [AppRoutingModule](about:blank)
  + [HttpClientModule](about:blank)
  + [TranslateModule](about:blank) (többnyelvűség támogatására)
  + Angular Material modulok ([MatDatepickerModule](about:blank), [MatDialogModule](about:blank), stb.)

**3. Szolgáltatások regisztrálása**

* Az alkalmazás szolgáltatásai regisztrálva vannak a [providers](about:blank) tömbben, például:
  + [ShoppingCartService](about:blank)
  + [ProductService](about:blank)

**4. Bootstrap komponens**

* Az alkalmazás indításakor a [bootstrap](about:blank) tömbben megadott [AppComponent](about:blank) kerül betöltésre.

## AuthGuard

Az **AuthGuard** egy Angular őr (guard), amely biztosítja, hogy csak hitelesített felhasználók férhessenek hozzá bizonyos útvonalakhoz. Ha a felhasználó nincs bejelentkezve, az őr átirányítja a bejelentkezési oldalra, és megőrzi a cél URL-t, hogy sikeres bejelentkezés után oda navigálhasson vissza.

**Főbb funkciók**

**1. Hozzáférés ellenőrzése**

* Az [AuthGuard](about:blank) az [AuthService](about:blank) segítségével ellenőrzi, hogy a felhasználó be van-e jelentkezve.
* Ha a felhasználó be van jelentkezve, az őr engedélyezi az útvonal elérését.

**2. Átirányítás bejelentkezési oldalra**

* Ha a felhasználó nincs bejelentkezve:
  + Az őr átirányítja a felhasználót a bejelentkezési oldalra.
  + Az aktuális URL-t ([state.url](about:blank)) a [redirectUrl](about:blank) paraméterben továbbítja, hogy a bejelentkezés után vissza lehessen navigálni.

**Konstruktor**

A konstruktor inicializálja az őr működéséhez szükséges szolgáltatásokat.

**Paraméterek**

* [authService](about:blank): Az [AuthService](about:blank) szolgáltatás, amely ellenőrzi a felhasználó bejelentkezési állapotát.
* [router](about:blank): Az Angular [Router](about:blank) szolgáltatása, amely az útvonalak közötti navigációért felelős.

**Metódusok**

**1. canActivate**

* **Leírás**: Ellenőrzi, hogy a felhasználó hozzáférhet-e az adott útvonalhoz.
* **Paraméterek**:
  + [route](about:blank): Az aktuális útvonal pillanatképe ([ActivatedRouteSnapshot](about:blank)).
  + [state](about:blank): Az aktuális útvonal állapota ([RouterStateSnapshot](about:blank)).
* **Működés**:
  + Ha a felhasználó be van jelentkezve ([authService.isLoggedIn()](about:blank)):
    - Az őr true értéket ad vissza, és engedélyezi az útvonal elérését.
  + Ha a felhasználó nincs bejelentkezve:
    - Az őr átirányítja a bejelentkezési oldalra.
    - A nyelvi paramétert (:lang) az URL-ből olvassa ki, vagy alapértelmezettként az en értéket használja.
    - A cél URL-t ([state.url](about:blank)) a [redirectUrl](about:blank) paraméterben továbbítja.
    - Az őr false értéket ad vissza.
* **Visszatérési érték**: boolean - Az útvonal elérhetőségének állapota.

## AuthService

Az **AuthService** egy Angular szolgáltatás, amely a felhasználók hitelesítéséhez és jogosultságkezeléséhez kapcsolódó funkciókat biztosítja. Ez a szolgáltatás kezeli a felhasználói bejelentkezési állapotot, a szerepköröket, valamint a token érvényességének ellenőrzését.

**1. Felhasználói szerepkörök kezelése**

* A [setRole](about:blank) és [getRole](about:blank) metódusok segítségével a felhasználói szerepkör beállítható és lekérdezhető.
* Az [isAdmin](about:blank) metódus ellenőrzi, hogy a felhasználó adminisztrátori jogosultsággal rendelkezik-e.

**2. Bejelentkezési állapot ellenőrzése**

* Az [isLoggedIn](about:blank) metódus ellenőrzi, hogy a felhasználó be van-e jelentkezve, a [localStorage](about:blank)-ban tárolt token alapján.

**3. Token érvényességének ellenőrzése**

* A [verifyToken](about:blank) metódus az [ApiService](about:blank) segítségével ellenőrzi, hogy a felhasználói token érvényes-e.

**Konstruktor**

A konstruktor inicializálja az [ApiService](about:blank) szolgáltatást, amely az API-hívások végrehajtásához szükséges.

**Paraméterek**

* [apiService](about:blank): Az [ApiService](about:blank) szolgáltatás, amely az API-hívások kezeléséért felelős.

## BookingService

A **BookingService** egy Angular szolgáltatás, amely a foglalások kezeléséhez szükséges API-hívásokat kezeli. Ez a szolgáltatás lehetővé teszi a foglalási adatok mentését az API-n keresztül.

**Főbb funkciók**

**1. Foglalás mentése**

* A [storeBooking](about:blank) metódus segítségével a foglalási adatokat elküldhetjük az API-nak.
* Az API végpont: [**POST http://localhost:8000/api/check-booking/store**](about:blank)**.**

**Konstruktor**

A konstruktor inicializálja az [HttpClient](about:blank) szolgáltatást, amely az API-hívások végrehajtásához szükséges.

**Paraméterek**

* [http](about:blank): Az Angular [HttpClient](about:blank) szolgáltatása az HTTP-kérések kezeléséhez.

**Tulajdonságok**

**1. apiUrl**

* **Típus**: string
* **Leírás**: Az API végpont URL-je, amely a foglalási adatok mentésére szolgál.
* **Érték**: [**http://localhost:8000/api/check-booking/store**](about:blank)**.**

**Metódusok**

**1. storeBooking**

* **Leírás**: Elküldi a foglalási adatokat az API-nak.
* **Paraméterek**:
  + [bookingData](about:blank): Egy objektum, amely tartalmazza a foglalás dátumát és időpontját:
    - [date](about:blank): A foglalás dátuma (string).
    - [time](about:blank): A foglalás időpontja (string).
* **Visszatérési érték**: Observable<any> - Az API válaszát tartalmazó adatfolyam.

## ProductService

A **ProductService** egy Angular szolgáltatás, amely a termékek kezeléséhez szükséges API-hívásokat kezeli. Ez a szolgáltatás lehetővé teszi a termékek hozzáadását, lekérdezését, frissítését és törlését az API-n keresztül.

**Főbb funkciók**

**1. Termék hozzáadása**

* Az [addProduct](about:blank) metódus új terméket ad hozzá az API-n keresztül.
* Az API végpont: [POST http://localhost:8000/api/products](about:blank).

**2. Egyedi termék lekérdezése**

* A [getProduct](about:blank) metódus egy adott terméket kér le az azonosítója alapján.
* Az API végpont: [GET http://localhost:8000/api/products/:id](about:blank).

**3. Összes termék lekérdezése**

* A [getProducts](about:blank) metódus az összes terméket visszaadja.
* Az API végpont: [GET http://localhost:8000/api/products](about:blank).

**4. Termék frissítése**

* Az [updateProduct](about:blank) metódus egy meglévő terméket frissít az azonosítója alapján.
* Az API végpont: [PUT http://localhost:8000/api/products/:id](about:blank).

**5. Termék törlése**

* A [deleteProduct](about:blank) metódus egy terméket töröl az azonosítója alapján.
* Az API végpont: [DELETE http://localhost:8000/api/products/:id](about:blank).

**Konstruktor**

A konstruktor inicializálja az [HttpClient](about:blank) szolgáltatást, amely az API-hívások végrehajtásához szükséges.

**Paraméterek**

* [http](about:blank): Az Angular [HttpClient](about:blank) szolgáltatása az HTTP-kérések kezeléséhez.

**Tulajdonságok**

**1. apiUrl**

* **Típus**: string
* **Leírás**: Az API alap URL-je, amely a termékek kezelésére szolgál.
* **Érték**: [**http://localhost:8000/api/products**](about:blank)**.**

**Metódusok**

**1. addProduct**

* **Leírás**: Új terméket ad hozzá az API-n keresztül.
* **Paraméterek**:
  + [productData](about:blank): A termék adatai [FormData](about:blank) formátumban.
* **Visszatérési érték**: [Observable<Product>](about:blank) - Az API által visszaadott termék.

## ShoppingCartService

A **ShoppingCartService** egy Angular szolgáltatás, amely a kosár funkcióit kezeli. Ez a szolgáltatás lehetővé teszi a termékek hozzáadását, eltávolítását, a kosár tartalmának lekérdezését, valamint a helyi (localStorage) és backend kosár szinkronizálását.

**Főbb funkciók**

**1. Backend kosár kezelése**

* **Termék hozzáadása**: A [addCartItem](about:blank) metódus segítségével termékeket adhatunk a backend kosárhoz.
* **Kosár lekérése**: A [getCartFromBackend](about:blank) metódus visszaadja a backend kosár tartalmát.
* **Termék eltávolítása**: A [removeCartItem](about:blank) metódus eltávolít egy terméket a backend kosárból.
* **Mennyiség csökkentése**: A [decreaseCartItem](about:blank) metódus csökkenti egy termék mennyiségét a backend kosárban.

**2. Helyi kosár kezelése**

* **Helyi kosár mentése**: A [saveCart](about:blank) metódus menti a kosár tartalmát a [localStorage](about:blank)-ba.
* **Helyi kosár törlése**: A [clearCart](about:blank) metódus törli a helyi kosár tartalmát.
* **Helyi kosár lekérése**: A [getCart](about:blank) metódus visszaadja a helyi kosár tartalmát.
* **Termék hozzáadása a helyi kosárhoz**: Az [addToLocalCart](about:blank) metódus terméket ad a helyi kosárhoz.

**Konstruktor**

A konstruktor inicializálja a szolgáltatást, és betölti a helyi kosár tartalmát a [localStorage](about:blank)-ból.

**Paraméterek**

* [http](about:blank): Az Angular [HttpClient](about:blank) szolgáltatása az API-hívások kezeléséhez.

**Tulajdonságok**

**1. baseUrl**

* **Típus**: string
* **Leírás**: A backend alap URL-je.
* **Érték**: [http://localhost:8000/](about:blank).

**2. cart**

* **Típus**: { product: Product; quantity: number }[]
* **Leírás**: A helyi kosár tartalmát tárolja.

**Metódusok**

**1. addCartItem**

* **Leírás**: Terméket ad a backend kosárhoz.
* **Paraméterek**:
  + [productId](about:blank): A termék azonosítója (number).
  + [quantity](about:blank): A termék mennyisége (number).

## UserService

A **UserService** egy Angular szolgáltatás, amely a felhasználói szerepkörök kezelésére szolgál. Ez a szolgáltatás a backend API-val kommunikál, hogy lekérje a felhasználó aktuális szerepkörét.

