VÁCI SZAKKÉPZÉSI CENTRUM  
BORONKAY GYÖRGY  
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM

VIZSGAREMEK  
Freya’s Garden – Növény csere-bere

Nacsa Levente  
Szabolics András  
Tábor Tünde  
2024-2025.

VÁCI SZAKKÉPZÉSI CENTRUM  
BORONKAY GYÖRGY  
MŰSZAKI TECHNIKUM ÉS GIMNÁZIUM  
VIZSGAREMEK  
Freya’s Garden – Növény csere-bere

Konzulens: Kovács László Készítette: Nacsa Levente  
Gyombolai Ferenc Péter Szabolics András  
 Tábor Tünde

# Hallgatói nyilatkozat

Alulírottak, ezúton kijelentjük, hogy a szakdolgozat saját, önálló munkánk, és korábban még sehol nem került publikálásra. Szakdolgozatunk a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium Szoftverfejlesztő és tesztelő technikus képzésén készítettük. Tudomásul vesszük, hogy szakdolgozatunkat a Váci Szakképzési Centrum Boronkay György Műszaki Technikum és Gimnázium tárolja.

Nacsa Levente Szabolics András Tábor Tünde

# Konzultációs lap

Vizsgázók neve: Nacsa Levente

Szabolics András

Tábor Tünde

Vizsgaremek címe: Freya’s Garden – Növény csere-bere

A projekt nyújtotta szolgáltatások:

* Regisztráció, bejelentkezés
* Hirdetések megosztása
* Hirdetések megtekintése, keresése
* Kapcsolódó cikkek megtekintése, szűrése, keresése
* Adminisztráció (Jogosultsági szintek)
* Adatbázis-kezelés

|  |  |  |
| --- | --- | --- |
| Sorszám | A konzultáció időpontja | A konzulens aláírása |
| 1. | 2024.10.17. |  |
| 2. | 2024.11.21. |  |
| 3. | 2024.12.13. |  |
| 4. | 2025.01.16. |  |
| 5. | 2025.02.20. |  |
| 6. | 2025.03.08. |  |

A vizsgaremek beadható: A vizsgaremeket átvettem:  
Vác, 2025. Vác, 2025.

Konzulens A szakképzést folytató  
intézmény felelős

Tartalomjegyzék

[1. Hallgatói nyilatkozat 4](#_Toc197983842)

[2. Konzultációs lap 5](#_Toc197983843)

[3. Témaválasztás 4](#_Toc197983844)

[4. Tervek, specifikáció 5](#_Toc197983845)

[4.1 Célkitűzés, feladat 5](#_Toc197983846)

[4.2 Koncepció 5](#_Toc197983847)

[4.2.1 Weboldal 5](#_Toc197983848)

[4.2.2 Marketplace 5](#_Toc197983849)

[4.2.3 Dev app 5](#_Toc197983850)

[4.3 Adatbázis 6](#_Toc197983851)

[4.3.1 Normálformák 6](#_Toc197983852)

[4.4 Design 8](#_Toc197983853)

[5. Munkamegosztás 9](#_Toc197983854)

[5.1 Tábor Tünde 9](#_Toc197983855)

[5.2 Nacsa Levente 9](#_Toc197983856)

[5.3 Szabolics András 9](#_Toc197983857)

[5.4 GitHub Projekt 9](#_Toc197983858)

[6. Fejlesztői dokumentáció 11](#_Toc197983859)

[6.1 Fejlesztői környezet 11](#_Toc197983860)

[6.1.1 Visual Studio Code 11](#_Toc197983861)

[6.1.2 Visual Studio 2022 11](#_Toc197983862)

[6.1.3 EchoAPI 11](#_Toc197983863)

[6.1.4 GIMP 11](#_Toc197983864)

[6.1.5 Inkscape 11](#_Toc197983865)

[6.1.6 XAMPP 12](#_Toc197983866)

[6.2 Használt programozási nyelvek, technológiák, könyvtárak 12](#_Toc197983867)

[6.2.1 HTML 12](#_Toc197983868)

[6.2.2 CSS 12](#_Toc197983869)

[6.2.3 JavaScript 12](#_Toc197983870)

[6.2.4 C# 12](#_Toc197983871)

[6.2.5 XAML 13](#_Toc197983872)

[6.2.6 JSON 13](#_Toc197983873)

[6.2.7 MySQL 13](#_Toc197983874)

[6.2.8 phpMyAdmin 13](#_Toc197983875)

[6.2.9 HTTP REST API 14](#_Toc197983876)

[6.2.10 Laravel 14](#_Toc197983877)

[6.2.11 PHP 14](#_Toc197983878)

[6.2.12 MAUI 15](#_Toc197983879)

[6.2.13 Composer 15](#_Toc197983880)

[6.3 Adatbázis 15](#_Toc197983881)

[6.3.1 Adatbázis alapadatai 15](#_Toc197983882)

[6.3.2 Táblák 16](#_Toc197983883)

[6.4 A rendszer felépítése 25](#_Toc197983884)

[6.4.1 Marketplace 25](#_Toc197983885)

[6.4.2 Weboldal 25](#_Toc197983886)

[6.4.3 Dev app 25](#_Toc197983887)

[6.4.4 REST API 25](#_Toc197983888)

[6.5 Használati esetmodell, szerepkörök 27](#_Toc197983889)

[6.5.1 Szerepkörök, jogosultságaik 27](#_Toc197983890)

[6.5.2 Szerepkörök halmazábrája 28](#_Toc197983891)

[6.6 A rendszer működése 28](#_Toc197983892)

[6.6.1 Adatbázis kapcsolat, Laravel 28](#_Toc197983893)

[6.6.2 Felhasználók átmeneti azonosítása, végpontok biztonsága 29](#_Toc197983894)

[6.6.3 elérhető kérések listája 30](#_Toc197983895)

[6.6.4 API kérések működése, példák 31](#_Toc197983896)

[6.6.5 Az MVVM és MVC architektúra 35](#_Toc197983897)

[6.6.6 Databinding 37](#_Toc197983898)

[6.7 Tesztelés 38](#_Toc197983899)

[6.7.1 API tesztelés 38](#_Toc197983900)

[6.8 Továbbfejlesztési lehetőségek 38](#_Toc197983901)

[7. Felhasználói dokumentáció 39](#_Toc197983902)

[7.1 A rendszer használata 39](#_Toc197983903)

[7.1.1 MARKETPLACE 39](#_Toc197983904)

[7.1.2 WEB 44](#_Toc197983905)

[7.1.3 DEV 46](#_Toc197983906)

[7.2 A rendszer indítása 46](#_Toc197983907)

[7.2.1 API 47](#_Toc197983908)

[7.2.2 WEB 47](#_Toc197983909)

[8. Források 47](#_Toc197983910)

[9. Mellékletek 48](#_Toc197983911)

[9.1 Forráskód (github) 48](#_Toc197983912)

# Témaválasztás

A Freya’s Garden célja egy önfenntartó, egészséges közösség alapítása és fenntartása az otthoni kertek működéséért a városokban.

Manapság nagyon sok háztartásban megoldható lenne egy konyhakert vagy akár egy kis ültetvény fenntartása. A Freya’s Garden lehetővé teszi, elősegíti ezeknek működését, megjelenését, fenntartását.

A weboldalunkon megtalálhatók inspiráló, segítőkész, avagy csak érdekes cikkek. A marketplace app letöltésével és használatával pedig könnyen adományozhatunk, cserélhetünk, adhatunk, vehetünk növényeket és kiegészítőket.

Az ötlet megfogalmazása során nagy szerepet játszott az, hogy egy valós problémát oldjunk meg. A környezetvédelem, és az önfenntarthatóság mind olyan témák, melyek valós problémákkal foglalkoznak, és valamennyire mindenki számára fontosok. Viszont mindezzel foglalkozni kimerítő reménytelen feladat lehet, főleg manapság. Ezért mi kisebb léptékben gondolkoztunk. Tudatos, jóindulatú közösségek megteremtésével egy nagyobb társadalmi változás első, kis lépéseinek adunk lehetőséget. Persze ez nem teljes megoldás, sokkal több faktor sokkal nagyobb léptékben változtatja környezetünket. Annyi viszont biztos, hogy a mindennapjainkban pozitív változást tud előidézni egy olyan kezdeményezés, mint a mi projektünk.

# Tervek, specifikáció

## Célkitűzés, feladat

A feladatunk volt megtervezni, leprogramozni, tesztelni, dokumentálni egy - két kliens applikációból egy adatbázisból és egy API-ból álló - rendszert.

## Koncepció

### Weboldal

Egy cikkek böngészésére és olvasására alkalmas platform, ami az adtabázisban tárolt cikkeket az API-on keresztül lekéri. A cikkek között lehet böngészni, amit filterek és keresés megkönnyíthet a felhasználó számára. A weboldalon még jelszóvisszaállításra is van lehetőség, ha a marketplace-en kérünk egy visszaállító emailt.

