Projektmunka

[Dokumentum alcíme]

Projektmunka

Készítette: Jancsurák Napsugár, Varga Antónia, Katona Alexandra

Tartalom

Fejlesztői környezet ……………………………………………..………………………. 4

Első implementációk – Leírás ………………………………………………………… 9

Második implementációk – Leírás …………………………………………………… 14

Harmadik implementációk – Leírás …………………………………………………. 19

Fejlesztői környezet

**Frontend:**

* **HTML5**: A **HTML5** (Hypertext Markup Language 5) az internetes tartalom egyik alapja, amely a weboldalak struktúráját és tartalmát határozza meg. Az előző verziókhoz képest a HTML5 számos új elemmel és funkcióval bővült, amelyek jelentősen javítják a weboldalak működését, hozzáférhetőségét és interaktivitását. Az egyik legfontosabb újítás, hogy a HTML5 már támogatja a **multimédiás elemek** (például audio és video) közvetlen lejátszását HTML kódból, anélkül, hogy külső pluginokra lenne szükség, így egyszerűbbé válik a tartalmak integrálása. A HTML5 által bevezetett **szemantikus HTML** elemek, mint például a <header>, <footer>, <article>, <section>, <nav>, nemcsak a weboldal szerkezetének tisztábbá tételét segítik, hanem hozzájárulnak a **SEO (keresőoptimalizálás)** javításához is. Ezen kívül a HTML5 támogatja az új **API-kat** is, mint a **Geolocation API** (geográfiai helymeghatározás), **Web Storage API** (helyi tárolás) és **WebSockets** (valós idejű kommunikáció), amelyek lehetővé teszik az interaktív weboldalak készítését, miközben a böngésző erőforrásaival takarékoskodnak.
* **CSS3**: A **CSS3** (Cascading Style Sheets, Level 3) a HTML5 mellett a weboldalak megjelenésének és dizájnjának alapvető szabályait határozza meg. A CSS3 az előző verziókhoz képest számos új funkcióval bővült, amelyek lehetővé teszik a fejlesztők számára, hogy dinamikus, interaktív és reszponzív felületeket készítsenek. A **reszponzív dizájn** azt jelenti, hogy az alkalmazás automatikusan alkalmazkodik a különböző képernyőméretekhez és eszközökhöz, így a felhasználók élménye minden eszközön (asztali gépen, mobilon, tableten) optimalizált marad. A **flexbox** és **CSS grid layout** technológiák a CSS3-ban lehetővé teszik az összetett elrendezések könnyed kezelését, miközben javítják a weboldalak teljesítményét is. A CSS3 segítségével a fejlesztők finomhangolhatják a **vizuális elemeket**, mint például a gombokat, navigációs sávokat, képeket, háttérszíneket és animációkat. Az új CSS funkciók, mint a **@keyframes** és **transitions**, lehetővé teszik a weboldal animációinak, áttűnéseinek és egyéb dinamikus hatásainak implementálását, amelyek az oldalon belépő és eltűnő elemek mozgásának és vizuális változásainak kezelését is biztosítják.
* **HTML5 és CSS3 együttműködése**: A **HTML5** és **CSS3** együtt biztosítják a weboldalak alapvető felépítését és megjelenését. A HTML5 biztosítja az oldal tartalmát és szerkezetét, míg a CSS3 a megjelenést, a vizuális stílusokat és az elrendezéseket szabályozza. A CSS3 lehetőséget biztosít az oldalak **reszponzív** megjelenítésére, így az alkalmazások automatikusan alkalmazkodnak a különböző kijelzők és eszközök méreteihez. A HTML5 új elemzései és struktúrái együttműködnek a CSS3 által kínált vizuális funkciókkal, lehetővé téve komplex felhasználói felületek és interaktív weboldalak gyors létrehozását. Emellett a CSS3-es animációk és a **flexbox** vagy **grid** alapú elrendezések segítenek a fejlesztőknek a dinamikus és vizuálisan gazdag felületek megvalósításában.
* **Bootstrap**: A **Bootstrap** egy nyílt forráskódú front-end keretrendszer, amely segít a webalkalmazások és weboldalak gyors és egyszerű felépítésében. A Bootstrap legnagyobb előnye, hogy előre definiált dizájn elemeket és **grid rendszert** biztosít a felhasználók számára, amellyel gyorsan létrehozhatók **reszponzív**, **szép** és **karbantartható** weboldalak. A **grid rendszer** segít az oldal elrendezésében, így a fejlesztők könnyen létrehozhatnak rugalmas, több oszlopos elrendezéseket anélkül, hogy bonyolult CSS kódot kellene írniuk. A Bootstrap tartalmaz egy sor előre megtervezett **UI komponenset**, mint például gombok, navigációs menük, űrlapok, kártyák és értesítési rendszerek. A keretrendszer biztosítja a reszponzivitást és az esztétikai követelményeket anélkül, hogy sok időt kellene tölteni a dizájn részletezésével.
* **WPF (Windows Presentation Foundation)**: A **WPF** egy **Microsoft** által kifejlesztett technológia, amely Windows alapú asztali alkalmazások fejlesztésére szolgál. A WPF az asztali alkalmazások felhasználói felületét kezeli, és számos fejlett grafikai elemet, animációt és interaktivitást kínál. A WPF lehetővé teszi a fejlesztők számára, hogy **dekoratív**, **dinamikus** és **gazdag** felhasználói élményt biztosítsanak a felhasználók számára, miközben teljes mértékben kihasználja a modern grafikus kártyák és a Windows operációs rendszer képességeit. A WPF **XAML** (Extensible Application Markup Language) szintaxist használ a felhasználói felületek deklaratív leírására, amely segíti a logika és a megjelenítés elválasztását, ezáltal tiszta és karbantartható kódot biztosít. A WPF támogatja az **MVVM** (Model-View-ViewModel) architektúrát, amely a kód tisztán tartására és a tesztelhetőség növelésére szolgál. A WPF erőteljes adatbinding mechanizmusa segíti a felhasználói felületek dinamikus frissítését, amikor az alkalmazás adatmodellje változik.