**Főbb funkciók**

**1. Felhasználói szerepkör lekérdezése**

* A [getUserRole](about:blank) metódus segítségével a felhasználó aktuális szerepköre lekérhető az API-ból.
* Az API végpont: [**GET http://localhost:8000/api/userRole/role**](about:blank)**.**

**Konstruktor**

**Leírás**

A konstruktor inicializálja az [HttpClient](about:blank) szolgáltatást, amely az API-hívások végrehajtásához szükséges.

**Paraméterek**

* [http](about:blank): Az Angular [HttpClient](about:blank) szolgáltatása az HTTP-kérések kezeléséhez.

**apiUrl**

* **Típus**: string
* **Leírás**: Az API alap URL-je, amely a felhasználói szerepkörök kezelésére szolgál.
* **Érték**: [**http://localhost:8000/api/userRole**](about:blank)**.**

## Loading component, HTML

LoginComponent betöltési kinézet.

Map component ---Hamarosan

**Assets/I18n/em.json, hu.json, zh.json**

Nyelvi változtatások paraméterei.

# Back End

**Kontrollerek**

## BookingController

A BookingController egy Laravel vezérlő, amely a foglalások kezelésére szolgál. Ez a vezérlő különböző funkciókat biztosít, például foglalások ellenőrzését, létrehozását, lekérdezését és jogosultságkezelést. Az alábbiakban részletesen bemutatjuk az egyes metódusokat és azok működését.

Metódusok

1. checkBooking

Leírás: Ellenőrzi, hogy egy adott dátum és időpont foglalt-e.

HTTP metódus: GET

URL: /api/check-booking

Paraméterek:

* date (kötelező): A foglalás dátuma.
* time (kötelező): A foglalás időpontja.

Válasz:

* booked: true: Az időpont foglalt.
* booked: false: Az időpont szabad.

2. storeBooking

Leírás: Új foglalást hoz létre, ha az adott időpont még nem foglalt.

HTTP metódus: POST

URL: /api/check-booking/store

Paraméterek:

* date (kötelező): A foglalás dátuma.
* time (kötelező): A foglalás időpontja (H:i formátumban).
* gender (kötelező): A foglaló személy neme.
* email (kötelező): A foglaló e-mail címe.

Funkciók:

* Ellenőrzi, hogy az adott időpont már foglalt-e.
* Ha nem foglalt, létrehoz egy új foglalást az adatbázisban.
* E-mail értesítést küld a foglalónak a foglalás részleteiről.

Válasz:

Sikeres foglalás esetén: A létrehozott foglalás adatai.

* Hiba esetén: 400 Bad Request (pl. ha az időpont már foglalt).

3. getBookedDates

Leírás: Az összes foglalt dátumot és időpontot adja vissza.

HTTP metódus: GET

URL: /api/check-booking/booked-dates

Paraméterek: Nincs.

Válasz:

* Egy JSON objektum, amely tartalmazza az összes foglalt dátumot és időpontot.

4. getUserBookings

Leírás: Egy adott felhasználó foglalásait adja vissza az e-mail címe alapján.

HTTP metódus: GET

URL: /api/user-bookings

Paraméterek:

* email (kötelező): A felhasználó e-mail címe.

Válasz:

* Sikeres lekérdezés esetén: A felhasználó foglalásai.
* Hiba esetén: 400 Bad Request (pl. ha az e-mail cím hiányzik).

5. getBookings

Leírás: Egy adott felhasználó foglalásait adja vissza az e-mail címe alapján, figyelembe véve a szerepkört.

HTTP metódus: GET

URL: /api/bookings

Paraméterek:

* role (kötelező): A felhasználó szerepköre.
* email (kötelező): A felhasználó e-mail címe.

Válasz:

* Sikeres lekérdezés esetén: A felhasználó foglalásai.
* Hiba esetén: 403 Unauthorized (pl. ha a felhasználó nem jogosult).

6. getAllBookings

Leírás: Az összes foglalást visszaadja, de csak adminisztrátorok és szuperadminisztrátorok számára érhető el.

HTTP metódus: GET

URL: /api/all-bookings

Paraméterek:

* role (kötelező): A felhasználó szerepköre.
* email (kötelező): A felhasználó e-mail címe.

Válasz:

* Sikeres lekérdezés esetén: Az összes foglalás dátum és idő szerint rendezve.
* Hiba esetén: 403 Unauthorized (pl. ha a felhasználó nem adminisztrátor vagy szuperadminisztrátor).
* Főbb Funkciók
* Foglalások ellenőrzése: Az adott dátum és időpont foglaltságának ellenőrzése.
* Foglalások létrehozása: Új foglalások mentése az adatbázisba, és e-mail értesítések küldése.
* Foglalások lekérdezése: Egyéni vagy összes foglalás lekérdezése különböző szűrési feltételekkel.
* Jogosultságkezelés: Csak adminisztrátorok és szuperadminisztrátorok férhetnek hozzá az összes foglaláshoz.

Hibakezelés

* Időpont foglaltság ellenőrzése: Ha az időpont már foglalt, a rendszer hibaüzenetet küld (400 Bad Request).
* Hiányzó adatok: Ha kötelező mezők hiányoznak (pl. e-mail cím), a rendszer hibaüzenetet ad vissza.
* Jogosultság hiánya: Ha egy felhasználó nem jogosult az adott műveletre, a rendszer 403 Unauthorized hibát küld.

E-mail Értesítések

* A storeBooking metódus e-mail értesítést küld a foglalónak a foglalás részleteiről a BookingConfirmationMail osztály segítségével. Ez biztosítja, hogy a felhasználó értesüljön a foglalás sikerességéről.

## CartController

A CartController egy Laravel vezérlő, amely a felhasználók kosarának kezelésére szolgál. Ez a vezérlő biztosítja a kosárhoz kapcsolódó műveletek végrehajtását, például a kosár tartalmának lekérdezését, termékek hozzáadását, frissítését, eltávolítását és a mennyiség csökkentését.

Metódusok

1. index

Leírás: A bejelentkezett felhasználó kosarának tartalmát adja vissza.

HTTP metódus: GET

URL: /api/cart

Funkciók:

* Lekérdezi a bejelentkezett felhasználó kosarának tartalmát az adatbázisból.
* Betölti a kosárban lévő termékek részleteit.

Válasz:

* A kosár tartalma, beleértve a termékek adatait és mennyiségét.

2. store

Leírás: Termék hozzáadása a kosárhoz.

HTTP metódus: POST

URL: /api/cart

Paraméterek:

* product\_id (kötelező): A hozzáadni kívánt termék azonosítója.
* quantity (kötelező): A hozzáadni kívánt mennyiség.
* Funkciók:
* Ellenőrzi, hogy a termék már szerepel-e a kosárban.
* Ha a termék már szerepel, növeli a mennyiséget.
* Ha a termék nem szerepel, új bejegyzést hoz létre a kosárban.

Válasz:

* Sikeres hozzáadás esetén a kosár frissített tartalma.
* Hiba esetén megfelelő hibaüzenet.

3. update

Leírás: Egy adott kosárbejegyzés frissítése.

HTTP metódus: PUT

URL: /api/cart/{id}

Paraméterek:

* quantity (kötelező): Az új mennyiség.

Funkciók:

* Megkeresi a megadott azonosítójú kosárbejegyzést.
* Frissíti a mennyiséget az adatbázisban.
* Betölti a termék részleteit a válaszhoz.

Válasz:

* A frissített kosárbejegyzés adatai.
* Hiba esetén megfelelő hibaüzenet.

4. destroy

Leírás: Egy termék eltávolítása a kosárból.

HTTP metódus: DELETE

URL: /api/cartdelete/{id}

Funkciók:

* Megkeresi a megadott azonosítójú terméket a kosárban.
* Törli a terméket a kosárból.

Válasz:

* Sikeres törlés esetén megerősítő üzenet.
* Hiba esetén megfelelő hibaüzenet.

5. decreaseQuantity

Leírás: Egy termék mennyiségének csökkentése a kosárban.

HTTP metódus: POST

URL: /api/cart/decrease/{productId}

Funkciók:

* Megkeresi a megadott terméket a kosárban.
* Ha a mennyiség nagyobb, mint 1, csökkenti a mennyiséget.
* Ha a mennyiség 1, eltávolítja a terméket a kosárból.

Válasz:

* Sikeres csökkentés vagy törlés esetén megerősítő üzenet.
* Hiba esetén megfelelő hibaüzenet.

Főbb Funkciók

* Kosár tartalmának lekérdezése: A bejelentkezett felhasználó kosarának teljes tartalmát visszaadja.
* Termék hozzáadása: Új termék hozzáadása a kosárhoz, vagy meglévő termék mennyiségének növelése.
* Kosár frissítése: Egy adott kosárbejegyzés mennyiségének módosítása.
* Termék eltávolítása: Egy termék teljes törlése a kosárból.
* Mennyiség csökkentése: Egy termék mennyiségének csökkentése, vagy eltávolítása, ha a mennyiség eléri az 1-et.

Hibakezelés

* Nem létező termék: Ha a megadott termék nem található a kosárban, a rendszer 404 Not Found hibát küld.
* Érvénytelen bemenet: Ha a bemeneti adatok nem felelnek meg a validációs szabályoknak, a rendszer 400 Bad Request hibát küld.
* Nem hitelesített felhasználó: Ha a felhasználó nincs bejelentkezve, a rendszer 401 Unauthorized hibát küld.

Jogosultságkezelés

* A CartController metódusai csak hitelesített felhasználók számára érhetők el. Az auth:sanctum middleware biztosítja, hogy csak bejelentkezett felhasználók férjenek hozzá a kosár funkcióihoz.

DeliveryController

A DeliveryController egy Laravel vezérlő, amely a szállítási adatok kezelésére szolgál. Ez a vezérlő biztosítja a szállítási adatok mentését az adatbázisba, valamint e-mail értesítések küldését a felhasználóknak a szállítási adatok megerősítéséről.

Metódusok

1. storeDelivery

Leírás: Szállítási adatok mentése és megerősítő e-mail küldése.

HTTP metódus: POST

URL: /api/delivery

Paraméterek:

* fullName (kötelező): A felhasználó teljes neve.
* email (kötelező): A felhasználó e-mail címe.
* address (kötelező): A szállítási cím.
* houseNumber (kötelező): A házszám.
* city (kötelező): A város.
* postalCode (kötelező): Az irányítószám.
* phone (kötelező): A felhasználó telefonszáma.

Funkciók:

* Ellenőrzi a bemeneti adatok érvényességét.
* Mentési műveletet hajt végre az adatbázisban a szállítási adatokkal.
* E-mail értesítést küld a felhasználónak a megadott e-mail címre a szállítási adatok megerősítéséről.

Válasz:

* Sikeres mentés és e-mail küldés esetén: Egy megerősítő üzenet.
* Hiba esetén: Validációs hibaüzenet vagy más megfelelő hibaüzenet.