### Marketplace

Fiók létrehozása és bejelentkezés után az applikáció fő célja, hogy növényeket cserélhessenek felhasználóink akár Android-ról is. Ezt hirdetések megosztásával és böngészésével lehet megoldani.

Emellett az alkalmazásban kezelhetjük fiókunkat és megnézhetjük más felhasználók fiókját.

### Dev app

Admin vagy Stats jogú felhasználóink számára fenntartott Windows-ra készült applikáció, amiből képesek új cikkeket írni, kezelni eddigi cikkeiket, valamint admin jogosultsággal módosítani a legtöbb adatbázis mezőt.

## Adatbázis

Itt látható a kezdetleges adatbázis terv, a projekt elején felvázolva, a [végleges adatbázis a fejlesztői dokumentációban a 6.3.2-es](#_Táblák) pontban elérhető.

### Normálformák

0.normálforma  
table{plant\_id, plant\_name, plant\_latin\_name, article\_id, article\_title, article\_source(The place of html code), user\_id, user\_username, user\_email, user\_city, user\_birthdate, user\_password(encrypted), post\_id, post\_city, post\_title, post\_description, post\_media, post\_media, post\_sell(bool)}

1.normálforma  
**plant**{plant\_id, plant\_name, plant\_latin\_name}

**other**{ article\_id, article\_title, article\_source(The place of html code), user\_id, user\_username, user\_email, user\_city, user\_birthdate, user\_password(encrypted), post\_id, post\_city, post\_title, post\_description, post\_media, post\_media, post\_sell(bool)}

2.normálforma  
**plant**{plant\_id, plant\_name, plant\_latin\_name}

**article**{article\_id, article\_title, plant\_id article\_source(The place of html code)}

**user**{user\_id, plant\_id, user\_username, user\_email, user\_city, user\_birthdate, user\_password(encrypted)}

**post**{post\_id, post\_city, post\_title, post\_description, post\_media, post\_media, post\_sell(bool)}

3.normálforma  
**plant**{plant\_id, plant\_name, plant\_latin\_name}

**article**{article\_id, article\_title, plant\_id article\_source(The place of html code)}

**user**{user\_id, user\_username, user\_email, user\_city, user\_birthdate, user\_password(encrypted)}

**post**{post\_id, post\_city, post\_title, post\_description, post\_media, post\_sell(bool)}

**user\_plant**{user\_id, plant\_id}

3.normálforma, a neveket egyszerűsítetük

**plant**{id, name, latin\_name}

**article**{id, title, id source(The place of html code)}

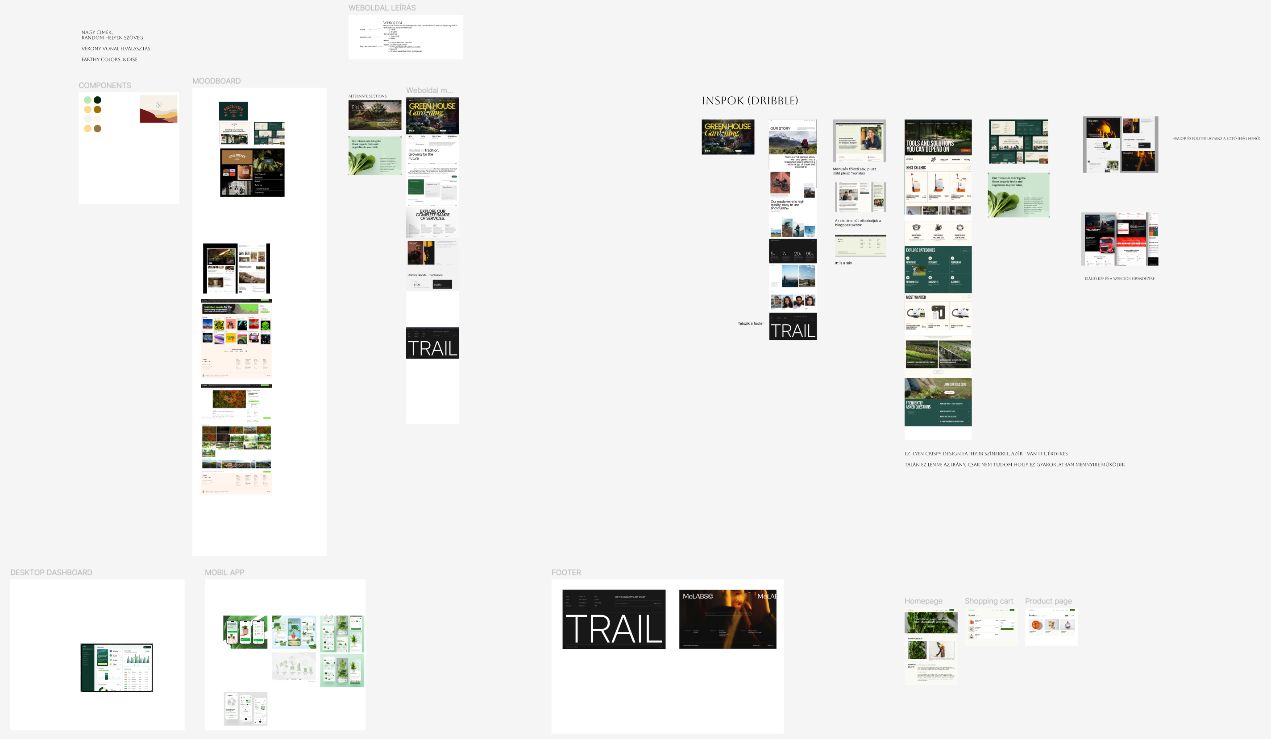
**user**{id, username, email, city, birthdate, password(encrypted)}

**post**{id, city, title, description, media, sell(bool)}

**user\_plant**{user\_id, plant\_id}

## Design

A design tervezéséhez, inspirációgyűjtéshez figma-t használtunk.



# Munkamegosztás

## Tábor Tünde

* Project szervezése
* Design alapelvei
* API keresések
* API automata seeder-ek
* API képek tárolása
* Marketplace/ dev app bejelentkezés
* Marketplace hirdetések megjelenítése, keresése
* Weboldal cikkek megjelenítésé, szűrése, keresése

## Nacsa Levente

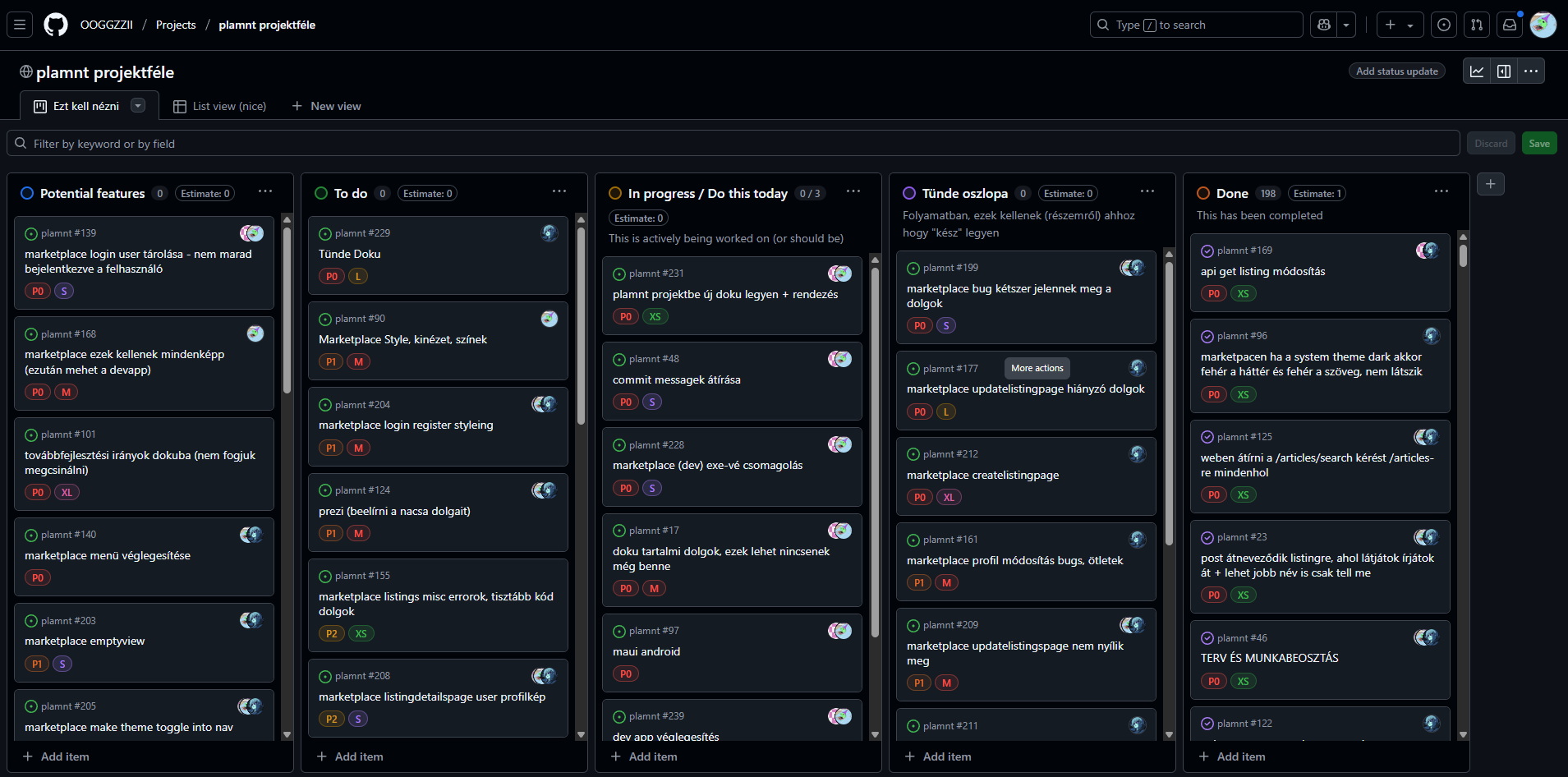
* Rest API architektúra
* API regisztráció, bejelentkezés
* API szerepkörök
* API adatbázis, modellek felállítása
* API tesztek
* Marketplace Android-os portolása
* Jelszóvisszaállítás, automata Email küldés
* Dokumentáció feladatfelosztás, tervezés