**Backend:**

* **C#**: A **C#** egy objektum-orientált, **erőteljes programozási nyelv**, amelyet a **Microsoft** fejlesztett ki, és a **.NET** platformon való fejlesztéshez használják. A C# modern programozási nyelvként számos fejlett funkcióval rendelkezik, amelyek segítenek a fejlesztőknek tiszta, karbantartható és hibamentes kódot írni. A C# erős típusellenőrzést, **garbage collection**-t (automatikus memória kezelés), valamint **LINQ** (Language Integrated Query) támogatást biztosít, amely megkönnyíti az adatkezelést és a komplex lekérdezések írását. A C# szoros integrációban áll a **.NET** platformmal, így a backend logika, adatkezelés és API fejlesztés ideális eszköze. A C# nyújtja az eszközöket, amelyek lehetővé teszik a fejlesztők számára a komplex üzleti logikák megvalósítását, miközben biztosítják a kód könnyű karbantarthatóságát.
* **.NET (vagy .NET Core/5/6/7)**: A **.NET** egy keretrendszer, amely lehetővé teszi a fejlesztők számára, hogy különböző típusú alkalmazásokat hozzanak létre, mint például **webalkalmazások**, **asztali alkalmazások**, **mobil alkalmazások**, és **szolgáltatások**. A **.NET Core** egy nyílt forráskódú és cross-platform verziója, amely támogatja a Windows, Linux és macOS rendszereket, így rugalmasan alkalmazkodik a fejlesztési igényekhez. A .NET platform gyors és skálázható alkalmazások fejlesztésére lett optimalizálva, amely különösen előnyös nagy terhelésű és adatintenzív alkalmazások esetén. Az újabb verziók, mint a **.NET 5, 6 és 7**, számos új funkciót és fejlesztési lehetőséget kínálnak, így a fejlesztők még gyorsabban és hatékonyabban tudják elkészíteni a kívánt alkalmazásokat.
* **ASP.NET Core**: Az **ASP.NET Core** a **.NET Core** keretrendszeren alapuló webfejlesztési keretrendszer, amely lehetővé teszi a modern, gyors és skálázható webalkalmazások, API-k és webszolgáltatások fejlesztését. Az ASP.NET Core alapvetően cross-platform, így az alkalmazások futtathatók Windows, Linux és macOS rendszereken is. Az ASP.NET Core az **MVC (Model-View-Controller)** architektúrát követi, amely biztosítja, hogy a kód jól szervezett és könnyen karbantartható legyen. Az **Entity Framework Core** az ASP.NET Core-ral együtt biztosítja az ORM (Object-Relational Mapping) lehetőséget, így az adatkezelés sokkal egyszerűbbé válik, és a fejlesztők közvetlenül a C# nyelvet használhatják az adatbázisok kezelésére.
* **SQL Server / MySQL / PostgreSQL / SQLite**: Az **adatbázisok** elengedhetetlenek a backend fejlesztés során, mivel az alkalmazásoknak strukturált módon kell tárolniuk és lekérdezniük az adatokat. Az adatbázis-kezelők közül a **SQL Server**, **MySQL**, **PostgreSQL** és **SQLite** mindegyike különböző funkciókat és előnyöket kínál. A **SQL Server** egy robusztus, vállalati szintű adatbázis-kezelő rendszer, amelyet a Microsoft fejlesztett ki, és kifejezetten nagy terhelésű alkalmazásokhoz alkalmas. A **MySQL** egy nyílt forráskódú, elterjedt relációs adatbázis-kezelő rendszer, amely az egyszerűbb alkalmazások számára kiváló választás. A **PostgreSQL** egy másik nyílt forráskódú adatbázis-kezelő, amely támogatja a bonyolultabb adatstruktúrákat és a transzakciók kezelését. Az **SQLite** egy könnyű, beágyazott adatbázis, amely ideális kisebb alkalmazásokhoz vagy helyi adatkezeléshez, mivel nincs szükség egy különálló szerverre a működéséhez. A választott adatbázis-kezelő rendszer az alkalmazás követelményeitől és méretétől függően változhat, de mindegyik biztosítja az adatok hatékony tárolását és gyors lekérdezését.
* **Entity Framework (EF)**: Az **Entity Framework** (EF) a **.NET** környezethez tartozó objektum-relációs leképezési (ORM) keretrendszer, amely segíti a fejlesztőket az adatbázisok és a programozási nyelvek közötti kapcsolat kezelésében. Az EF lehetővé teszi, hogy a fejlesztők C# osztályokat és objektumokat használjanak az adatbázis-táblák reprezentálására, így elkerülhetik a közvetlen SQL lekérdezések írását. A **Code First** megközelítéssel az adatbázist a fejlesztői kód alapján generálhatjuk, míg a **Database First** megközelítésnél már létező adatbázisból hozhatjuk létre a kódot. Az EF egyszerűsíti az adatbázis műveletek kezelését, mint az adatok lekérdezése, frissítése, törlése, miközben támogatja a komplex lekérdezéseket és tranzakciókat is. Az **EF Core** az EF könnyebb, gyorsabb és platformfüggetlen verziója, amely kifejezetten a **.NET Core** környezethez készült.
* **REST API**: A **REST (Representational State Transfer)** egy architektúra-stílus, amely az interneten keresztül történő adatkommunikációhoz használatos. A **RESTful API** egy olyan alkalmazásprogramozási felület (API), amely HTTP protokollt használ az adatok átvitelére a szerver és a kliens között. A REST API-k **stateless** működnek, azaz minden kérés tartalmazza a szükséges információkat, és nincs szükség a szerver számára a kérés előzményeinek tárolására. A REST API lehetővé teszi, hogy az alkalmazások különböző platformok és rendszerek között kommunikáljanak, miközben biztosítja a nagy skálázhatóságot, gyors adatátvitelt és könnyű integrálhatóságot. A RESTful API-k egyszerűsíthetik az alkalmazások közötti adatcserét, mivel támogatják a JSON formátumot, amely könnyen olvasható és széles körben használt.

Első implementációk – Leírás

**Áttekintés:**

Az első implementáció célja az ingatlanbérlési platform alapjainak lefektetése, egy olyan webes alkalmazás kialakítása, amely egyszerű és intuitív módon szolgálja ki a felhasználók két fő csoportját: a tulajdonosokat és a bérlőket.

* **Célfelhasználók:** Magánszemélyek vagy kisebb ingatlanügynökségek, akik ingatlanokat szeretnének bérbe adni, illetve azok, akik kiadó lakásokat keresnek.
* **Hasonló rendszerek:** Airbnb, Ingatlan.com, Rentingo. Ezekhez képest az első implementáció egyszerűbb, a felhasználók alapvető igényeit célozza meg.
* **Innováció és alkalmazhatóság:** A projekt könnyen bővíthető. A jelenlegi alapok lehetőséget biztosítanak további fejlesztésekre, például fizetési rendszer, értékelési rendszer vagy mobilalkalmazás integrálására.