Főbb Funkciók

* Adatvalidáció: A bemeneti adatok ellenőrzése, hogy megfeleljenek az elvárt formátumnak és típusnak.
* Adatbázis-műveletek: A szállítási adatok mentése az adatbázisba, beleértve a felhasználó nevét, címét, e-mail címét és telefonszámát.
* E-mail küldés: Automatikus e-mail értesítés küldése a felhasználónak a szállítási adatok megerősítéséről.

Hibakezelés

* Validációs hibák: Ha a bemeneti adatok nem felelnek meg a validációs szabályoknak (pl. hiányzó mezők, érvénytelen e-mail cím), a rendszer hibaüzenetet küld a felhasználónak.
* E-mail küldési hibák: Ha az e-mail küldése sikertelen, a rendszer naplózza a hibát, de a válaszban nem feltétlenül jelenik meg.
* Adatbázis hibák: Ha az adatok mentése sikertelen, a rendszer megfelelő hibaüzenetet küld.

Jogosultságkezelés

* A storeDelivery metódus eléréséhez nincs külön jogosultsági ellenőrzés megadva, így minden felhasználó, aki eléri az API-t, használhatja ezt a funkciót. Ha szükséges, jogosultsági ellenőrzést lehet hozzáadni az auth middleware segítségével.

E-mail Értesítések

* A szállítási adatok mentése után a rendszer automatikusan e-mail értesítést küld a felhasználónak a megadott e-mail címre. Az e-mail tartalmazza a szállítási adatok megerősítését, és a DeliveryConfirmationMail osztály segítségével kerül kiküldésre.

## ProductController

A ProductController egy Laravel vezérlő, amely a termékek kezelésére szolgál. Ez a vezérlő biztosítja a termékek létrehozását, frissítését, törlését, lekérdezését és listázását. A vezérlő támogatja a termékekhez tartozó képek feltöltését és tárolását is.

Metódusok

1. store

Leírás: Új termék létrehozása.

HTTP metódus: POST

URL: /api/products

Paraméterek:

* name (kötelező): A termék neve.
* price (kötelező): A termék ára.
* image (opcionális): A termékhez tartozó kép fájlja (formátum: jpg, jpeg, png, max. méret: 2 MB).

Funkciók:

* Validálja a bemeneti adatokat.
* Ha van feltöltött kép, elmenti azt a public/images mappába, és generál egy URL-t.
* Létrehozza a terméket az adatbázisban a megadott adatokkal.

Válasz:

* Sikeres létrehozás esetén a létrehozott termék adatai.
* Hiba esetén validációs hibaüzenet.

2. index

Leírás: Az összes termék listázása.

HTTP metódus: GET

URL: /api/products

Funkciók:

* Lekérdezi az összes terméket az adatbázisból.

Válasz:

* Az összes termék adatai JSON formátumban.
* Üres lista, ha nincs termék az adatbázisban.

3. update

Leírás: Egy meglévő termék adatainak frissítése.

HTTP metódus: PUT

URL: /api/products/{id}

Paraméterek:

* name (kötelező): A termék neve.
* price (kötelező): A termék ára.
* description (opcionális): A termék leírása.
* image (opcionális): A termékhez tartozó kép fájlja (formátum: jpg, jpeg, png, max. méret: 2 MB).

Funkciók:

* Megkeresi a megadott azonosítójú terméket az adatbázisban.
* Validálja a bemeneti adatokat.
* Frissíti a termék adatait az adatbázisban.

Válasz:

* Sikeres frissítés esetén a frissített termék adatai.
* Hiba esetén 404 Not Found, ha a termék nem található.

4. destroy

Leírás: Egy termék törlése.

HTTP metódus: DELETE

URL: /api/products/{id}

Funkciók:

* Megkeresi a megadott azonosítójú terméket az adatbázisban.
* Törli a terméket az adatbázisból.

Válasz:

* Sikeres törlés esetén egy megerősítő üzenet.
* Hiba esetén 404 Not Found, ha a termék nem található.

5. show

Leírás: Egy adott termék adatainak lekérdezése.

HTTP metódus: GET

URL: /api/products/{id}

Funkciók:

* Megkeresi a megadott azonosítójú terméket az adatbázisban.

Válasz:

* A termék adatai, ha megtalálható.
* Hiba esetén 404 Not Found, ha a termék nem található.
* Főbb Funkciók
* Termékek létrehozása: Új termékek hozzáadása az adatbázishoz, opcionálisan képfeltöltéssel.
* Termékek listázása: Az összes termék lekérdezése és megjelenítése.
* Termékek frissítése: Meglévő termékek adatainak módosítása, beleértve a képek frissítését is.
* Termékek törlése: Nem használt vagy hibás termékek eltávolítása az adatbázisból.
* Egyedi termék lekérdezése: Egy adott termék részletes adatainak megjelenítése.

Hibakezelés

* Validációs hibák: Ha a bemeneti adatok nem felelnek meg a validációs szabályoknak (pl. hiányzó mezők, érvénytelen fájlformátum), a rendszer hibaüzenetet küld.
* Nem létező termék: Ha a megadott azonosítójú termék nem található, a rendszer 404 Not Found hibát küld.
* Fájlfeltöltési hibák: Ha a kép feltöltése sikertelen, a rendszer naplózza a hibát, és nem menti el a terméket.

Jogosultságkezelés

* A vezérlő metódusaihoz nincs explicit jogosultsági ellenőrzés megadva. Ha szükséges, az auth middleware segítségével biztosítható, hogy csak hitelesített felhasználók férjenek hozzá a termékek kezeléséhez.

## UsersController

A UsersController egy Laravel vezérlő, amely a felhasználók kezelésére szolgál. Ez a vezérlő biztosítja a felhasználók regisztrációját, bejelentkezését, kijelentkezését, szerepkörök kezelését, valamint a felhasználói adatok frissítését és lekérdezését. A vezérlő többnyelvű válaszokat is támogat.

Metódusok

1. register

Leírás: Új felhasználó regisztrálása.

HTTP metódus: POST

URL: /api/register/{lang}

Paraméterek:

* name (kötelező): A felhasználó neve.
* email (kötelező): A felhasználó e-mail címe (egyedi).
* password (kötelező): A felhasználó jelszava (min. 6, max. 10 karakter, megerősítéssel).

Funkciók:

* Validálja a bemeneti adatokat.
* Létrehozza a felhasználót az adatbázisban.
* Az első regisztrált felhasználót szuperadminisztrátorrá lépteti elő.
* E-mail értesítést küld a regisztráció sikerességéről.
* Többnyelvű válaszokat biztosít (en, hu, zh).

Válasz:

* Sikeres regisztráció esetén megerősítő üzenet.
* Hiba esetén validációs hibaüzenet.

2. login

Leírás: Felhasználó bejelentkezése.

HTTP metódus: POST

URL: /api/login/{lang}

Paraméterek:

* email (kötelező): A felhasználó e-mail címe.
* password (kötelező): A felhasználó jelszava.

Funkciók:

* Ellenőrzi a megadott hitelesítési adatokat.
* Sikeres hitelesítés esetén létrehoz egy API tokent.
* Többnyelvű válaszokat biztosít (en, hu, zh).

Válasz:

* Sikeres bejelentkezés esetén az API token.
* Hiba esetén 401 Unauthorized hibaüzenet.

3. logout

Leírás: Felhasználó kijelentkezése.

HTTP metódus: POST

URL: /api/logout/{lang}

Funkciók:

* Kijelentkezteti a felhasználót az aktuális munkamenetből.
* Többnyelvű válaszokat biztosít (en, hu, zh).

Válasz:

* Sikeres kijelentkezés esetén megerősítő üzenet.

4. getUserRole

Leírás: A bejelentkezett felhasználó szerepkörének lekérdezése.

HTTP metódus: GET

URL: /api/user/role

Funkciók:

* Lekérdezi a bejelentkezett felhasználó szerepkörét.

Válasz:

* A felhasználó szerepköre (0: felhasználó, 1: admin, 2: szuperadmin).

5. getUserInfo

Leírás: A bejelentkezett felhasználó adatainak lekérdezése.

HTTP metódus: GET

URL: /api/user/info

Funkciók:

* Lekérdezi a bejelentkezett felhasználó adatait és szerepkörét.

Válasz:

* A felhasználó adatai és szerepköre.

6. upgradeToAdmin

Leírás: Egy felhasználó adminisztrátorrá léptetése.

HTTP metódus: POST

URL: /api/user/upgrade/{email}

Funkciók:

* Csak szuperadminisztrátor jogosult a műveletre.
* Az adott e-mail címhez tartozó felhasználót adminisztrátorrá lépteti elő.

Válasz:

* Sikeres előléptetés esetén megerősítő üzenet.
* Hiba esetén 401 Unauthorized vagy 400 Bad Request.

7. downgradeFromAdmin

Leírás: Egy adminisztrátor visszaminősítése normál felhasználóvá.

HTTP metódus: POST

URL: /api/user/downgrade/{email}

Funkciók:

* Csak szuperadminisztrátor jogosult a műveletre.
* Az adott e-mail címhez tartozó adminisztrátort visszaminősíti normál felhasználóvá.

Válasz:

* Sikeres visszaminősítés esetén megerősítő üzenet.
* Hiba esetén 401 Unauthorized vagy 400 Bad Request.

8. updateUserName

Leírás: A bejelentkezett felhasználó nevének frissítése.

HTTP metódus: PUT

URL: /api/user/update-name/{lang}

Paraméterek:

* name (kötelező): Az új név.

Funkciók:

* Frissíti a bejelentkezett felhasználó nevét.
* Többnyelvű válaszokat biztosít (en, hu, zh).

Válasz:

* Sikeres frissítés esetén megerősítő üzenet.
* Hiba esetén validációs hibaüzenet.

9. updatePassword

Leírás: A bejelentkezett felhasználó jelszavának frissítése.

HTTP metódus: PUT

URL: /api/user/update-password/{lang}

Paraméterek:

* new\_password (kötelező): Az új jelszó (min. 6, max. 10 karakter, megerősítéssel).

Funkciók:

* Frissíti a bejelentkezett felhasználó jelszavát.
* Többnyelvű válaszokat biztosít (en, hu, zh).

Válasz:

* Sikeres frissítés esetén megerősítő üzenet.
* Hiba esetén validációs hibaüzenet.

10. getUserList

Leírás: Az összes felhasználó listájának lekérdezése.

HTTP metódus: GET

URL: /api/users/{lang}

Funkciók:

* Lekérdezi az összes felhasználót az adatbázisból.
* Többnyelvű válaszokat biztosít (en, hu, zh).

Válasz:

* Az összes felhasználó adatai.

11. getUserByEmail

Leírás: Egy adott felhasználó adatainak lekérdezése e-mail cím alapján.