## Szabolics András

* API statikus seeder-ek
* API adatgyűjtés
* Marketplace/dev app login, felhasználó tárolása
* Prezentáció fő készítője
* Dokumentáció kerete
* Rajzok – profilképek, helyfoglaló
* Weboldal formázása
* Marketplace formázása
* API tesztek

## GitHub Projekt

A feladatok számontartásához, beosztásához egy közös GitHub projektet használtunk, amivel rendszerezve tudtuk kezelni a különféle részfeladatait projektünknek.

  
GitHub Projekt

# Fejlesztői dokumentáció

## Fejlesztői környezet

### Visual Studio Code

A Visual Studio Code, amelyet általában VS Code-ként is emlegetnek, egy forráskód-szerkesztő, amelyet a Microsoft fejlesztett ki Windows, Linux, macOS és webböngészők számára. A szolgáltatások közé tartozik a hibakeresés támogatása, a szintaktikai kiemelés, az intelligens kódkiegészítés, a kódrészletek, a kódrefaktorálás és a beágyazott verziókezelés a Git segítségével. A felhasználók módosíthatják a témát, a billentyűparancsokat, a beállításokat, telepíthetnek bővítményeket, amelyek funkcionalitást adnak, és beállíthatják a VS-kódot IDE-ként való használatra.

### Visual Studio 2022

A Visual Studio IDE egy kreatív indítópult, amellyel szerkesztheti, összeállíthatja a kódot, majd közzétehet egy alkalmazást. A legtöbb IDE által biztosított szabványos szerkesztőn és hibakeresőn túl a Visual Studio fordítókat, kódkiegészítő eszközöket, grafikus tervezőket és még sok más szolgáltatást is tartalmaz a szoftverfejlesztési folyamat javítása érdekében.

### EchoAPI

Az EchoAPI egy minden az egyben API fejlesztési együttműködési platform, amely olyan alapvető funkciókat kínál, mint az API hibakeresés, tervezés, automatizált tesztelés, stresszteszt és dokumentáció generálás.

### GIMP

A GNU Image Manipulation Program, közismert mozaikszóval GIMP, egy ingyenes és nyílt forráskódú raszteres grafikus szerkesztő, amely képmanipulációra (retusálásra) és képszerkesztésre, szabad formájú rajzolásra, különböző képfájlformátumok átkódolására és speciálisabb feladatokra használható. Beépülő modulokkal bővíthető, és szkriptelhető. Nem rajzolásra készült, bár egyes művészek és alkotók így használták.

### Inkscape

Az Inkscape egy ingyenes és nyílt forráskódú vektorgrafikus szerkesztő GNU/Linux, Windows és macOS számára. A funkciók gazdag készletét kínálja, és széles körben használják művészi és műszaki illusztrációkhoz, például rajzfilmekhez, clip art-okhoz, logókhoz, tipográfiához, diagramokhoz és folyamatábrákhoz. Vektorgrafikát használ, hogy éles nyomatokat és korlátlan felbontású rendereléseket tegyen lehetővé, és nincs meghatározott számú pixelhez kötve, mint például a rasztergrafikák.

### XAMPP

Az XAMPP az Apache Friends által kifejlesztett ingyenes és nyílt forráskódú, többplatformos webszerver-megoldási csomag, amely főleg az Apache Server-ből, MariaDB adatbázisból és PHP nyelvű HTTP-tolmácsokból áll. Mivel a legtöbb tényleges webszerver-telepítés ugyanazokat az összetevőket használja, mint a XAMPP, lehetővé teszi az átállást a helyi tesztkiszolgálóról egy élő szerverre.

## Használt programozási nyelvek, technológiák, könyvtárak

### HTML

A HTML (angolul: HyperText Markup Language) egy leíró nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és mára már internetes szabvánnyá vált a W3C (World Wide Web Consortium) támogatásával. Az aktuális változata az 5, mely az SGML általános jelölőnyelv egy konkrét alkalmazása.

### CSS

A Cascading Style Sheets (CSS) egy stíluslapnyelv, amelyet egy jelölőnyelven, például HTML-ben vagy XML-ben (beleértve az XML-dialektusokat, például az SVG-t, MathML-t vagy XHTML-t) írt dokumentumok megjelenítésének és stílusának meghatározására használnak. A CSS a World Wide Web egyik sarokköve, a HTML és a JavaScript mellett.

### JavaScript

JavaScript amelyet gyakran JS-nek rövidítenek, a HTML és a CSS mellett a World Wide Web programozási nyelve és alaptechnológiája. A webhelyek 99 százaléka JavaScriptet használ a kliensoldalon a weboldalviselkedéséhez. A webböngészőknek van egy dedikált JavaScript motorja, amely végrehajtja az ügyfélkódot. Ezeket a motorokat egyes szervereken és számos alkalmazásban is használják. A legnépszerűbb rendszer a nem-kliensoldali használathoz a Node.js.

### C#

A C# egy általános célú, magas szintű programozási nyelv, amely több paradigmát is támogat. A C# magában foglalja a statikus gépelést erős gépelést, lexikális hatókörű, felszólító, deklaratív, funkcionális, általános, objektum-orientált (osztály-alapú) és komponens-orientált programozási tudományágat.

### XAML

Az XAML egy deklaratív jelölőnyelv. A .NET programozási modellhez hasonlóan az XAML leegyszerűsíti a felhasználói felület létrehozását egy .NET-alkalmazáshoz.

### JSON

A JSON (JavaScript Object Notation) egy könnyű adatcsere-formátum. Az ember számára könnyű írni és olvasni. A gépek könnyen értelmezhetők és generálhatók. A JavaScript programozási nyelv szabvány ECMA-262 3. kiadás – 1999. december egy részhalmazán alapul. A JSON egy szövegformátum, amely teljesen nyelvfüggetlen, de olyan konvenciókat használ, amelyek a C nyelvcsalád programozói számára ismertek, beleértve a C, C++, C#, Java, JavaScript, Perl, Python és sok más nyelvet.

#### JSON Web Token

A JSON Web Token (JWT) egy kompakt, URL biztonságos eszköz a két fél között átadandó követelések megjelenítésére. A JWT-ben található követelések JSON-objektumként vannak kódolva, amely egy JSON Web Signature (JWS) struktúra hasznos terheléseként vagy egy JSON Web Encryption (JWE) struktúra egyszerű szövegeként használatos, lehetővé téve a követelések digitális aláírását vagy az integritás védelmét üzenet-hitelesítő kóddal (MAC) és/vagy titkosítással.

### MySQL

A MySQL szoftver egy nagyon gyors, többszálú, többfelhasználós és robusztus SQL (Structured Query Language) adatbázis-kiszolgálót biztosít. A MySQL Server kritikus fontosságú, nagy terhelésű termelési rendszerekhez, valamint tömegesen telepített szoftverekbe való beágyazáshoz készült. Az Oracle az Oracle Corporation és/vagy leányvállalatainak bejegyzett védjegye.