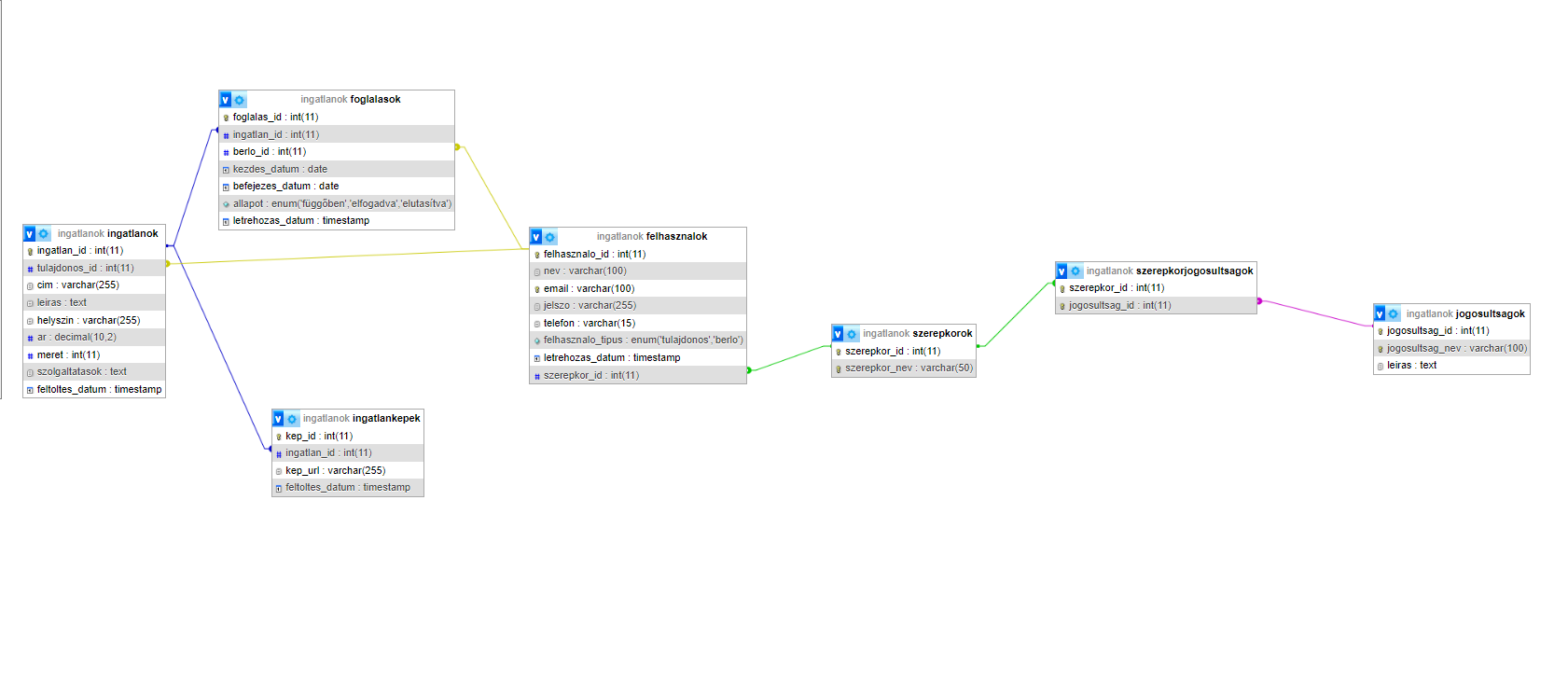
**Az implementáció fő funkcionalitása**

Főbb komponensek:

1. Felhasználók kezelése:
   * Regisztráció (tulajdonosként vagy bérlőként).
   * Bejelentkezés és profilkezelés.
   * Szerepkörök és jogosultságok dinamikus kezelése.
2. Ingatlanok kezelése:
   * Tulajdonosok feltölthetik ingatlanjaikat képekkel és leírással.
   * Az ingatlanok listázása, szűrése (pl. ár, helyszín, méret alapján).
   * Egyedi ingatlan részleteinek megtekintése (ár, méret, képek).
3. Foglalások és kommunikáció:
   * Bérlők foglalási kérelmeket küldhetnek a tulajdonosoknak.
   * Tulajdonosok foglalási kérelmeket kezelhetnek (elfogadás/elutasítás).

**3. Adatbázis és kapcsolatok**

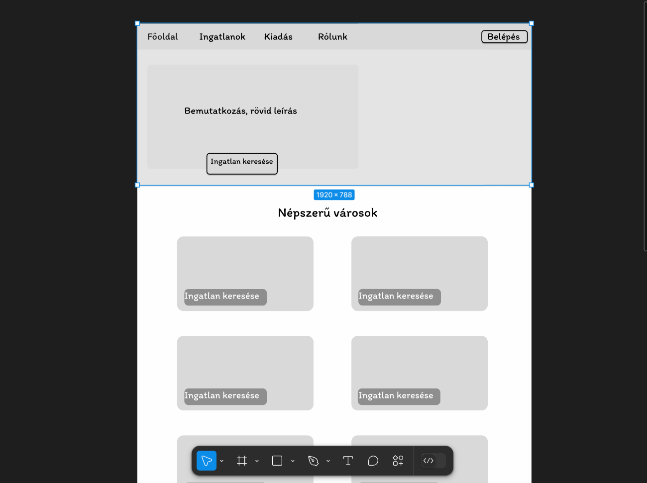
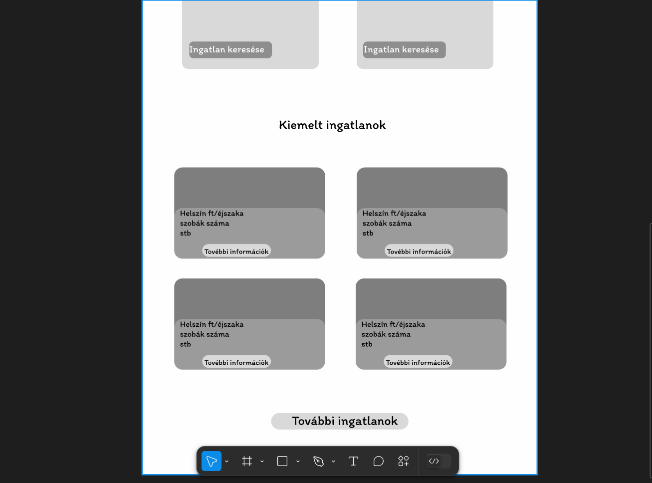
Adatbázis terv: Az adatbázis logikai felépítése biztosítja az adatkezelés biztonságát és rugalmasságát.