HTTP metódus: GET

URL: /api/user/{email}

Funkciók:

* Lekérdezi az adott e-mail címhez tartozó felhasználót.

Válasz:

* A felhasználó adatai, ha megtalálható.
* Hiba esetén 404 Not Found.

Főbb Funkciók

* Felhasználók regisztrációja és hitelesítése: Regisztráció, bejelentkezés, kijelentkezés.
* Szerepkörök kezelése: Felhasználók adminisztrátorrá léptetése vagy visszaminősítése.
* Felhasználói adatok frissítése: Név és jelszó módosítása.
* Felhasználók lekérdezése: Egyéni vagy összes felhasználó adatainak lekérdezése.

Hibakezelés

* Validációs hibák: Ha a bemeneti adatok nem felelnek meg a validációs szabályoknak, a rendszer hibaüzenetet küld.
* Nem hitelesített felhasználó: Ha a felhasználó nincs bejelentkezve, a rendszer 401 Unauthorized hibát küld.
* Nem létező felhasználó: Ha a megadott e-mail címhez nem található felhasználó, a rendszer 404 Not Found hibát küld.

Jogosultságkezelés

* A vezérlő metódusaihoz az auth middleware biztosítja, hogy csak hitelesített felhasználók férjenek hozzá.
* Az adminisztrátori és szuperadminisztrátori műveletekhez további jogosultsági ellenőrzések vannak beépítve.

Mail

## BookingConfirmationMail

A BookingConfirmationMail egy Laravel Mailable osztály, amely a foglalások megerősítésére szolgáló e-mailek kezelését végzi. Ez az osztály felelős az e-mail sablonhoz szükséges adatok átadásáért, az e-mail tárgyának beállításáért, valamint az e-mail sablon rendereléséért.

Funkciók

1. Konstruktor (\_\_construct)

Leírás: Az osztály példányosításakor a foglalási adatokat (bookingData) kapja meg, amelyeket az e-mail sablonhoz továbbít.

Bemenet:

* bookingData: Egy tömb, amely tartalmazza a foglalás részleteit (pl. dátum, időpont, felhasználó adatai).

Funkció:

* A kapott adatokat elérhetővé teszi az osztályon belül, hogy azokat az e-mail sablonban fel lehessen használni.

2. build

Leírás: Az e-mail felépítéséért felelős metódus.

Funkciók:

* Beállítja az e-mail tárgyát (Booking Confirmation).
* Meghatározza az e-mail sablont, amelyet a booking-confirmation.blade.php fájlban található nézetként renderel.
* Az e-mail sablonhoz továbbítja a foglalási adatokat (bookingData).

Kimenet:

* Egy teljesen felépített e-mail objektum, amely készen áll a küldésre.
* Főbb Funkciók
* Foglalási adatok kezelése: A foglalási adatok továbbítása az e-mail sablon számára.
* E-mail tárgyának beállítása: Az e-mail tárgya alapértelmezetten Booking Confirmation.
* E-mail sablon renderelése: Az e-mail tartalmának generálása a megadott nézet (Blade sablon) alapján.

Használat

* A BookingConfirmationMail osztályt a foglalások kezeléséért felelős vezérlő (pl. BookingController) használja. A foglalás létrehozása után a vezérlő példányosítja az osztályt, és átadja neki a foglalási adatokat, majd az e-mailt elküldi a felhasználónak.

Hibakezelés

* Hiányzó foglalási adatok: Ha a bookingData nem tartalmazza a szükséges adatokat, az e-mail sablon hibásan jelenhet meg. Ezért fontos, hogy a vezérlő megfelelően előkészítse az adatokat.
* Sablon hiánya: Ha a megadott nézetfájl (emails.booking-confirmation) nem létezik, a rendszer hibát dob. Győződj meg arról, hogy a sablon létezik és megfelelően van definiálva.

E-mail Sablon

* Az e-mail sablon a booking-confirmation.blade.php fájlban található. Ez a sablon felelős az e-mail tartalmának megjelenítéséért, és a bookingData változó segítségével jeleníti meg a foglalási adatokat.

## DeliveryConfirmationMail

A DeliveryConfirmationMail egy Laravel Mailable osztály, amely a szállítási adatok megerősítésére szolgáló e-mailek kezelését végzi. Ez az osztály felelős az e-mail sablonhoz szükséges adatok átadásáért, az e-mail tárgyának beállításáért, valamint az e-mail sablon rendereléséért.

Funkciók

1. Konstruktor (\_\_construct)

Leírás: Az osztály példányosításakor a szállítási adatokat (deliveryData) kapja meg, amelyeket az e-mail sablonhoz továbbít.

Bemenet:

* deliveryData: Egy tömb, amely tartalmazza a szállítási adatok részleteit (pl. név, cím, e-mail, telefonszám).

Funkció:

* A kapott adatokat elérhetővé teszi az osztályon belül, hogy azokat az e-mail sablonban fel lehessen használni.

2. build

Leírás: Az e-mail felépítéséért felelős metódus.

Funkciók:

* Beállítja az e-mail tárgyát (Delivery Confirmation).
* Meghatározza az e-mail sablont, amelyet a delivery-confirmation.blade.php fájlban található nézetként renderel.
* Az e-mail sablonhoz továbbítja a szállítási adatokat (deliveryData).

Kimenet:

* Egy teljesen felépített e-mail objektum, amely készen áll a küldésre.

Főbb Funkciók

* Szállítási adatok kezelése: A szállítási adatok továbbítása az e-mail sablon számára.
* E-mail tárgyának beállítása: Az e-mail tárgya alapértelmezetten Delivery Confirmation.
* E-mail sablon renderelése: Az e-mail tartalmának generálása a megadott nézet (Blade sablon) alapján.

Használat

* A DeliveryConfirmationMail osztályt a szállítási adatok kezeléséért felelős vezérlő (pl. DeliveryController) használja. A szállítási adatok mentése után a vezérlő példányosítja az osztályt, és átadja neki a szállítási adatokat, majd az e-mailt elküldi a felhasználónak.

Hibakezelés

* Hiányzó szállítási adatok: Ha a deliveryData nem tartalmazza a szükséges adatokat, az e-mail sablon hibásan jelenhet meg. Ezért fontos, hogy a vezérlő megfelelően előkészítse az adatokat.
* Sablon hiánya: Ha a megadott nézetfájl (emails.delivery-confirmation) nem létezik, a rendszer hibát dob. Győződj meg arról, hogy a sablon létezik és megfelelően van definiálva.

E-mail Sablon

* Az e-mail sablon a delivery-confirmation.blade.php fájlban található. Ez a sablon felelős az e-mail tartalmának megjelenítéséért, és a deliveryData változó segítségével jeleníti meg a szállítási adatokat.

## RegistrationSuccessful

A RegistrationSuccessful egy Laravel Mailable osztály, amely a felhasználók regisztrációjának sikerességéről szóló e-mailek kezelésére szolgál. Ez az osztály felelős az e-mail sablonhoz szükséges adatok átadásáért, az e-mail tárgyának beállításáért, valamint az e-mail sablon rendereléséért.

Funkciók

1. Konstruktor (\_\_construct)

Leírás: Az osztály példányosításakor a felhasználó adatait (user) kapja meg, amelyeket az e-mail sablonhoz továbbít.

Bemenet:

* user: Egy objektum vagy tömb, amely tartalmazza a regisztrált felhasználó adatait (pl. név, e-mail cím).

Funkció:

* A kapott adatokat elérhetővé teszi az osztályon belül, hogy azokat az e-mail sablonban fel lehessen használni.

2. build

Leírás: Az e-mail felépítéséért felelős metódus.

Funkciók:

* Beállítja az e-mail tárgyát (Sikeres regisztráció).
* Meghatározza az e-mail sablont, amelyet a registration-successful.blade.php fájlban található nézetként renderel.

Kimenet:

* Egy teljesen felépített e-mail objektum, amely készen áll a küldésre.

Főbb Funkciók

* Felhasználói adatok kezelése: A regisztrált felhasználó adatai továbbítása az e-mail sablon számára.
* E-mail tárgyának beállítása: Az e-mail tárgya alapértelmezetten Sikeres regisztráció.
* E-mail sablon renderelése: Az e-mail tartalmának generálása a megadott nézet (Blade sablon) alapján.

Használat

* A RegistrationSuccessful osztályt a felhasználók regisztrációjáért felelős vezérlő (pl. UsersController) használja. A regisztráció sikeres befejezése után a vezérlő példányosítja az osztályt, és átadja neki a felhasználó adatait, majd az e-mailt elküldi a felhasználónak.

Hibakezelés

* Hiányzó felhasználói adatok: Ha a user nem tartalmazza a szükséges adatokat, az e-mail sablon hibásan jelenhet meg. Ezért fontos, hogy a vezérlő megfelelően előkészítse az adatokat.
* Sablon hiánya: Ha a megadott nézetfájl (emails.registration-successful) nem létezik, a rendszer hibát dob. Győződj meg arról, hogy a sablon létezik és megfelelően van definiálva.

E-mail Sablon

* Az e-mail sablon a registration-successful.blade.php fájlban található. Ez a sablon felelős az e-mail tartalmának megjelenítéséért, és a user változó segítségével jeleníti meg a regisztrált felhasználó adatait.

## Modellek

## Booking

A Booking egy Laravel modell, amely a foglalások kezelésére szolgál. Ez a modell az adatbázisban található bookings táblához kapcsolódik, és lehetővé teszi a foglalások létrehozását, lekérdezését, frissítését és törlését.

Főbb Funkciók

* Adatbázis-kezelés: A Booking modell az Eloquent ORM segítségével biztosítja a foglalások adatbázisban történő kezelését.
* Tömeges kitöltés támogatása: A modell lehetővé teszi a foglalások egyszerű létrehozását és frissítését a fillable mezők használatával.
* Adatstruktúra biztosítása: Meghatározza, hogy mely mezők tartoznak egy foglaláshoz (pl. dátum, időpont, nem, e-mail cím).
* Adatbázis Mezők

A modell az alábbi mezőket kezeli az adatbázisban:

* date: A foglalás dátuma.
* time: A foglalás időpontja.
* gender: A foglaló személy neme.
* email: A foglaló e-mail címe.
* Tömeges Kitöltés

A fillable tulajdonság segítségével meghatározható, hogy mely mezők tölthetők ki tömegesen. Ez biztosítja, hogy csak a megadott mezők frissíthetők vagy hozhatók létre egyszerű tömbök használatával. A Booking modellben a következő mezők vannak engedélyezve:

* date
* time
* gender
* email

Ez megakadályozza, hogy illetéktelen mezők kerüljenek az adatbázisba.

Kapcsolatok

* A jelenlegi implementációban a Booking modell nem tartalmaz más modellekhez kapcsolódó relációkat. Ha a foglalások más entitásokhoz (pl. felhasználókhoz vagy helyszínekhez) kapcsolódnak, a relációkat a modellben definiálni kell, például belongsTo, hasMany, vagy hasOne relációk segítségével.