### phpMyAdmin

A phpMyAdmin egy PHP nyelven írt ingyenes szoftvereszköz, amely a MySQL webes adminisztrációjának kezelésére szolgál. A phpMyAdmin a MySQL és a MariaDB műveletek széles skáláját támogatja. A gyakran használt műveletek (adatbázisok, táblák, oszlopok, relációk, indexek, felhasználók, engedélyek stb. kezelése) végrehajthatók a felhasználói felületen keresztül, miközben továbbra is lehetősége van bármilyen SQL utasítás közvetlen végrehajtására.

### HTTP REST API

A REST API a REpresentational State Transfer API rövidítése. Ez egyfajta API (Application Programming Interface), amely lehetővé teszi a kommunikációt a különböző rendszerek között az interneten keresztül. A REST API-k úgy működnek, hogy kéréseket küldenek és válaszokat fogadnak, jellemzően JSON formátumban, az ügyfél és a szerver között. A REST API-k HTTP-módszereket (például GET, POST, PUT, DELETE) használnak az erőforrásokon végrehajtható műveletek meghatározásához. Ezek a módszerek igazodnak a CRUD (Létrehozás, Olvasás, Frissítés, Törlés) műveletekhez, amelyek az erőforrások weben történő manipulálására szolgálnak

### Laravel

A Laravel egy webalkalmazás-keretrendszer kifejező, elegáns szintaxissal. A webes keretrendszer struktúrát és kiindulópontot biztosít az alkalmazás létrehozásához.

### PHP

A PHP, amely egy széles körben használt nyílt forráskódú, általános célú szkriptnyelv, amely különösen alkalmas webfejlesztésre, és beágyazható a HTML-be. Szintaxisa C-re, Java-ra és Perl-re támaszkodik.

#### Faker

A Faker egy PHP-könyvtár, amely hamis adatokat generál, ezt kihasználva fel tudtuk tölteni tesztadatokkal a programjainkat, melyekkel könnyen tudtunk manuális teszteket végezni, emellett viszont a Laravel beépített tesztjeihez is Faker-t használtunk. A legtöbb képernyőkép az appról, valamint a weboldalról jó példa a manuális tesztekre.

### MAUI

A .NET többplatformos alkalmazás felhasználói felülete (.NET MAUI) egy platformfüggetlen keretrendszer natív mobil-és asztali alkalmazások C# és XAML használatával történő létrehozásához.

#### MVVM (Model VIew ViewModel) Community Toolkit

A CommunityToolkit.Mvvm csomag (más néven MVVM Toolkit, korábbi nevén Microsoft.Toolkit.Mvvm) egy modern, gyors és moduláris MVVM könyvtár. A .NET Community Toolkit része.

### Composer

A Composer egy PHP függőségkezelési eszköz. Lehetővé teszi, hogy deklarálja azokat a könyvtárakat, amelyektől a projektje függ, és kezeli (telepíti/frissíti) azokat a ön helyett.

## Adatbázis

Adatbázisunk kezeléséhez egy Laravel [REST API](#_REST_API)-t (6.4.4-es pont) használunk.

*Ahol képet küldünk vagy fogadunk, ott a képet az adatbázisban nem tároljuk, csak az api alap url-jéhez csatolandó feltétlen szükséges elérési útvonalat. A teljes elérési útvonalat a GET request adja vissza.*

Pl.: http://localhost:8069/storage/public/articles/articles\_a2b384ec-a788-45df-9d67-fe0cf2194da3.webp

### Adatbázis alapadatai

* Adatbázis szerver: MySQL
* Adatbázis neve: freyas\_garden
* Illesztés: utf8mb4\_unicode\_ci
* Motor: InnoDb SQL

### Táblák

#### Plants

A plants táblában általános adatokat tárolunk különféle növenyekről. A rendszer indulásakor statikus adatokkal töltődik fel. Oszlopai:

* id
  + Elsődleges kulcs
  + Szám
  + Pl.: 3
* name
  + Leíró tulajdonság
  + A növény neve magyarul
  + Szöveg
  + Pl.: Banán
* latin\_name
  + Leíró tulajdonság
  + A növény neve latinul
  + Szöveg
  + Pl.: Musa
* type\_id
  + Idegen kulcs
  + A növény típusának azonosító száma
  + Szám
  + Pl.: 1

#### Types

A plants táblához csatlakozik a types szótár tábla, mely a növény típusokat tárolja. A rendszer indulásakor statikus adatokkal töltődik fel. Oszlopai:

* id
  + Elsődleges kulcs
  + Szám
  + Pl.: 1
* name
  + Leíró tulajdonság
  + A típus neve
  + Szöveg
  + Pl.: gyümölcs

#### Users

A users táblában tároljuk a felhasználóink adatait. Dinamikusan feltöltött tábla, regisztráció esetén új rekord kerül bele. Oszlopai:

* id
  + Elsődleges kulcs
  + Szám
  + Pl.: 1
* username
  + Leíró tulajdonság
  + A fiók felhasználóneve
  + Szöveg
  + Pl.: admin
* email
  + Leíró tulajdonság
  + A felhasználó e-mail címe
  + Szöveg (e-mail)
  + Pl.: admin@freyasgarden.com
* city
  + Leíró tulajdonság
  + A felhasználó tartózkodási helye, területe
  + Szöveg
  + Pl.: Vác
* birthdate
  + Leíró tulajdonság
  + A felhasználó születési dátuma
  + Dátum
  + Pl.: 2004-07-20
* password
  + Leíró tulajdonság
  + A felhasználó jelszava
  + (HASH-kóddal titkosított) Szöveg
  + Pl.: $2y$12$9hr1rGGbJoAXnvpjd9AcQuGMLLPXw9xYZy0jQWkMYK9qV8/7Iutw6
* role\_id
  + Idegen kulcs
  + A felhasználó jogi szerepkörének azonosító száma
  + Szám
  + Pl.: 1
* pitcure
  + Leíró tulajdonság
  + A felhasználó profilképének választott képe
  + Elérési útvonalat leíró szöveg
  + Pl.: placeholders/Shovel.png
* description
  + Leíró tulajdonság
  + A felhasználó leírása
  + Szöveg
  + Pl.: Freya’s Garden fejlesztő, paradicsom termesztő
* remember\_token
  + Leíró tulajdonság
  + A felhasználó átmeneti azonosításához szükséges kód
  + Szöveg
  + Pl.: H7GNTScqql

#### Roles

A users táblához csatlakozik a roles szótár tábla, mely a [különféle szerepkörök](#_Használati_esetmodell,_szerepkörök)et (6.5-ös pont) tárolja. A rendszer indulásakor statikus adatokkal töltődik fel. Oszlopai:

* id
  + Elsődleges kulcs
  + A szerepkör azonosító száma
  + Szám
  + Pl.: 1
* name
  + Leíró tulajdonság
  + A szerepkör megnevezése
  + Szöveg
  + Pl.: admin

#### User\_plants

A user\_plants tábla tárolja a felhasználók számontartott növényeit, azoknak adatait. Kapcsoló táblaként is szolgál a users, plants, és listings táblák között. Oszlopai:

* id
  + Elsődleges kulcs
  + A számontartott növény(ek) azonosító száma
  + Szám
  + Pl.: 1
* user\_id
  + Idegen kulcs
  + A kapcsolódó felhasználó azonosító száma
  + Szám
  + Pl.: 62
* plant\_id
  + Idegen kulcs
  + A számontartott növény(ek) fajtája
  + Szám
  + Pl.: 90
* stag\_id
  + Idegen kulcs
  + A számontartott növény(ek) stádiumának azonosítója
  + Szám
  + Pl.: 5
* count
  + Leíró tulajdonság
  + A számontartott növény(ek) darabszáma
  + Szám
  + Pl.: 5

#### Stages

A user\_plants táblához csatlakozik a stages szótártábla, mely a különféle növény-életciklus stádiumokat tárolja. A rendszer indulásakor statikus adatokkal töltődik fel. Oszlopai:

* id
  + Elsődleges kulcs
  + A stádium azonosító száma
  + Szám
  + Pl.: 1
* name
  + Leíró tulajdonság
  + A stádium megnevezése
  + Szöveg
  + Pl.: mag

#### Listings

A listings táblában tároljuk a marketplace-en közzétett hirdetések adatait. Egy listing létezéséhez szükséges feltétel egy user\_plants létezése. Oszlopai:

* id
  + Elsődleges kulcs
  + A hirdetés azonosító száma
  + Szám
  + Pl.: 1
* user\_plants-id
  + Idegen kulcs
  + A hirdetésben szereplő számontartott növény(ek) azonosító száma
  + Szám
  + Pl.: 4
* title
  + Leíró tulajdonság
  + A hirdetés címe
  + Szöveg
  + Pl.: 2 kiló paradicsom!!!
* description
  + Leíró tulajdonság
  + A hirdetés leírása
  + Szöveg
  + Pl.: Nagyon finomak, érettek 😊
* city
  + Leíró tulajdonság
  + A hirdetéshez releváns terület
  + Szöveg
  + Pl.: Göd
* media
  + Leíró tulajdonság
  + A hirdetéshez csatolt kép(ek)
  + Elérési útvonalt leíró szöveg
  + Pl.: "[\"listings\\\/listings\_6f0c8869-82db-4d23-b605-e...
* price
  + Leíró tulajdonság
  + A meghirdetett ár
  + Szám
  + Pl.: 5000