* Kapcsolatok:
  + A Felhasznalok tábla központi, innen indulnak a kapcsolatok az Ingatlanok, Foglalasok táblák felé.
  + Az Ingatlanok a tulajdonosokhoz kötődnek, és egyedi foglalásokkal, képekkel kapcsolódnak.
  + A Szerepkorok és Jogosultsagok táblák biztosítják a felhasználók jogosultsági szintjeit.

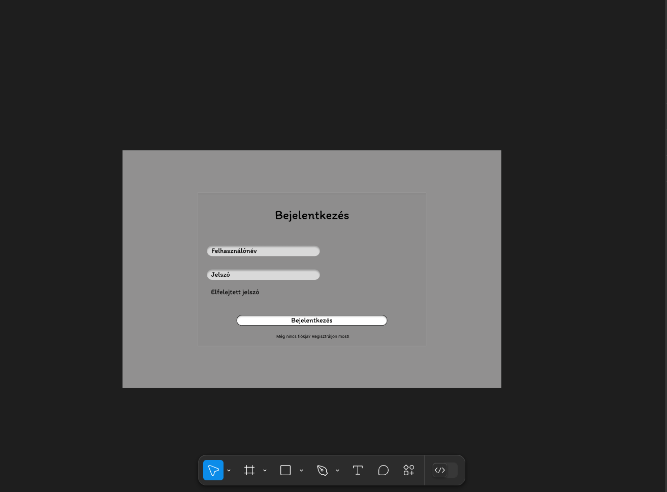
**Drótváz és vizuális terv**

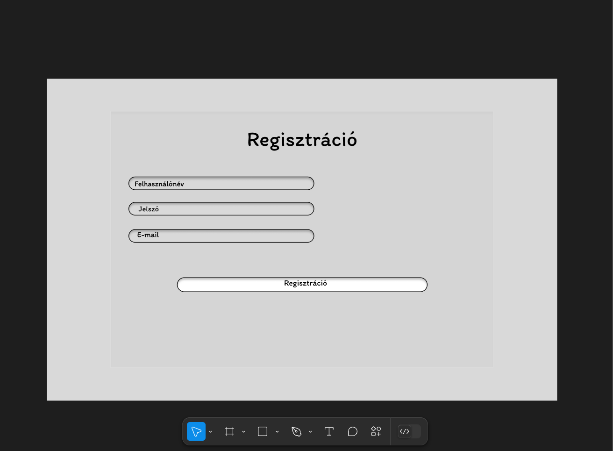
Drótváz**:** A weboldal alapvető elrendezése így épül fel:

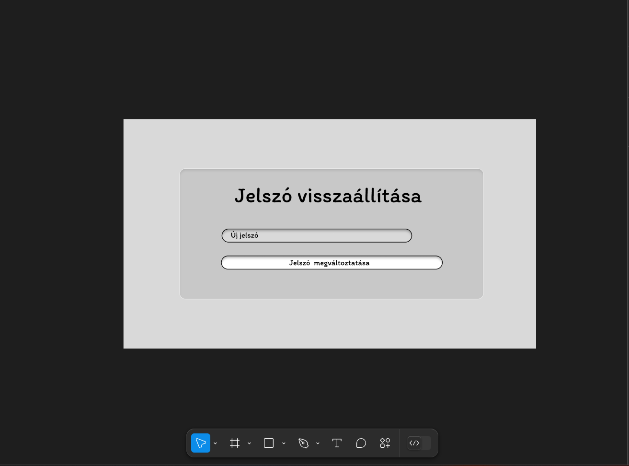
1. Főoldal**:**
   * Keresőmező: Ár, helyszín, méret szűrési lehetőségekkel.
   * Legújabb ingatlanok listája.



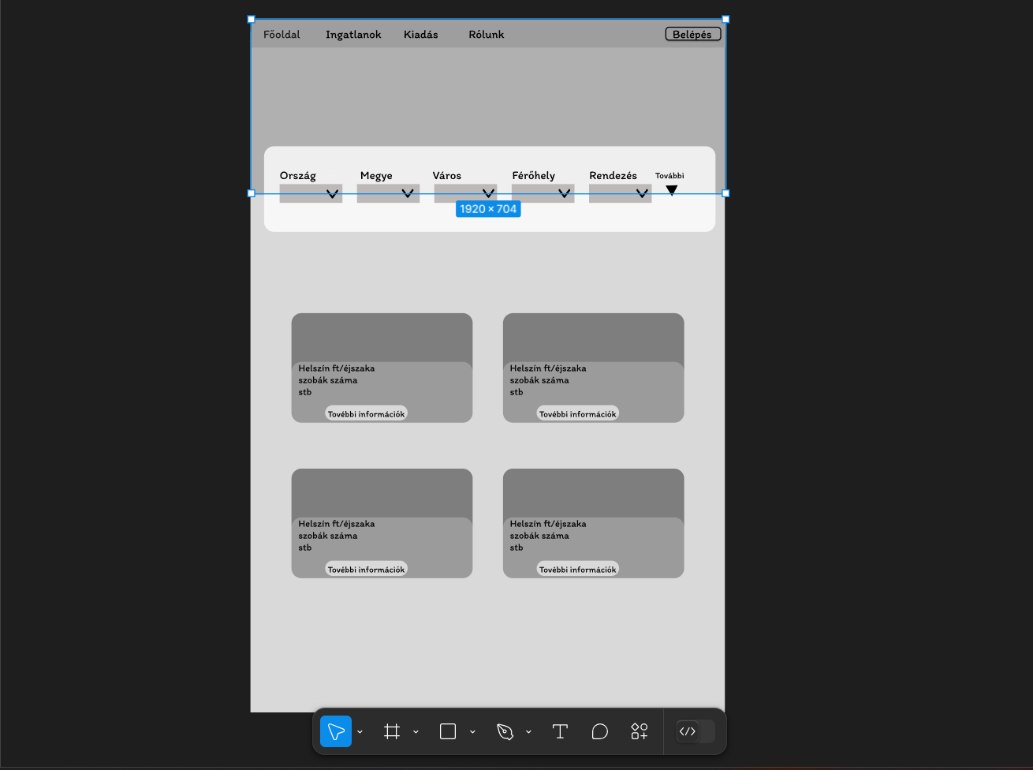
1. Felhasználói profiloldal:
   * Regisztrációs/Bejelentkezési űrlap.
   * Felhasználói adatok módosítása.