Használat

* A Booking modell az Eloquent ORM segítségével használható a foglalások kezelésére. Az alábbi műveletek végezhetők el vele:

Foglalás létrehozása:

* Új foglalás hozzáadása az adatbázishoz a create metódus segítségével.
* Foglalások lekérdezése:
* Foglalások lekérdezése az all, find, vagy where metódusokkal.
* Foglalás frissítése:
* Egy meglévő foglalás adatainak módosítása.

Foglalás törlése:

* Egy foglalás eltávolítása az adatbázisból.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a foglalás létrehozásakor, a rendszer validációs hibát dob.
* Érvénytelen adatok: Ha a foglalás mezői nem felelnek meg az elvárt formátumnak (pl. érvénytelen dátum vagy e-mail cím), a rendszer hibát generál.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a rendszer megfelelő hibakezelést biztosít.

## Cart

A Cart egy Laravel modell, amely a felhasználók kosarának kezelésére szolgál. Ez a modell az adatbázisban található carts táblához kapcsolódik, és lehetővé teszi a kosárban lévő termékek kezelését, például azok hozzáadását, frissítését és eltávolítását.

Főbb Funkciók

* Adatbázis-kezelés: A Cart modell az Eloquent ORM segítségével biztosítja a kosár adatainak kezelését.
* Tömeges kitöltés támogatása: A modell lehetővé teszi a kosárbejegyzések egyszerű létrehozását és frissítését a fillable mezők használatával.
* Kapcsolatok kezelése: A modell relációkat definiál a felhasználók (User) és a termékek (Product) modellekkel.

Adatbázis Mezők

A modell az alábbi mezőket kezeli az adatbázisban:

* user\_id: A kosárhoz tartozó felhasználó azonosítója.
* product\_id: A kosárban lévő termék azonosítója.
* quantity: A termék mennyisége a kosárban.

Tömeges Kitöltés

* A fillable tulajdonság segítségével meghatározható, hogy mely mezők tölthetők ki tömegesen. Ez biztosítja, hogy csak a megadott mezők frissíthetők vagy hozhatók létre egyszerű tömbök használatával. A Cart modellben a következő mezők vannak engedélyezve:
* user\_id
* product\_id
* quantity

Ez megakadályozza, hogy illetéktelen mezők kerüljenek az adatbázisba.

Kapcsolatok

A Cart modell relációkat definiál más modellekkel:

Felhasználó (user):

* A kosár egy adott felhasználóhoz tartozik.
* Reláció típusa: belongsTo.
* Termék (product):
* A kosár egy adott terméket tartalmaz.
* Reláció típusa: belongsTo.

Ezek a relációk lehetővé teszik, hogy a kosárhoz tartozó felhasználó és termék adatai könnyen lekérdezhetők legyenek.

Használat

* A Cart modell az Eloquent ORM segítségével használható a kosár adatainak kezelésére. Az alábbi műveletek végezhetők el vele:

Kosárbejegyzés létrehozása:

* Új kosárbejegyzés hozzáadása az adatbázishoz a create metódus segítségével.

Kosárbejegyzések lekérdezése:

* Kosárbejegyzések lekérdezése az all, find, vagy where metódusokkal.

Kosárbejegyzés frissítése:

* Egy meglévő kosárbejegyzés adatainak módosítása.

Kosárbejegyzés törlése:

* Egy kosárbejegyzés eltávolítása az adatbázisból.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a kosárbejegyzés létrehozásakor, a rendszer validációs hibát dob.
* Érvénytelen adatok: Ha a kosár mezői nem felelnek meg az elvárt formátumnak (pl. érvénytelen azonosítók vagy mennyiség), a rendszer hibát generál.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a rendszer megfelelő hibakezelést biztosít.

## Delivery

A Delivery egy Laravel modell, amely a szállítási adatok kezelésére szolgál. Ez a modell az adatbázisban található deliveries táblához kapcsolódik, és lehetővé teszi a szállítási adatok létrehozását, lekérdezését, frissítését és törlését.

Főbb Funkciók

* Adatbázis-kezelés: A Delivery modell az Eloquent ORM segítségével biztosítja a szállítási adatok kezelését.
* Tömeges kitöltés támogatása: A modell lehetővé teszi a szállítási adatok egyszerű létrehozását és frissítését a fillable mezők használatával.
* Adatstruktúra biztosítása: Meghatározza, hogy mely mezők tartoznak egy szállítási rekordhoz (pl. név, cím, e-mail, telefonszám).

Adatbázis Mezők

A modell az alábbi mezőket kezeli az adatbázisban:

* full\_name: A felhasználó teljes neve.
* email: A felhasználó e-mail címe.
* address: A szállítási cím.
* house\_number: A házszám.
* city: A város.
* postal\_code: Az irányítószám.
* phone: A felhasználó telefonszáma.

Tömeges Kitöltés

A fillable tulajdonság segítségével meghatározható, hogy mely mezők tölthetők ki tömegesen. Ez biztosítja, hogy csak a megadott mezők frissíthetők vagy hozhatók létre egyszerű tömbök használatával. A Delivery modellben a következő mezők vannak engedélyezve:

* full\_name
* email
* address
* house\_number
* city
* postal\_code
* phone
* Ez megakadályozza, hogy illetéktelen mezők kerüljenek az adatbázisba.

Kapcsolatok

* A jelenlegi implementációban a Delivery modell nem tartalmaz más modellekhez kapcsolódó relációkat. Ha a szállítási adatok más entitásokhoz (pl. rendelésekhez vagy felhasználókhoz) kapcsolódnak, a relációkat a modellben definiálni kell, például belongsTo, hasMany, vagy hasOne relációk segítségével.

Használat

* A Delivery modell az Eloquent ORM segítségével használható a szállítási adatok kezelésére. Az alábbi műveletek végezhetők el vele:

Szállítási adat létrehozása:

* Új szállítási rekord hozzáadása az adatbázishoz a create metódus segítségével.

Szállítási adatok lekérdezése:

* Szállítási rekordok lekérdezése az all, find, vagy where metódusokkal.

Szállítási adat frissítése:

* Egy meglévő szállítási rekord adatainak módosítása.

Szállítási adat törlése:

* Egy szállítási rekord eltávolítása az adatbázisból.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a szállítási adat létrehozásakor, a rendszer validációs hibát dob.
* Érvénytelen adatok: Ha a szállítási mezők nem felelnek meg az elvárt formátumnak (pl. érvénytelen e-mail cím vagy telefonszám), a rendszer hibát generál.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a rendszer megfelelő hibakezelést biztosít.

## Product

A Product egy Laravel modell, amely a termékek kezelésére szolgál. Ez a modell az adatbázisban található products táblához kapcsolódik, és lehetővé teszi a termékek létrehozását, lekérdezését, frissítését és törlését.

Főbb Funkciók

* Adatbázis-kezelés: A Product modell az Eloquent ORM segítségével biztosítja a termékek adatainak kezelését.
* Tömeges kitöltés támogatása: A modell lehetővé teszi a termékek egyszerű létrehozását és frissítését a fillable mezők használatával.
* Adatstruktúra biztosítása: Meghatározza, hogy mely mezők tartoznak egy termékhez (pl. név, ár, leírás, kép URL).

Adatbázis Mezők

A modell az alábbi mezőket kezeli az adatbázisban:

* name: A termék neve.
* description: A termék leírása.
* price: A termék ára.
* imageURL: A termékhez tartozó kép URL-je.

Tömeges Kitöltés

A fillable tulajdonság segítségével meghatározható, hogy mely mezők tölthetők ki tömegesen. Ez biztosítja, hogy csak a megadott mezők frissíthetők vagy hozhatók létre egyszerű tömbök használatával. A Product modellben a következő mezők vannak engedélyezve:

* name
* description
* price
* imageURL

Ez megakadályozza, hogy illetéktelen mezők kerüljenek az adatbázisba.

Kapcsolatok

* A jelenlegi implementációban a Product modell nem tartalmaz más modellekhez kapcsolódó relációkat. Ha a termékek más entitásokhoz (pl. kategóriákhoz vagy rendelésekhez) kapcsolódnak, a relációkat a modellben definiálni kell, például belongsTo, hasMany, vagy hasOne relációk segítségével.

Használat

* A Product modell az Eloquent ORM segítségével használható a termékek adatainak kezelésére. Az alábbi műveletek végezhetők el vele:

Termék létrehozása:

* Új termék hozzáadása az adatbázishoz a create metódus segítségével.

Termékek lekérdezése:

* Termékek lekérdezése az all, find, vagy where metódusokkal.

Termék frissítése:

* Egy meglévő termék adatainak módosítása.

Termék törlése:

* Egy termék eltávolítása az adatbázisból.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a termék létrehozásakor, a rendszer validációs hibát dob.
* Érvénytelen adatok: Ha a termék mezői nem felelnek meg az elvárt formátumnak (pl. érvénytelen ár vagy URL), a rendszer hibát generál.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a rendszer megfelelő hibakezelést biztosít.

## User

A User egy Laravel modell, amely a felhasználók kezelésére szolgál. Ez a modell az adatbázisban található users táblához kapcsolódik, és lehetővé teszi a felhasználók adatainak kezelését, például azok létrehozását, frissítését, lekérdezését és törlését. A modell emellett szerepkör-alapú jogosultságkezelést is biztosít.

Főbb Funkciók

* Adatbázis-kezelés: A User modell az Eloquent ORM segítségével biztosítja a felhasználók adatainak kezelését.
* Tömeges kitöltés támogatása: A modell lehetővé teszi a felhasználók egyszerű létrehozását és frissítését a fillable mezők használatával.
* Szerepkör-kezelés: A modell támogatja a felhasználók szerepkör-alapú jogosultságainak kezelését (pl. felhasználó, adminisztrátor, szuperadminisztrátor).
* Kapcsolatok kezelése: A modell relációkat definiál más modellekkel, például a kosár (Cart) modellel.

Adatbázis Mezők

A modell az alábbi mezőket kezeli az adatbázisban:

* name: A felhasználó neve.
* email: A felhasználó e-mail címe.
* password: A felhasználó jelszava (titkosítva tárolva).
* role: A felhasználó szerepköre (0: normál felhasználó, 1: adminisztrátor, 2: szuperadminisztrátor).

Tömeges Kitöltés

A fillable tulajdonság segítségével meghatározható, hogy mely mezők tölthetők ki tömegesen. Ez biztosítja, hogy csak a megadott mezők frissíthetők vagy hozhatók létre egyszerű tömbök használatával. A User modellben a következő mezők vannak engedélyezve:

* name
* email
* password
* role

Ez megakadályozza, hogy illetéktelen mezők kerüljenek az adatbázisba.