#### Articles

Az articles táblában tároljuk a weboldalon megtalálható cikkek adatait. Oszlopai:

* id
  + Elsődleges kulcs
  + A cikk azonosító száma
  + Szám
  + Pl.: 7
* title
  + Leíró tulajdonság
  + A cikk címe
  + Szöveg
  + Pl.: Receptek répával
* plant\_id
  + Idegen kulcs
  + A cikkhez kapcsolódó növény azonosító száma, nem kötelező
  + Szám
  + Pl.: 34
* author\_id
  + Idegen kulcs
  + A cikk szerzőjének azonosító száma (a users táblára mutat)
  + Szám
  + Pl.: 1
* category\_id
  + Idegen kulcs
  + A cikk kategóriájának azonosító száma
  + Szám
  + Pl.: 2
* description
  + Leíró tulajdonság
  + A cikk rövid leírása
  + Szöveg
  + Pl.: Ezzel az öt recepttel feldobhatod a mindennapjaidat!
* content
  + Leíró tulajdonság
  + A cikk tartalma, markdown-ban tárolva
  + Hosszú szöveg
* source
  + Leíró tulajdonság
  + A cikk forrása, link, nem kötelező
  + Szöveg
  + pl.: http://ankunding.org/officia-deserunt-pariatur-totam-a-ipsam-maiores-minima

#### Categories

Az articles táblához csatlakozik a categories szótártábla, mely a különféle kategóriákat tárolja. A rendszer indulásakor statikus adatokkal töltődik fel. Oszlopai:

* id
  + Elsődleges kulcs
  + A kategória azonosító száma
  + Szám
  + Pl.: 3
* name
  + Leíró tulajdonság
  + A kategória megnevezése
  + Szöveg
  + Pl.: recept

A képen szöveg, képernyőkép, Betűtípus, szám látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

## A rendszer felépítése

A projekt egy Laravel REST API-ból, egy Laravel weboldalból, egy MAUI - fejlesztőknek szánt, elsősorban asztali applikációból (*Dev app*) és egy MAUI - felhasználóknak szánt, elsősorban mobil applikációból (*Marketplace*) áll.

### Marketplace

A Marketplace a fő működtető szerve a projektünknek, regisztrált felhasználóinknak elérhető. Itt hirdetéseket tekinthetünk meg, vagy közzé tehetünk saját hirdetéseket is.

### Weboldal

Weboldalunk fő célja az ismeretterjesztés. Bárki számára elérhető, nem szükséges a regisztráció. Itt különféle cikkeket tekinthetünk meg.

### Dev app

Az asztali alkalmazás a fejlesztők számára lesz elérhető. Itt, a megfelelő jogosultság birtokában, kezelhetjük az adatbázist.

### REST API

Mivel projektünk minden főbb része az adatbázissal dolgozik egy saját Web API használata általánosítja, kezelhetőbbé teszi a megvalósítást. Az API publikus, közösség számára is elérhető. Bizonyos funkciók (POST, PUT, DELETE) csak megfelelő jogosultsággal érhetők el. (lásd [Az API működése,](#_Adatbázis_kapcsolat) 6.6.1-es pont)

#### A .env fájl

A REST API és a WEB projekt könyvtárában megtalálható .env fájlban sok fontos adat beállítható.



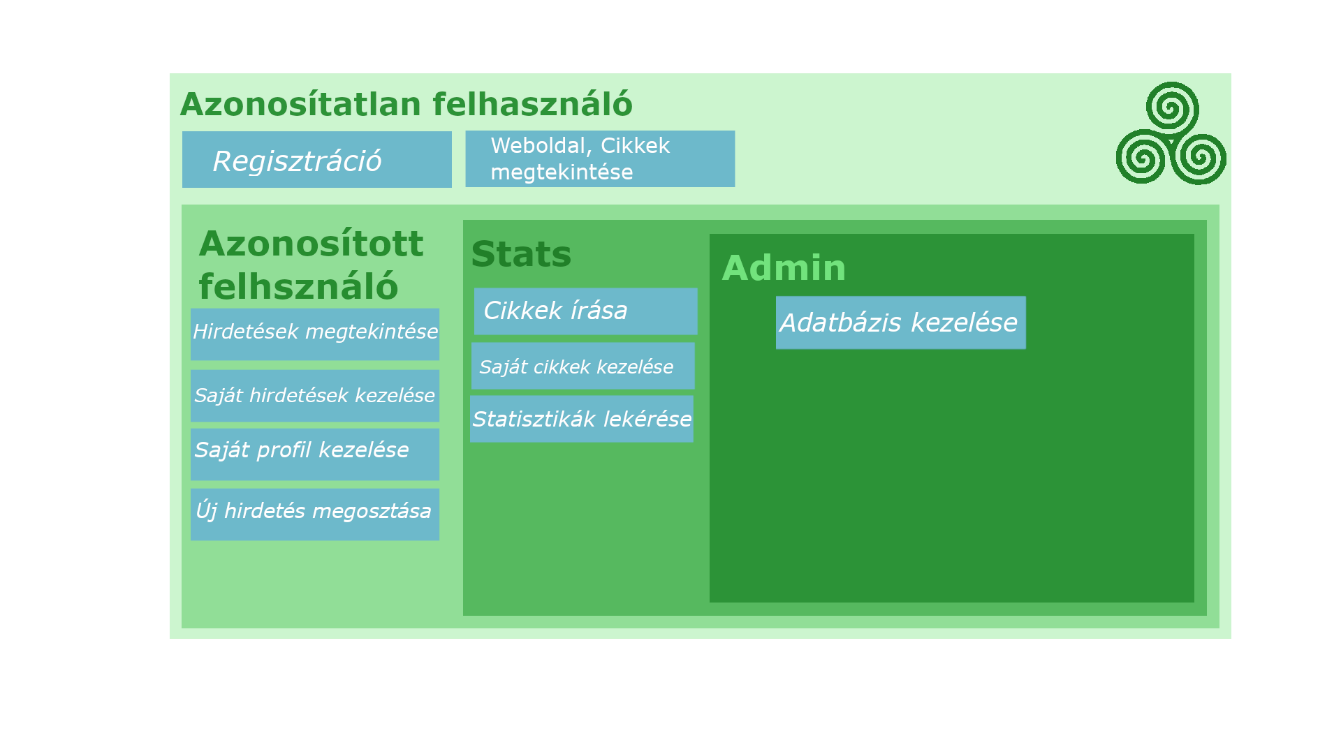
## Használati esetmodell, szerepkörök

A programjainkban a felhasználók különböző szerepkörökre vannak osztva, ezekkel a szerepkörökkel különböző jogosultságok járnak. A szerepkörök hierarchikusan épülnek egymásra, tehát egy magasabb szintű szerepkör rendelkezik az alatta lévő szerepkör jogosultságaival.

### Szerepkörök, jogosultságaik

1. Azonosítatlan felhasználó
   1. web app használata
      1. cikkek olvasása, keresése, szűrése
      2. asztali, telefonos app letöltése
   2. Marketplace használata
      1. regisztráció
      2. bejelentkezés
      3. új jelszó kérése (létező fiók esetén)
2. Azonosított felhasználó
   1. Marketplace használata
      1. hirdetések megtekintése, keresése
         1. egy hirdetés kiemelt megtekintése, hirdető adatok, elérhetőségek megtekintése
      2. új hirdetés közzététele
      3. saját hirdetések módosítása, törlése
      4. profil adatok módosítása
3. Stats
   1. Dev app használata
      1. Felhasználók megtekintése
4. Admin
   1. Dev app használata
      1. Felhasználói fiók törlése

### Szerepkörök halmazábrája



## A rendszer működése

### Adatbázis kapcsolat, Laravel

Az adatbázisunkat a Laravelben elkészített REST API-on keresztül kezeljük. Az API tölti fel adatokkal, valamint az kezeli a weboldalról, és az appokból érkező kéréseket. Ezzel központosított az adatbázis kapcsolat, könnyebben skálázható, módosítható, és kiegészíthető a későbbiekben.

#### Migrációk

Az API Laravelben kezelhető migrációkat használ. A migrációk felelősek a táblák létrehozásáért, definiálásáért, valamint a táblák közti kapcsolatok felépítésért.

#### Seeder-ek

A seeder-ek szintén Laravelben kezelhetők, ezek felelősek némely táblák autómatikus feltöltéséért indításkor.