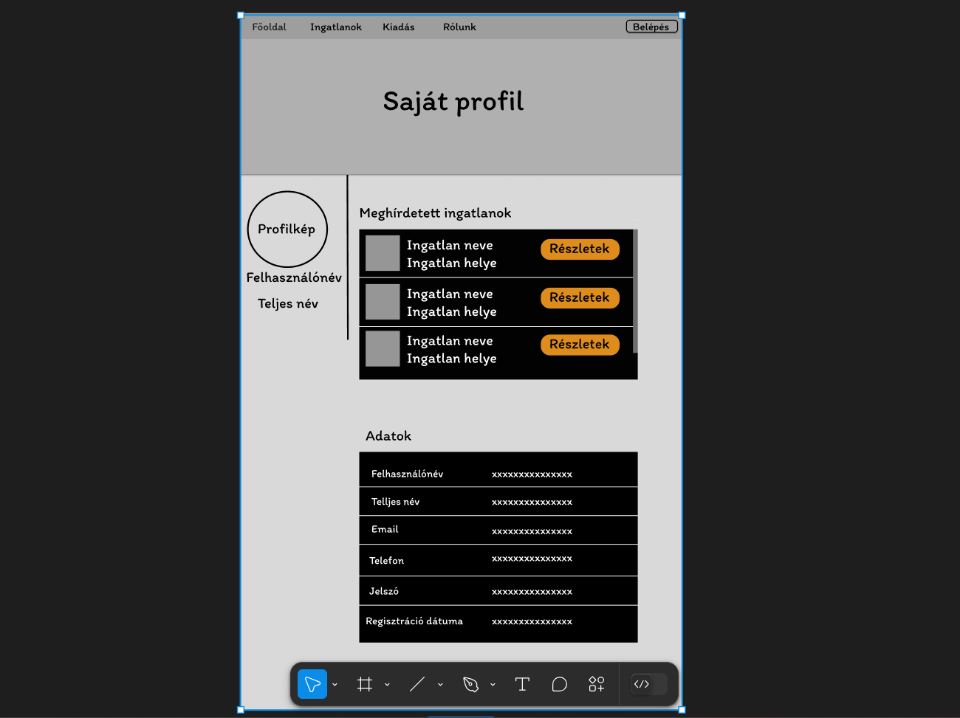




1. Ingatlan részletek oldala:
   * Teljes leírás, képgaléria.
   * Foglalás gomb lehetőség.



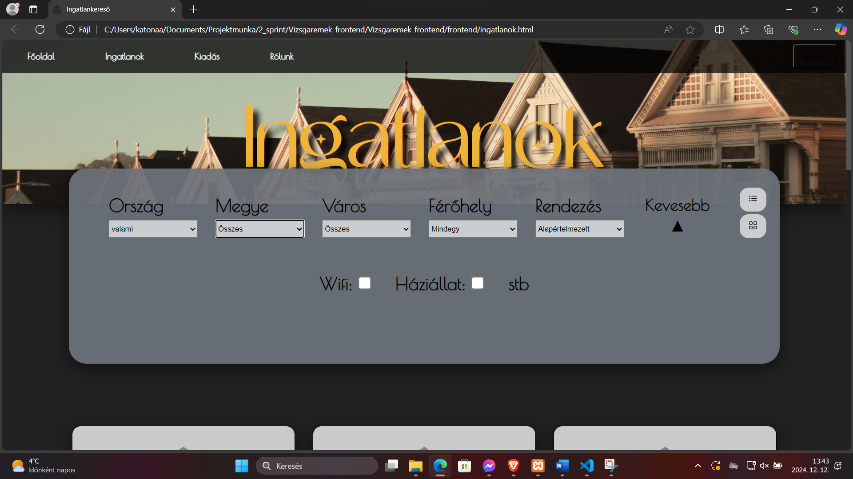
1. Adminisztrációs felület (Tulajdonosoknak):
   * Ingatlanok listázása, új ingatlanok feltöltése, meglévők szerkesztése.



**React alapú weboldal**

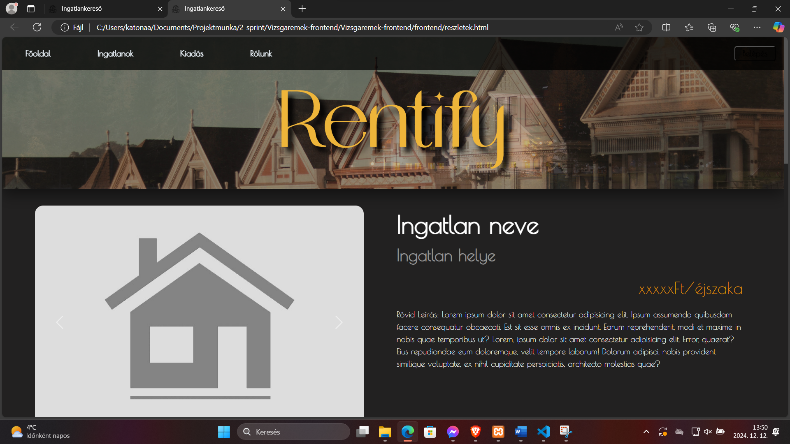
1. Keresőmodul:
   * Felhasználók bizonyos feltételek szerint ingatlanokra a legördülő menükkel.

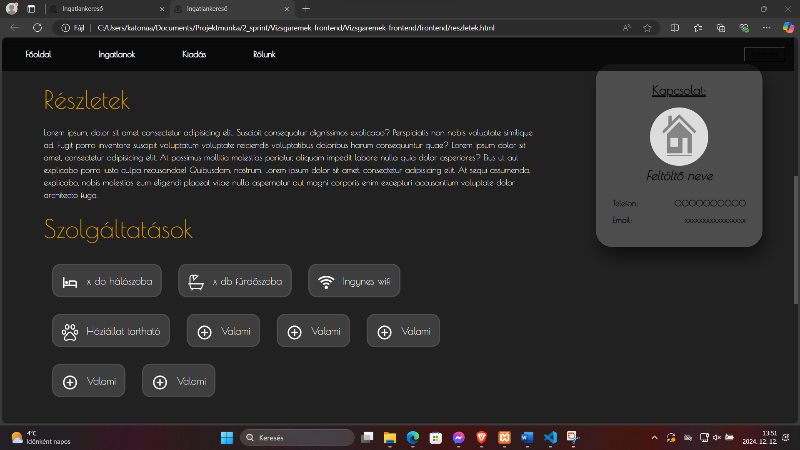




**Részletező oldal:**

* Egy adott ingatlan részletes adatai jelennek meg, például képek, ár, leírás.