Rejtett Mezők

A hidden tulajdonság segítségével meghatározható, hogy mely mezők legyenek elrejtve a JSON válaszokban. A User modellben az alábbi mezők vannak elrejtve:

* password
* remember\_token

Ez biztosítja, hogy érzékeny adatok ne kerüljenek ki a válaszokban.

Típuskonverziók

A casts tulajdonság segítségével meghatározható, hogy bizonyos mezők milyen típusra legyenek konvertálva. A User modellben az alábbi típuskonverziók vannak megadva:

* email\_verified\_at: datetime
* password: hashed

Ez biztosítja, hogy a jelszavak automatikusan titkosítva legyenek, és az időbélyegek megfelelő formátumban jelenjenek meg.

Kapcsolatok

A User modell relációkat definiál más modellekkel:

Kosár (cart):

* Egy felhasználó több kosárbejegyzéssel rendelkezhet.
* Reláció típusa: hasMany.

Szerepkör-kezelés

* A modell szerepkör-alapú jogosultságkezelést biztosít az alábbi metódusok segítségével:
* isAdmin:
* Ellenőrzi, hogy a felhasználó adminisztrátor-e (szerepkör: 1).

isSuperAdmin:

* Ellenőrzi, hogy a felhasználó szuperadminisztrátor-e (szerepkör: 2).

isUser:

* Ellenőrzi, hogy a felhasználó normál felhasználó-e (szerepkör: 0).
* Ezek a metódusok egyszerűsítik a jogosultságok ellenőrzését a vezérlőkben és más alkalmazásrészekben.

Használat

* A User modell az Eloquent ORM segítségével használható a felhasználók adatainak kezelésére. Az alábbi műveletek végezhetők el vele:

Felhasználó létrehozása:

* Új felhasználó hozzáadása az adatbázishoz a create metódus segítségével.

Felhasználók lekérdezése:

* Felhasználók lekérdezése az all, find, vagy where metódusokkal.

Felhasználó frissítése:

* Egy meglévő felhasználó adatainak módosítása.

Felhasználó törlése:

* Egy felhasználó eltávolítása az adatbázisból.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a felhasználó létrehozásakor, a rendszer validációs hibát dob.
* Érvénytelen adatok: Ha a felhasználó mezői nem felelnek meg az elvárt formátumnak (pl. érvénytelen e-mail cím), a rendszer hibát generál.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a rendszer megfelelő hibakezelést biztosít.

## Migrációk

## add**\_**role**\_**to**\_**user**\_**table

Ez a migrációs fájl a users tábla módosítására szolgál, amely során egy új role oszlop kerül hozzáadásra. Az oszlop a felhasználók szerepkörét tárolja, és alapértelmezett értéke 0.

Főbb Funkciók

* Szerepkör-kezelés támogatása:
* A role oszlop lehetővé teszi a felhasználók szerepkörének tárolását és kezelését (pl. normál felhasználó, adminisztrátor, szuperadminisztrátor).

Alapértelmezett érték:

* Az oszlop alapértelmezett értéke 0, amely a normál felhasználót jelöli.
* Táblamódosítás
* role Oszlop
* Típus: Egész szám (integer).
* Alapértelmezett érték: 0.

Funkció:

* Tárolja a felhasználó szerepkörét:
* 0: Normál felhasználó.
* 1: Adminisztrátor.
* 2: Szuperadminisztrátor.

Hibakezelés

* Hiányzó oszlop: Ha a migráció visszavonása után újra futtatjuk, a role oszlop ismét hozzáadásra kerül.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Alapértelmezett érték hiánya: Ha az oszlop alapértelmezett értéke nem kerül beállításra, a rendszer hibát dobhat, amikor új rekordot hozunk létre.

Használat

Migráció futtatása:

* A migráció futtatásával a users tábla módosul, és hozzáadódik a role oszlop.

Migráció visszavonása:

* A migráció visszavonásával a role oszlop eltávolításra kerül a users táblából.

## create**\_**bookings**\_**table

Ez a migrációs fájl a bookings tábla létrehozásáért felelős. A tábla a foglalások kezelésére szolgál, és tárolja a foglalások alapvető adatait, például a dátumot és az időpontot.

Főbb Funkciók

Foglalások tárolása:

* A bookings tábla lehetővé teszi a foglalások adatainak tárolását, például a dátumot és az időpontot.

Időbélyegek kezelése:

* A tábla automatikusan tárolja a rekordok létrehozásának és utolsó frissítésének időpontját.

Táblaszerkezet

* bookings Tábla

Mezők:

* id: Egyedi azonosító (automatikusan növekvő).
* date: A foglalás dátuma.
* time: A foglalás időpontja.
* created\_at és updated\_at: Az adatok létrehozásának és utolsó frissítésének időbélyegei.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a foglalás létrehozásakor, a rendszer validációs hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Táblanév ütközés: Ha már létezik egy bookings nevű tábla az adatbázisban, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával a bookings tábla létrejön az adatbázisban.

Migráció visszavonása:

* A migráció visszavonásával a bookings tábla törlődik az adatbázisból.

## create**\_**products**\_**table

Ez a migrációs fájl a products tábla létrehozásáért felelős. A tábla a termékek kezelésére szolgál, és tárolja a termékek alapvető adatait, például a nevüket, leírásukat, árukat és a hozzájuk tartozó képek URL-jét.

Főbb Funkciók

Termékek tárolása:

* A products tábla lehetővé teszi a termékek adatainak tárolását, például a nevüket, árukat és opcionálisan a leírásukat és képeik URL-jét.

Időbélyegek kezelése:

* A tábla automatikusan tárolja a rekordok létrehozásának és utolsó frissítésének időpontját.
* Táblaszerkezet
* products Tábla

Mezők:

* id: Egyedi azonosító (automatikusan növekvő).
* name: A termék neve.
* description: A termék leírása (opcionális).
* price: A termék ára, legfeljebb 8 számjegyű, 2 tizedesjegy pontossággal.
* imageURL: A termékhez tartozó kép URL-je (opcionális).
* created\_at és updated\_at: Az adatok létrehozásának és utolsó frissítésének időbélyegei.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a termék létrehozásakor, a rendszer validációs hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Táblanév ütközés: Ha már létezik egy products nevű tábla az adatbázisban, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával a products tábla létrejön az adatbázisban.

Migráció visszavonása:

* A migráció visszavonásával a products tábla törlődik az adatbázisból.

## add**\_**time**\_**to**\_**bookings**\_**table

Ez a migrációs fájl a bookings tábla módosítására szolgál, amely során egy új time oszlop kerül hozzáadásra. Az oszlop a foglalások időpontját tárolja, lehetővé téve a pontosabb időbeli nyilvántartást.

Főbb Funkciók

* Időpontok tárolása:
* A time oszlop lehetővé teszi a foglalások pontos időpontjának rögzítését.

Táblamódosítás:

* A migráció hozzáadja a time oszlopot a bookings táblához a date oszlop után.

Táblaszerkezet Módosítása

* time Oszlop
* Típus: Idő (time).
* Elhelyezkedés: A date oszlop után kerül hozzáadásra.

Funkció:

* Tárolja a foglalás pontos időpontját.

Hibakezelés

* Hiányzó oszlop: Ha a migráció futtatása előtt a bookings tábla nem létezik, a rendszer hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Duplikált oszlop: Ha a time oszlop már létezik a bookings táblában, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával a time oszlop hozzáadásra kerül a bookings táblához.

Migráció visszavonása:

* A migráció visszavonásával a time oszlop eltávolításra kerül a bookings táblából.

Megjegyzés

* A jelenlegi kód alapján a up metódus kommentelve van, így a migráció nem hajt végre módosítást. A migráció használatához a kommenteket el kell távolítani, és a kódot aktiválni kell.

## add**\_**gender**\_**to**\_**bookings**\_**table

Ez a migrációs fájl a bookings tábla módosítására szolgál, amely során egy új gender oszlop kerül hozzáadásra. Az oszlop a foglalást végző személy nemét tárolja.

Főbb Funkciók

Nem tárolása:

* A gender oszlop lehetővé teszi a foglalást végző személy nemének rögzítését.

Táblamódosítás:

* A migráció hozzáadja a gender oszlopot a bookings táblához a time oszlop után.
* Táblaszerkezet Módosítása
* gender Oszlop
* Típus: Szöveg (string).
* Elhelyezkedés: A time oszlop után kerül hozzáadásra.

Hibakezelés

* Hiányzó tábla: Ha a bookings tábla nem létezik a migráció futtatásakor, a rendszer hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Duplikált oszlop: Ha a gender oszlop már létezik a bookings táblában, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával a gender oszlop hozzáadásra kerül a bookings táblához.

Migráció visszavonása:

* A migráció visszavonásával a gender oszlop eltávolításra kerül a bookings táblából.

## create**\_**deliveries**\_**table

Ez a migrációs fájl a deliveries tábla létrehozásáért felelős. A tábla a szállítási adatok kezelésére szolgál, és tárolja a szállításhoz szükséges információkat, például a címzett nevét, címét, e-mail címét és telefonszámát.

Főbb Funkciók

Szállítási adatok tárolása:

* A deliveries tábla lehetővé teszi a szállítási adatok rögzítését, például a címzett nevét, címét és elérhetőségeit.

Időbélyegek kezelése:

* A tábla automatikusan tárolja a rekordok létrehozásának és utolsó frissítésének időpontját.

Táblaszerkezet

* deliveries Tábla

Mezők:

* id: Egyedi azonosító (automatikusan növekvő).
* full\_name: A címzett teljes neve.
* email: A címzett e-mail címe.
* address: A szállítási cím.
* house\_number: A házszám.
* city: A város.
* postal\_code: Az irányítószám.
* phone: A címzett telefonszáma.
* created\_at és updated\_at: Az adatok létrehozásának és utolsó frissítésének időbélyegei.

Hibakezelés

* Hiányzó mezők: Ha egy kötelező mező hiányzik a szállítási adat létrehozásakor, a rendszer validációs hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Táblanév ütközés: Ha már létezik egy deliveries nevű tábla az adatbázisban, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával a deliveries tábla létrejön az adatbázisban.

Migráció visszavonása:

* A migráció visszavonásával a deliveries tábla törlődik az adatbázisból.

## add**\_**email**\_**to**\_**bookings**\_**table

Ez a migrációs fájl a bookings tábla módosítására szolgál, amely során egy új email oszlop kerül hozzáadásra. Az oszlop a foglalást végző személy e-mail címét tárolja.

Főbb Funkciók

* E-mail cím tárolása:
* Az email oszlop lehetővé teszi a foglalást végző személy e-mail címének rögzítését.

Táblamódosítás:

* A migráció hozzáadja az email oszlopot a bookings táblához a gender oszlop után.
* Táblaszerkezet Módosítása
* email Oszlop
* Típus: Szöveg (string).
* Elhelyezkedés: A gender oszlop után kerül hozzáadásra.

Funkció:

* Tárolja a foglalást végző személy e-mail címét, amely később felhasználható értesítések küldésére vagy azonosításra.