### Felhasználók átmeneti azonosítása, végpontok biztonsága

A felhasználók azonosítása bearer token segítségével valósul meg ami minden egyes bejelentkezésnél generálódik az adott felhasználóhoz, valamint tartalmazza a felhasználó jogkörét. A token-eket adatabázisban tároljuk és minden ezt igényló végpont meghívása előtt egy middleware ellenőrzi a token érvényességét, valamint, hogy megfelelő jogosultágú e a felhasználó.

Az API végpontjait biztonság szerint 6 fajtára lehet osztani

#### Nyílt végpontok

Ezeket akárki elérheti bármilyen hitelesítés nélkül, ilyen útvonalak például a weboldalhoz használtak mivel azon a felületen nem lehet bejelentkezni és nem is lehet adatokat módosítani.

#### Felhasználók számára elérhető végpontok

Ezek a végpontok leginkább a mobilos alkalmazás használatához szükségesek, tehát felhasználói profil megtekintése módosítása és hirdetések megosztása és böngészése. Minden regisztrált és aktív token-nel rendelkező felhasználónak van joga ezeket a végpontokat használni.

#### Stats jogkörrel elérhető végpontok

Ezek a végpontok már csak Admin és Stats jögkörel rendelkező felhasználóknak érhetőek el és a dev appban cikkek létrehozására használhatóak.

#### Csak Admin jogkörrel elérhető végpontok

Kizárólag admin-oknak fenttartott végpontok az adatbázis hátramarad tábláinak módosítására, valamint felhasználói jogkörök megváltoztatására.

#### Saját vagy Admin

Ezek a végpontok például hirdetések és cikkek módosítására szolgálnak. A felhasználók értelem szerűen csak a saját hirdetéseiket tudják módosítani, viszont Admin-ok is belepiszkálhatnak, ha úgy látják jónak.

### elérhető kérések listája

A képen szöveg, képernyőkép, áramkör látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.

### API kérések működése, példák

#### REST jellegű működés

API-nk egy REST elvein alapuló kommunikációs felületet biztosít, ami azt jelenti, hogy a rendszer erőforrás-orientált megközelítést alkalmaz a HTTP metódusok (GET, POST, PUT, DELETE) segítségével. Bár nem minden esetben követjük a REST teljes mértékű specifikációját, alapul vettük annak főbb elveit az egyszerűség és egységesség érdekében.

#### Válaszok struktúrája

Minden API hívás során a visszaadott válasz egységes struktúrában érkezik, mely a következő kulcsokat tartalmazza:

* **status:** A kérés eredményének kódja (HTTP státuszkód).
* **message:** Informatív üzenet, mely segít az esetleges hibák vagy sikeres műveletek értelmezésében.
* **data:** Az API által szolgáltatott tényleges adatokat tartalmazó objektum.

#### Base URL

A példa dokumentációnkban az alap URL: http://localhost:8069/api/

Fontos megjegyezni, hogy a *localhost* helyettesíthető a megfelelő domain névvel vagy IP címmel abban az esetben, ha nem helyi környezetben futtatjuk az API-t.

#### GET kérések

Egyes adatmodell esetek lekérése például:   
GET http://localhost:8069/api/article/{cikkCím}

Ezzel a kéréssel egy cikket kérhetünk le az adatbázisból a címe alapján, amit az URL végén adunk meg.

A kérés fejében érdemes megadni az elvárt választ (Content-type: Apllication/json) ezt elhalasztva a laravel autómatikusan visszadhat egy HTML filet ami okozhat hibákat. Mint azt GET kéréseknél szokás a kérés testében nem kell semminek lennie.

Példa válasz egy ilyen sikeres kérésre:   
{

    "status": 200,

    "message": "{cikkCím} article found",

    "data": {

"id": Cikk elsődleges kulcsa,

"title": "Cikk címe({cikkCím})",

"category": "Cikk kategóriája",

"description": "Rövid leírás a cikkről",

"updated\_at": "Utolsó frissítés dátuma",

"plant\_name": "A cikkhez kapcsolt növény neve",

"type": növény típusa,

"author": "A felhasználó, aki létrehozta a cikket",

"source": "források a cikkhez",

"content": "A tartalom mardown formátumban, és kép linkekkel a szövegbe ágyazva"

"created\_at": "Cikk létrehozásának átuma"

}

}

Sikerlen kérésre példa (nem talált erőforrás)

{

"status": 404,

"message": "Article not found",

"data": {}

}

#### Post kérések

Új erőforrások létrehozására használjuk ezeket az útvonalakat  
POST http://127.0.0.1:8069/api/regiser.

Ennek az kérésnek a fejlécében ugyanúgy érdemes megadni a válasz formátumát. Valamint Ennek a kérésnek a testében meg kell adni a bejelentkezési adatokat, valamint az email címet json formátumban:

{

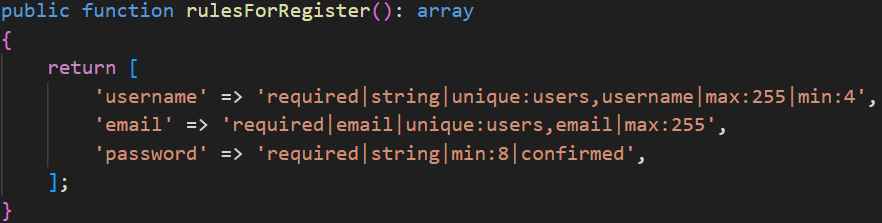
"username": "új felhasználónk felhasználóneve (egyedi)”,

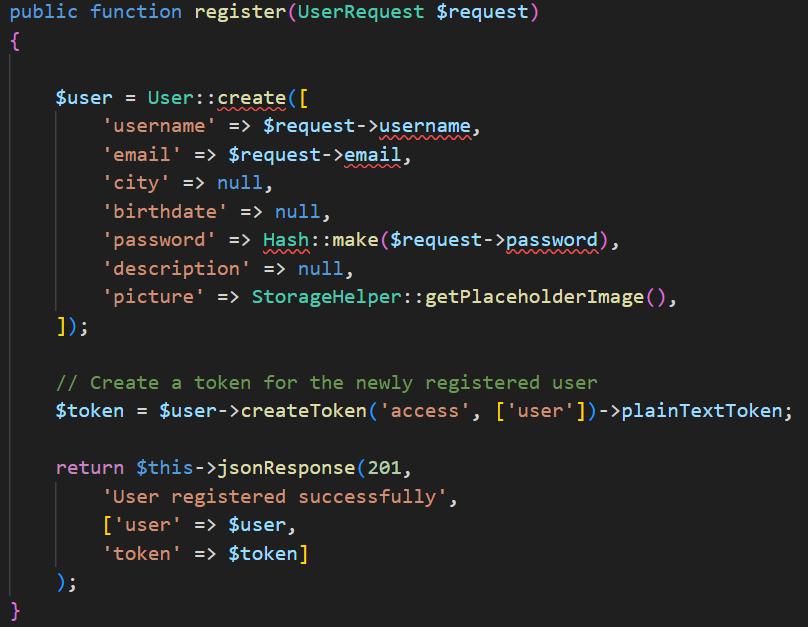
"email": "új felhsználónk email címe (egyedi)",

"password": "Új felhasználónk jelszava",

"password\_confirmation": "Jelszó újra",

}

Ezek az adatok a laravel beépített request osztályával validálásra kerülnek:  


Majd, ha minden rendben volt az adatokkal az új rekordot a controller-ben létrehozzuk és hozzáadjuk az adatbázishoz:   


Ez a funkció belépési tokent is létrehoz a felhasználónak és végül visszaküldi az új objektumot json formájában:   
{

"status": 200,

"message": "User registered successfully",

"user": {

"id": új felhasználó elsődleges kulcsa,

"username": "Megadott felhasználónév",

"email": "Megadott email cím"

},

"token": "1|abcdef1234567890" – autómatikusan generált bearer token – a bejelentkezést igénylő kérések fejlécében ezt kell megadni Bearer-token: után.

}

### Az MVVM és MVC architektúra

A két rövidítés egy-egy architektúrát takar, melyek segítségével hatékonyabban, rendezettebben, és átláthatóbban tudjuk felépíteni az adott programjainkat. A program részei jól elkülönülnek feladatukat és helyüket tekintve.

#### MVVM

(Model View ViewModel) Ezt a Marketplace-ben, és a Dev-appban használjuk

#### MVC

(Model View Controller) Ezt az API-ban\*, valamint a weboldalban használjuk

Mindkét esetben hasonló a jelentés, és az elemek funkciói.

* Model

A model-ek feladata az adott objektum vagy típus adatainak, tulajdonságainak leírása, rendszerezése.