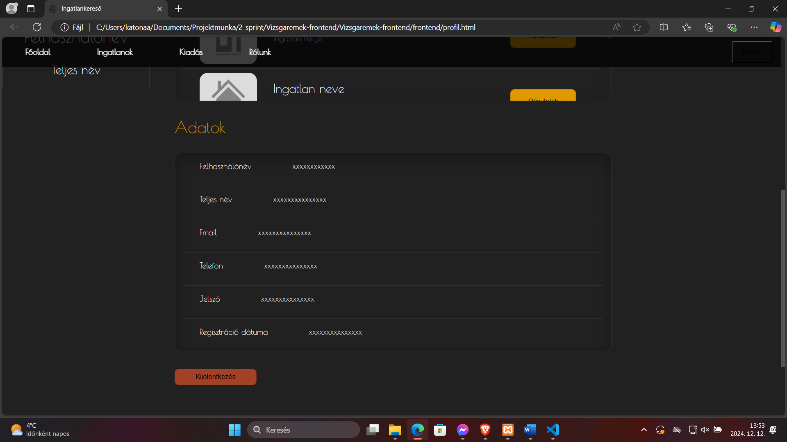




**Admin felület:**

* Tulajdonosok által kezelhető panel: új ingatlan hozzáadása, meglévő adatok szerkesztése.





Második implementációk - Leírás

A második iteráció során az ingatlanbérlési platform fejlesztése elsősorban a felhasználói élmény javítására és a működés központi funkcióinak további kidolgozására összpontosított. Ennek keretében kiemelt figyelmet kapott a **bejelentkezési, regisztrációs funkciók**, a **profiloldal** kialakítása, valamint az adatbázis-alapú **adatok kilistázása** és kezelése.

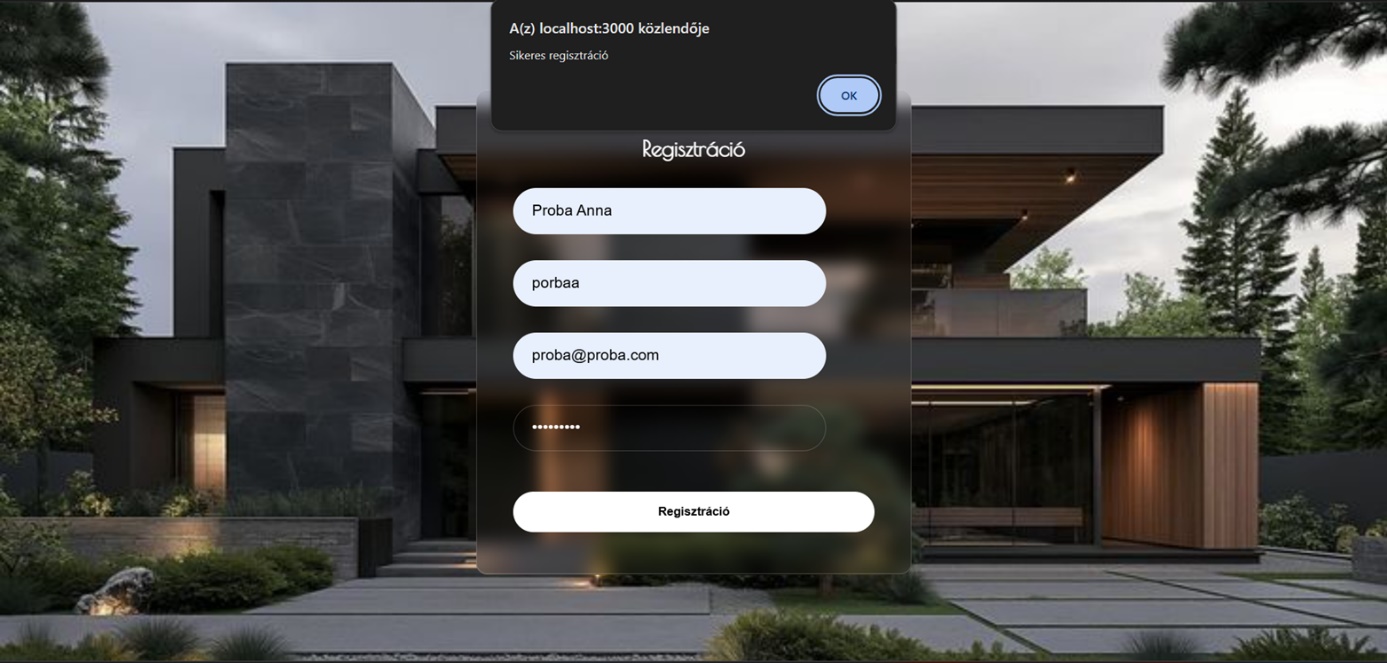
**1. Bejelentkezés és regisztráció**

**1.1 Regisztráció funkció**

A regisztráció során a felhasználók kétféle szerepkör közül választhatnak:

* **Tulajdonosok**, akik ingatlanjaikat szeretnék kiadni.
* **Bérlők**, akik bérelhető ingatlanokat keresnek.

A regisztráció folyamata a következőképpen épül fel:

1. **Űrlap kitöltése**
   * Felhasználónév, név, email-cím, jelszó (kötelezően kitöltendő mezők).
2. **Szerepkör választás**
   * A felhasználók megjelölik, hogy tulajdonosként vagy bérlőként kívánnak regisztrálni.
3. **Validáció és mentés**
   * Az űrlapon megadott adatok validálása (például email formátum ellenőrzése).
   * Az adatbázisban a Felhasználók tábla frissítése.