Hibakezelés

* Hiányzó tábla: Ha a bookings tábla nem létezik a migráció futtatásakor, a rendszer hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Duplikált oszlop: Ha az email oszlop már létezik a bookings táblában, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával az email oszlop hozzáadásra kerül a bookings táblához.

Migráció visszavonása:

* A migráció visszavonásával az email oszlop eltávolításra kerül a bookings táblából.

## create**\_**carts**\_**table

Ez a migrációs fájl a carts tábla létrehozásáért felelős. A tábla a felhasználók kosarának kezelésére szolgál, és tárolja a kosárban lévő termékek adatait, például a felhasználó azonosítóját, a termék azonosítóját és a mennyiséget.

Főbb Funkciók

Kosár adatok tárolása:

* A carts tábla lehetővé teszi a felhasználók kosarának tartalmának rögzítését, beleértve a termékeket és azok mennyiségét.

Kapcsolatok kezelése:

* A tábla relációkat definiál a users és products táblákkal, biztosítva a kosár elemeinek összekapcsolását a felhasználókkal és termékekkel.

Időbélyegek kezelése:

* A tábla automatikusan tárolja a rekordok létrehozásának és utolsó frissítésének időpontját.

Táblaszerkezet

* carts Tábla

Mezők:

* id: Egyedi azonosító (automatikusan növekvő).
* user\_id: A kosárhoz tartozó felhasználó azonosítója (idegen kulcs a users táblára).
* product\_id: A kosárban lévő termék azonosítója (idegen kulcs a products táblára).
* quantity: A termék mennyisége a kosárban (alapértelmezett érték: 1).
* created\_at és updated\_at: Az adatok létrehozásának és utolsó frissítésének időbélyegei.

Kapcsolatok

* user\_id:
* Kapcsolat a users táblával.
* Ha egy felhasználó törlésre kerül, a hozzá tartozó kosárbejegyzések is törlődnek (onDelete('cascade')).
* product\_id:
* Kapcsolat a products táblával.
* Ha egy termék törlésre kerül, a hozzá tartozó kosárbejegyzések is törlődnek (onDelete('cascade')).

Hibakezelés

* Hiányzó tábla: Ha a users vagy products tábla nem létezik a migráció futtatásakor, a rendszer hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy más adatbázis-hiba lép fel, a migráció sikertelen lesz.
* Táblanév ütközés: Ha már létezik egy carts nevű tábla az adatbázisban, a migráció nem fog lefutni.

Használat

Migráció futtatása:

* A migráció futtatásával a carts tábla létrejön az adatbázisban, és a relációk beállításra kerülnek.

Migráció visszavonása:

* A migráció visszavonásával a carts tábla törlődik az adatbázisból.

Seederek

## DatabaseSeeder

A DatabaseSeeder osztály a Laravel adatbázis-seederek központi osztálya, amely az adatbázis feltöltésére szolgál teszt- vagy alapértelmezett adatokkal. Ez az osztály lehetővé teszi, hogy a különböző modellekhez tartozó seederek egyszerűen futtathatók legyenek.

Főbb Funkciók

Adatbázis feltöltése:

* A DatabaseSeeder osztály a run metódusban meghatározott utasítások alapján tölti fel az adatbázist.

Felhasználók létrehozása:

* A jelenlegi implementáció egy tesztfelhasználót hoz létre a users táblában.

Factory-k használata:

* A seederek támogatják a Laravel gyári osztályait, amelyekkel tömegesen generálhatók tesztadatok.

Jelenlegi Implementáció

Tesztfelhasználó létrehozása:

* Egy Test User nevű felhasználó kerül létrehozásra a users táblában az alábbi adatokkal:
* Név: Test User
* E-mail: test@example.com
* Ez a felhasználó hasznos lehet az alkalmazás tesztelése során.
* Tömeges felhasználó-generálás (kommentelve):

A kód tartalmaz egy kommentelt példát, amely 10 véletlenszerű felhasználót hozna létre a User gyár segítségével. Ez a funkció aktiválható, ha szükséges.

Használat

Seeder futtatása:

* A DatabaseSeeder osztály futtatásával a meghatározott adatok bekerülnek az adatbázisba.
* Parancs: php artisan db:seed

Tömeges adatgenerálás:

* A kommentelt kód aktiválásával tömegesen generálhatók felhasználók a gyárak segítségével.

Hibakezelés

* Hiányzó gyárak: Ha a User modellhez tartozó gyár nincs definiálva, a rendszer hibát dob.
* Adatbázis hibák: Ha az adatbázis nem érhető el, vagy a táblák nem léteznek, a seeder futtatása sikertelen lesz.
* Egyediség megsértése: Ha a létrehozott adatok megsértik az adatbázis egyedi kulcsainak szabályait (pl. e-mail cím), a seeder hibát dob.

Testreszabás

További seederek hozzáadása:

* A DatabaseSeeder osztályban további seederek hívhatók meg a call metódus segítségével.
* Példa: $this->call(ProductSeeder::class);

Adatok módosítása:

* A létrehozott tesztfelhasználó adatai (pl. név, e-mail) könnyen módosíthatók a run metódusban.

## Útvonalak

**api**

Az api.php fájl a Laravel API útvonalainak definiálására szolgál. Ez a fájl tartalmazza az alkalmazás különböző funkcióihoz tartozó végpontokat, amelyek a vezérlőkön keresztül hívhatók meg. Az útvonalak egy része hitelesítést igényel, míg mások nyilvánosan elérhetők.

Főbb Funkciók

* Felhasználói műveletek:

Regisztráció, bejelentkezés, kijelentkezés.

* Felhasználói adatok frissítése (név, jelszó).
* Felhasználók listázása és szerepkörök kezelése.

Foglalások kezelése:

* Foglalások ellenőrzése, létrehozása és lekérdezése.
* Felhasználói foglalások és összes foglalás listázása.

Termékek kezelése:

* Termékek listázása, létrehozása, frissítése és törlése.

Szállítási adatok kezelése:

* Szállítási adatok mentése.

Kosár kezelése:

* Kosár tartalmának lekérdezése, termékek hozzáadása, frissítése és törlése.

Útvonalak

1. Felhasználói műveletek

Nyilvános végpontok:

* POST /{lang}/register: Felhasználó regisztrációja.
* POST /{lang}/login: Felhasználó bejelentkezése.
* POST /{lang}/logout: Felhasználó kijelentkezése.

Hitelesítést igénylő végpontok:

* POST /{lang}/update-username: Felhasználó nevének frissítése.
* POST /{lang}/update-password: Felhasználó jelszavának frissítése.
* GET /{lang}/getUserList: Felhasználók listázása.
* GET /user: Bejelentkezett felhasználó adatainak lekérdezése.
* PUT /users/{email}/upgrade: Felhasználó adminisztrátorrá léptetése.
* PUT /users/{email}/downgrade: Adminisztrátor visszaminősítése normál felhasználóvá.
* POST /checkoldpassword: Régi jelszó ellenőrzése.
* GET /getUserByEmail/{email}: Felhasználó adatainak lekérdezése e-mail alapján.

2. Foglalások kezelése

Nyilvános végpontok:

* GET /check-booking: Foglalás ellenőrzése adott dátumra és időpontra.
* POST /check-booking/store: Új foglalás létrehozása.
* GET /check-booking/booked-dates: Foglalt dátumok lekérdezése.
* Hitelesítést igénylő végpontok:
* GET /user-bookings: Bejelentkezett felhasználó foglalásainak lekérdezése.
* GET /bookings: Foglalások listázása.
* GET /all-bookings: Összes foglalás lekérdezése (adminisztrátorok számára).

3. Termékek kezelése

Nyilvános végpontok:

* GET /products: Összes termék listázása.
* GET /products/{id}: Egy adott termék adatainak lekérdezése.

Hitelesítést igénylő végpontok:

* POST /products: Új termék létrehozása.
* PUT /products/{id}: Termék adatainak frissítése.
* DELETE /products/{id}: Termék törlése.

4. Szállítási adatok kezelése

Nyilvános végpontok:

* POST /delivery: Szállítási adatok mentése.

5. Kosár kezelése

Hitelesítést igénylő végpontok:

* GET /cart: Kosár tartalmának lekérdezése.
* POST /cart: Termék hozzáadása a kosárhoz.
* PUT /cart/{id}: Kosárbejegyzés frissítése.
* DELETE /cartdelete/{id}: Termék eltávolítása a kosárból.
* POST /cart/decrease/{productId}: Termék mennyiségének csökkentése a kosárban.

Hitelesítés

* A hitelesítést igénylő végpontok a Sanctum middleware segítségével védettek. Csak bejelentkezett felhasználók férhetnek hozzá ezekhez az útvonalakhoz.
* A Sanctum biztosítja az API token alapú hitelesítést.

Nyelvi Támogatás

* Az útvonalak egy része nyelvi prefixet használ ({lang}), amely lehetővé teszi a többnyelvű válaszokat (pl. en, hu, zh).

Hibakezelés

* Nem létező végpont: Ha egy végpont nem létezik, a rendszer 404 Not Found hibát küld.
* Hitelesítés hiánya: Ha egy védett végpontot hitelesítés nélkül próbálnak elérni, a rendszer 401 Unauthorized hibát küld.
* Jogosultság hiánya: Ha egy felhasználó nem rendelkezik megfelelő jogosultsággal, a rendszer 403 Forbidden hibát küld.

## web

A web.php fájl a Laravel alkalmazás webes útvonalainak definiálására szolgál. Ezek az útvonalak a böngészőből érhetők el, és általában HTML nézeteket vagy egyszerű szöveges válaszokat adnak vissza. A jelenlegi implementáció tartalmaz alapértelmezett nézeteket, e-mail küldési teszteket és egy e-mail sablon megjelenítését.

Főbb Funkciók

* Alapértelmezett nézet megjelenítése:
* A gyökér (/) útvonal a Laravel alapértelmezett welcome nézetét jeleníti meg.

E-mail küldési tesztek:

* Teszt e-mail küldése egyszerű szöveges tartalommal.
* Teszt e-mail küldése a Laravel queue rendszerének használatával.

E-mail sablon megjelenítése:

* Egy regisztrációs sikerességet jelző e-mail sablon megtekintése a böngészőben.

Útvonalak

1. Alapértelmezett nézet

Útvonal: /

Funkció:

* A Laravel alapértelmezett welcome nézetét jeleníti meg.

Használat:

* Böngészőben megnyitva az alkalmazás kezdőlapját jeleníti meg.

2. Teszt e-mail küldése

Útvonal: /test-email

Funkció:

* Egy egyszerű szöveges e-mail küldése a megadott címzettnek.

Válasz:

* Egy szöveges üzenet (E-mail elküldve!), amely jelzi, hogy az e-mail sikeresen elküldésre került.