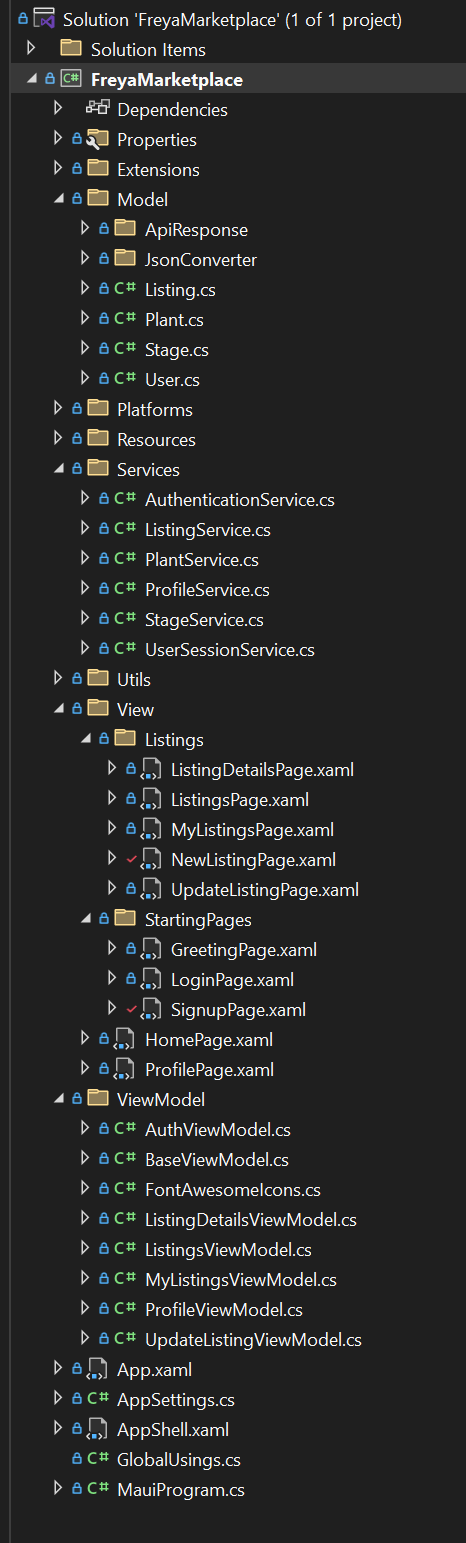
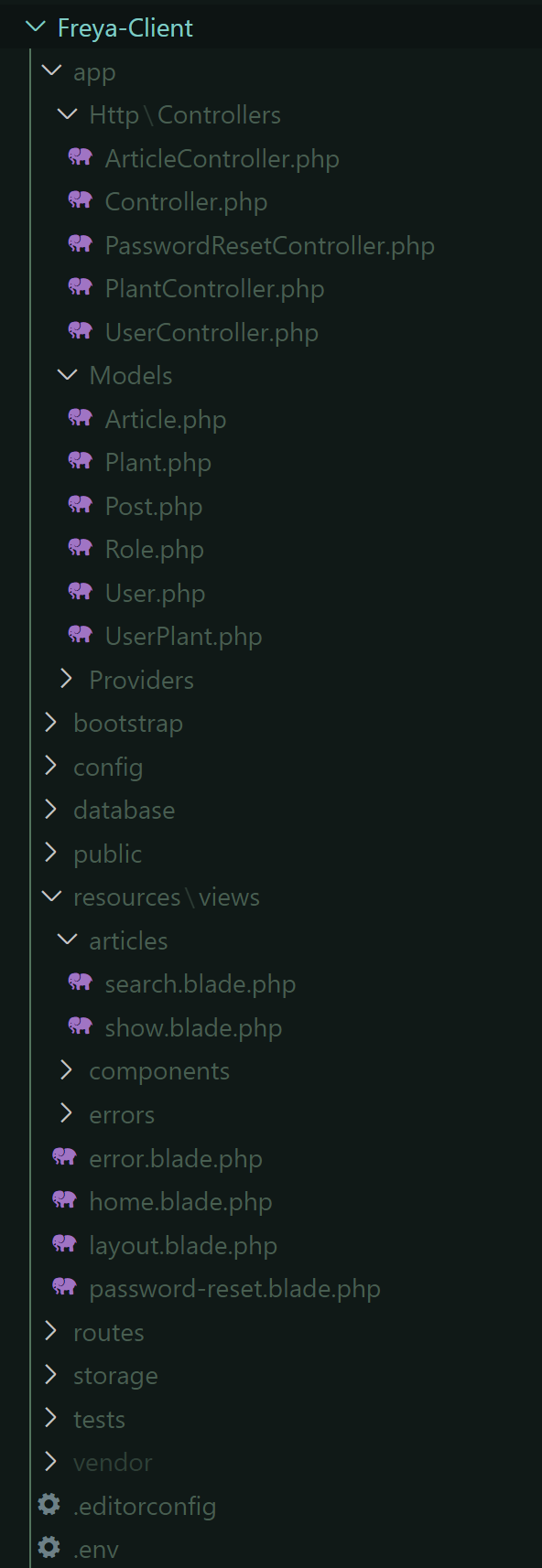
* View

A view-k a felhasználó által látott nézetek, túlnyomóan logikai műveletek nélküli frontend fájlok.  
\*Az API-ban nem használunk view-kat, hiszen annak nincs frontend-része.

* ViewModel/Controller

Feladatuk a logikai műveletek kezelése, a View-k és a Modelek összekapcsolása. Emellett használtunk még Service-eket, melyeket főleg az API-val való kommunikációért felelnek. (pl. Hirdetések lekérése). A Service-ben megkapott adatokat a Viewmodelnek továbbítjuk.

Alábbiakban képernyőképek láthatók az architektúra fájljairól a marketplace, valamint a weboldal esetében.

### Databinding

A *databinding* (adatkötés) segítségével a View és a ViewModel közötti kapcsolatot hatékonyan és átláthatóan tudjuk megvalósítani. Lehetővé teszi, hogy a felhasználó által látott adatok automatikusan frissüljenek, amikor az alatta lévő adat is változik. A kapcsolat mindkét irányban fentálhat (pl. Adatbeviteli mezők esetén)

A Binding különösen hasznos MVVM architektúrában, ahol a ViewModel szolgál a logika központjaként, és a nézetnek nem kell külön kódot írni az adatok megjelenítéséhez vagy frissítéséhez.

MAUI-ban a XAML nyelv segítségével könnyedén megadhatjuk, melyik elem milyen adatforráshoz legyen kötve. Például:

  
  
*A fenti példában a Listing.City értéke automatikusan megjelenik a nézetben, a megadott formátumban. Ha az érték a háttérben változik, a nézet automatikusan frissül.*

Communitytoolkit.Maui  
  
A programban a CommunityToolkit.Mvvm könyvtárat is használjuk, amely jelentősen megkönnyíti a ViewModel-ek létrehozását és az adatkötések működését. Ez a csomag többek között az [ObservableProperty] és [RelayCommand] attribútumokat biztosítja, amelyek csökkentik a felesleges kód mennyiségét, és tisztább, karbantarthatóbb ViewModel-eket eredményeznek.

## Tesztelés

### API tesztelés

#### ECHO API

Az [ECHOAPI](#_EchoAPI)-t használtuk az API manuális tesztelésére a fejlesztés során, mivel ezzel az alkalmazásoktól függetlenül tudtuk a végpontokat tesztelni.

## Továbbfejlesztési lehetőségek

* Letölthető appok
* Bejelentkezés Google-el
* Web
  + Forum
  + Bejelentkezés, ezáltal személyre szabott cikkek
* Dev APP
  + Felhasználói adatok módosítása
  + Grafikus kimutatások
  + Cikkek kezelése, létrehozása
  + Hirdetések kezelése
* Marketplace
  + Csevegés vagy üzenetküldés
  + Hirdetés létrehozása, számontartása közzététel nélkül

# Felhasználói dokumentáció

A továbbiakban szerepelő képernyőképeken megjelenő szövegek és beágyazott képek egy része [faker](#_Faker)-rel (5.2.11) létrehozott [teszadatt](#_Tesztelés) (5.7), ezáltal nem valós.