**1.2 Bejelentkezés funkció**

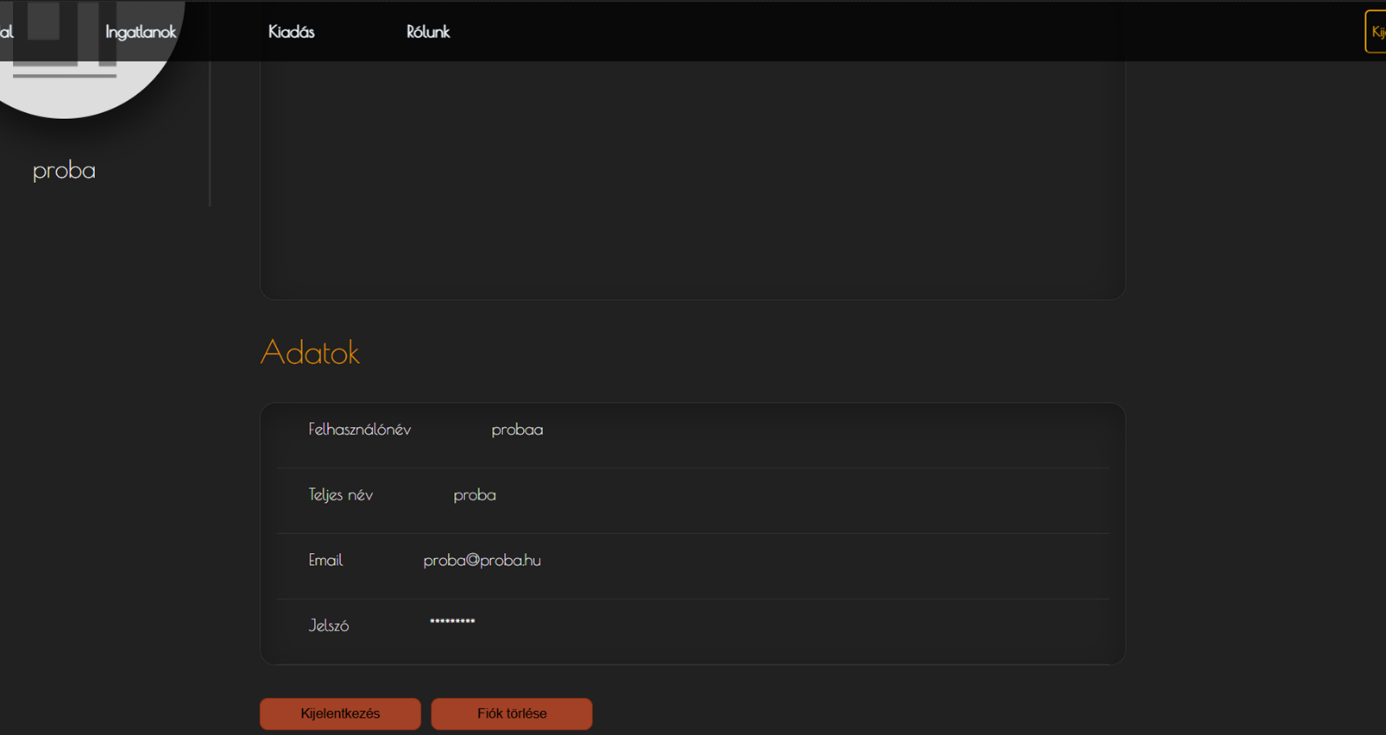
A bejelentkezés egy egyszerű felületen keresztül történik, ahol a felhasználók felhasználónevük és jelszavuk megadásával férhetnek hozzá fiókjukhoz. A következő fejlesztéseket implementáltuk:

* **Hitelesítés:** A megadott adatok ellenőrzése az adatbázisban tárolt értékek alapján.
* **Hibakezelés:** Hibás adatok esetén értesítés a felhasználónak („Érvénytelen email vagy jelszó” üzenet).
* **Token alapú munkamenet-kezelés:** Sikeres bejelentkezés után a felhasználó egy időkorlátos munkamenet-tokent kap, amely biztosítja az alkalmazás biztonságos használatát.

**2. Profiloldal**

A profiloldal funkciójának célja, hogy a felhasználók kezelhessék saját adataikat és áttekintsék a tevékenységüket.

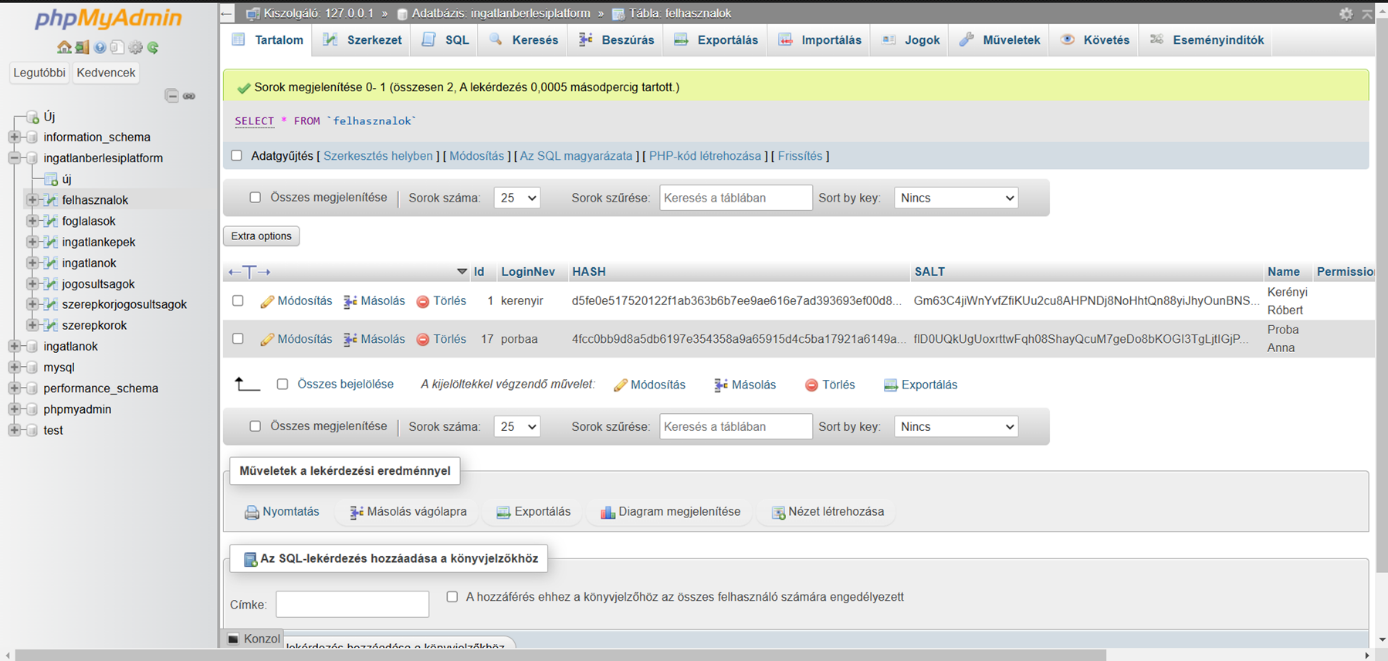
**2.1 Felhasználói adatok megjelenítése**

A profiloldalon a regisztráció során megadott adatok jelennek meg.