3. Teszt e-mail küldése queue használatával

Útvonal: /test-queue

Funkció:

* Egy egyszerű szöveges e-mail hozzáadása a Laravel queue rendszeréhez.

Válasz:

* Egy szöveges üzenet (E-mail hozzáadva a queue-hoz!), amely jelzi, hogy az e-mail sikeresen hozzáadódott a queue-hoz.

4. E-mail sablon megjelenítése

Útvonal: /test-email-template

Funkció:

* Egy regisztrációs sikerességet jelző e-mail sablon megjelenítése a böngészőben.

Használat:

* A sablon megtekintésére szolgál, amely segíti a sablon fejlesztését és tesztelését.

Hibakezelés

E-mail küldési hibák:

* Ha az e-mail küldése sikertelen (pl. rossz konfiguráció vagy nem elérhető e-mail szerver), a rendszer hibát dob.

Queue hibák:

* Ha a queue rendszer nincs megfelelően konfigurálva, a /test-queue útvonal nem fog működni.

Sablon hibák:

* Ha a megadott e-mail sablon nem létezik, a /test-email-template útvonal hibát dob.

Használat

E-mail küldés tesztelése:

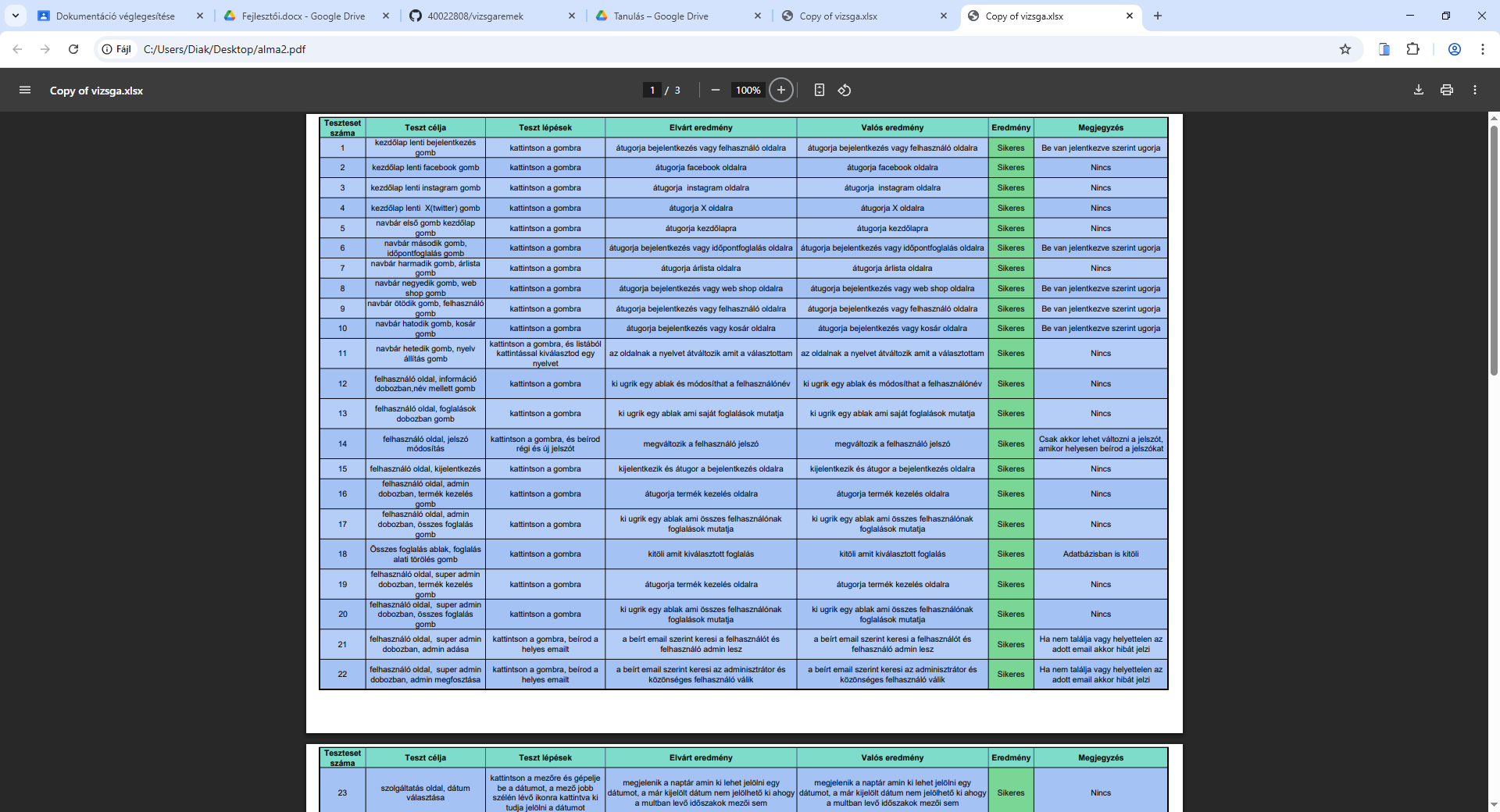
* A /test-email és /test-queue útvonalak segítségével ellenőrizhető az e-mail küldési funkciók működése.

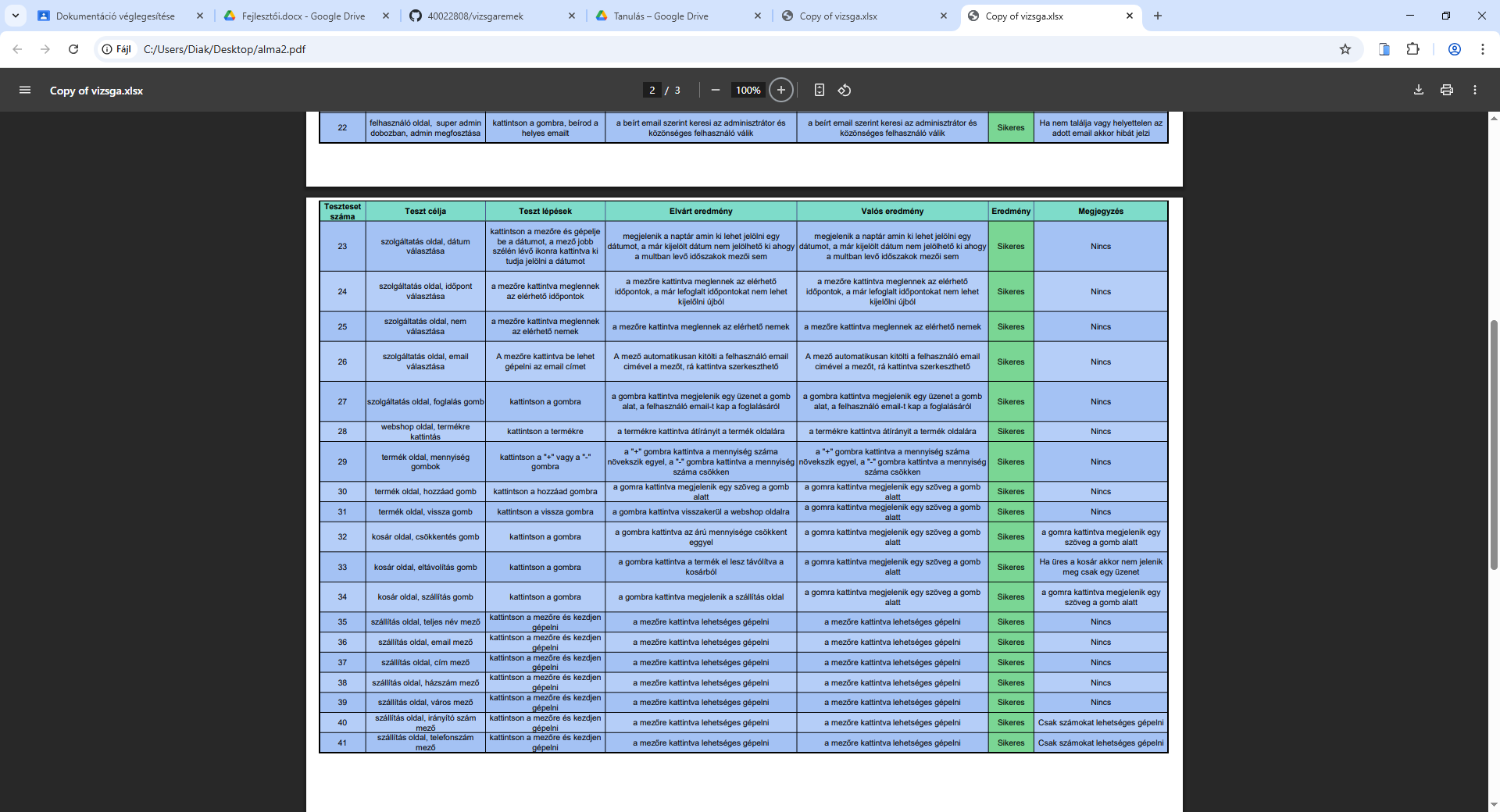
E-mail sablon fejlesztése:

* A /test-email-template útvonal lehetővé teszi az e-mail sablon megtekintését és tesztelését a böngészőben.

Alapértelmezett nézet elérése:

* A / útvonal az alkalmazás kezdőlapjának megjelenítésére szolgál.

Tesztelés



Összegzés

A projekt megvalósítása során szorosan együttműködő csapatként dolgoztunk, ahol minden tag saját feladatkörrel és felelősségi területtel járult hozzá a végeredményhez. A frontend fejlesztés Angular keretrendszerben, míg a backend fejlesztés Laravel-ben valósult meg, ami lehetővé tette a modern, dinamikus felhasználói élmény és a megbízható szerveroldali logika kialakítását.

A projekt során számos kihívással szembesültünk, különösen a két különböző technológia összehangolása jelentett kezdetben nehézséget. A frontend és backend közötti kommunikáció beállítása, az API-k megfelelő strukturálása, valamint az adatkezelés és jogosultságkezelés pontos kidolgozása mind-mind időigényes és tanulságos feladatnak bizonyult.

Ezen kívül a csapatmunka szervezése is fontos szempont volt: a feladatok felosztása, az időbeosztás betartása, és a közös hibakeresés folyamata mind hozzájárultak ahhoz, hogy fejlesszük a szakmai és együttműködési készségeinket. A rendszeres megbeszélések, egymás támogatása és az egységes cél kitűzése segített abban, hogy a kezdeti problémákat átlendítsük, és folyamatosan haladni tudjunk a fejlesztésben.

Mindezen nehézségek ellenére sikerült egy működő, jól strukturált és esztétikailag is letisztult weboldalt létrehozni, amely valós igényt elégít ki, és amelyre mindannyian büszkék lehetünk. A projekt nemcsak szakmailag jelentett kihívást, hanem lehetőséget adott arra is, hogy megtapasztaljuk az éles munka során felmerülő problémák megoldását, és fejlesszük közös problémamegoldó képességeinket.