**2.2 Szerepkör-specifikus tartalom**

* **Tulajdonosok:** Listázhatják és kezelhetik feltöltött ingatlanaikat (például új ingatlan hozzáadása, meglévők szerkesztése).
* **Bérlők:** Áttekinthetik korábbi foglalásaikat, kommunikációs előzményeiket.

**3. Adatok kilistázása adatbázisból**

A második iteráció során kiemelt figyelmet kapott az adatbázisból történő adatlekérés optimalizálása és megjelenítése.

**3.1 Ingatlanok listázása**

Az ingatlanok megjelenítésére szolgáló modul többféle szűrési lehetőséget biztosít:

* **Alapvető szűrők:** Helyszín, méret, rendezés.
* **Speciális szűrők:** Felszereltség (Wi-Fi), vihető-e háziállat az ingatlanba.

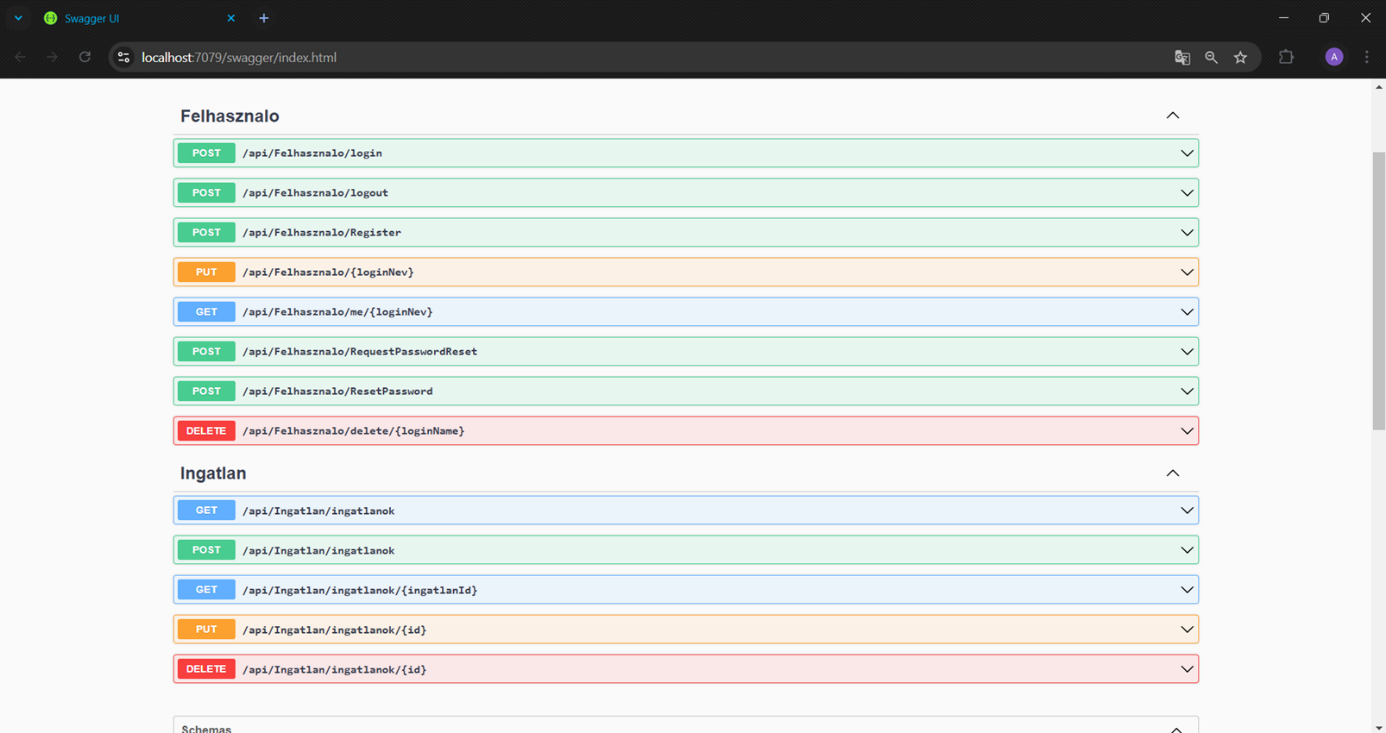
Az adatbázisból lekért adatok az alábbi információkat tartalmazzák:

* Kép, ár, cím, rövid leírás.
* Az ingatlanra kattintva a részletes adatok is megtekinthetők (például több kép, hosszabb leírás, tulajdonos elérhetősége).

**4. Működési logika és technikai megoldások**

**4.1 Backend fejlesztések**

* **REST API bővítések:** Az adatlekérdezések REST API végpontokon keresztül történnek, amelyeket C# és ASP.NET Core technológiával implementáltunk.
* **Adatbázis műveletek:** Az Entity Framework Core segítségével biztosítottuk az ORM kapcsolatot, amely lehetővé tette a dinamikus lekérdezéseket és az adatok validációját.



**4.2 Frontend fejlesztések**

* **React komponensek:** Az egyes funkciókat, például a bejelentkezést, regisztrációt és adatlistázást különálló React-komponensekbe szerveztük, hogy a kód könnyen karbantartható legyen.
* **Reszponzív design:** A Bootstrap grid rendszere és egyedi CSS3 megoldások biztosítják az alkalmazás zökkenőmentes működését minden eszközön.

**5. Tesztelés és hibajavítások**

Az iteráció végén átfogó tesztelést végeztünk:

* **Unit tesztek:** A backend API-k funkcionalitásának ellenőrzésére.
* **Integrációs tesztek:** Az adatbázis és a frontend összekapcsolásának tesztelésére.
* **Felhasználói tesztek:** Az alkalmazás logikájának és felhasználói élményének értékelésére.

**6. Összegzés**

A második iteráció során a platform funkcionalitásának alapvető elemei elkészültek, különös tekintettel a felhasználók kezelési folyamataira és az adatbázis-integrációra. Ezek az alapok biztosítják a további bővítések és fejlesztések sikerességét.