## A rendszer használata

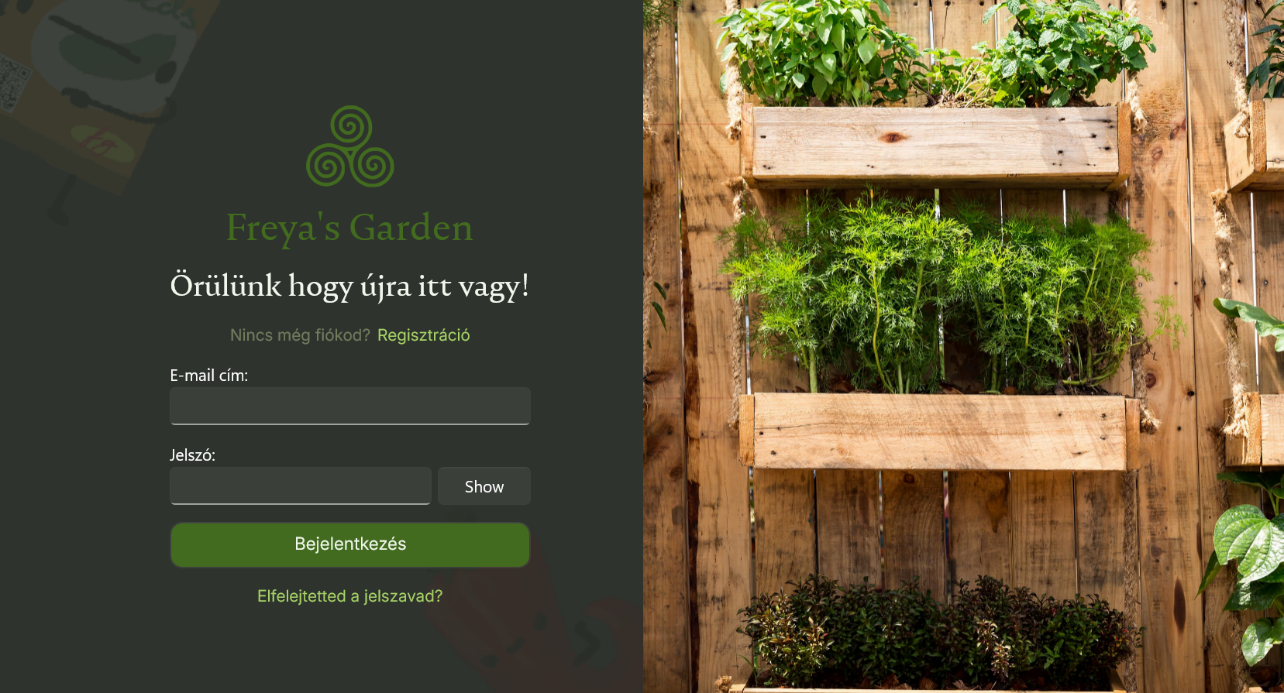
### MARKETPLACE

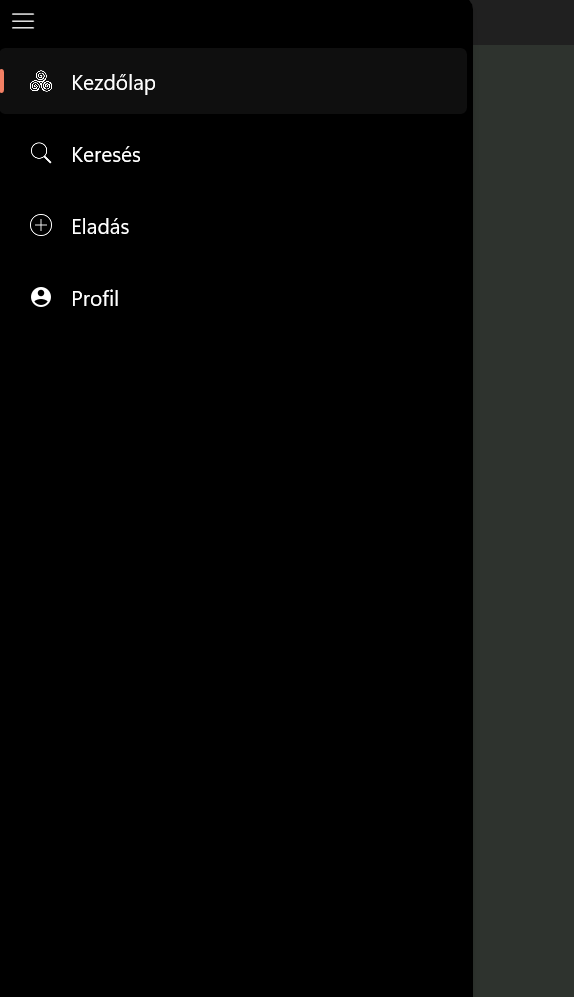
Az app megnyitásakor először regisztráció, avagy már létező felhasználói fiókkal való bejelentkezés szükséges. (lásd [Használati esetmodell, szerepkörök](#_Használati_esetmodell,_szerepkörök) (6.5)) Enélkül nem lehet használni az app többi funkcióját. Ezután a menüvel, gombokkal vagy más funkciókkal navigálhatunk az appon belül.

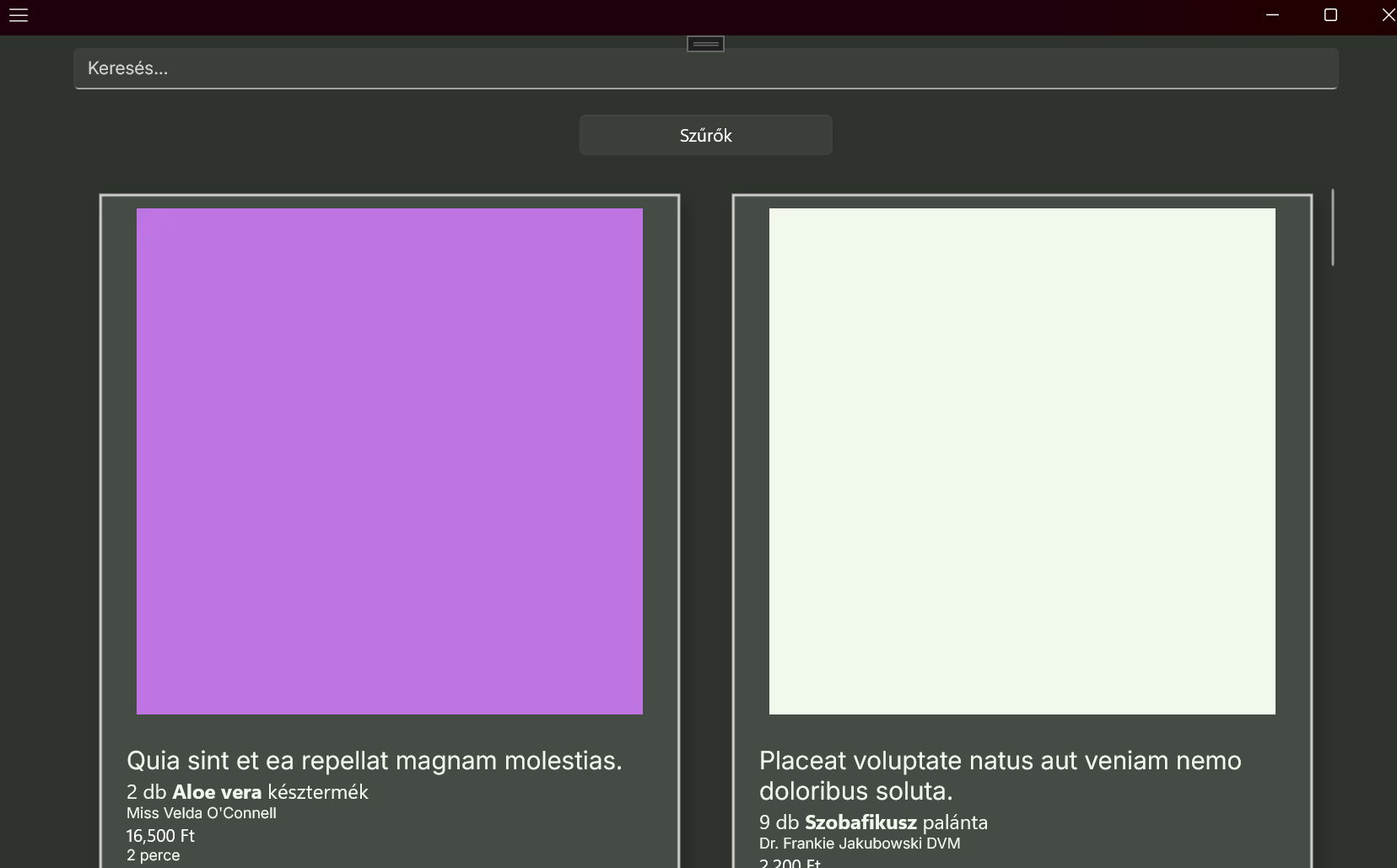
* Bejelentkezés
  + jelszó módosítása (weben keresztül)
* Profil
  + Személyes adatok megadása, módosítása
    - Lakhely, terület
    - Elérhetőségek
    - E-mail cím
  + Saját hirdetések megtekintése
* Piac
  + Hirdetések megtekintése
    - Kártyákon megjelenő hirdetések
      * Kép(ek)
      * Általános leírás, adatok
      * Hirdető megtekintése gomb
    - Szöveges keresősáv
  + Egy hirdetés adatai
    - Hirdető adatai
    - Ár
    - Leírás
    - További adatok
* Új hirdetés megosztása
  + Hirdetés adatai
    - Kép(ek)
    - Leírás
    - Név
    - Növény/hirdetett dolog fajtája

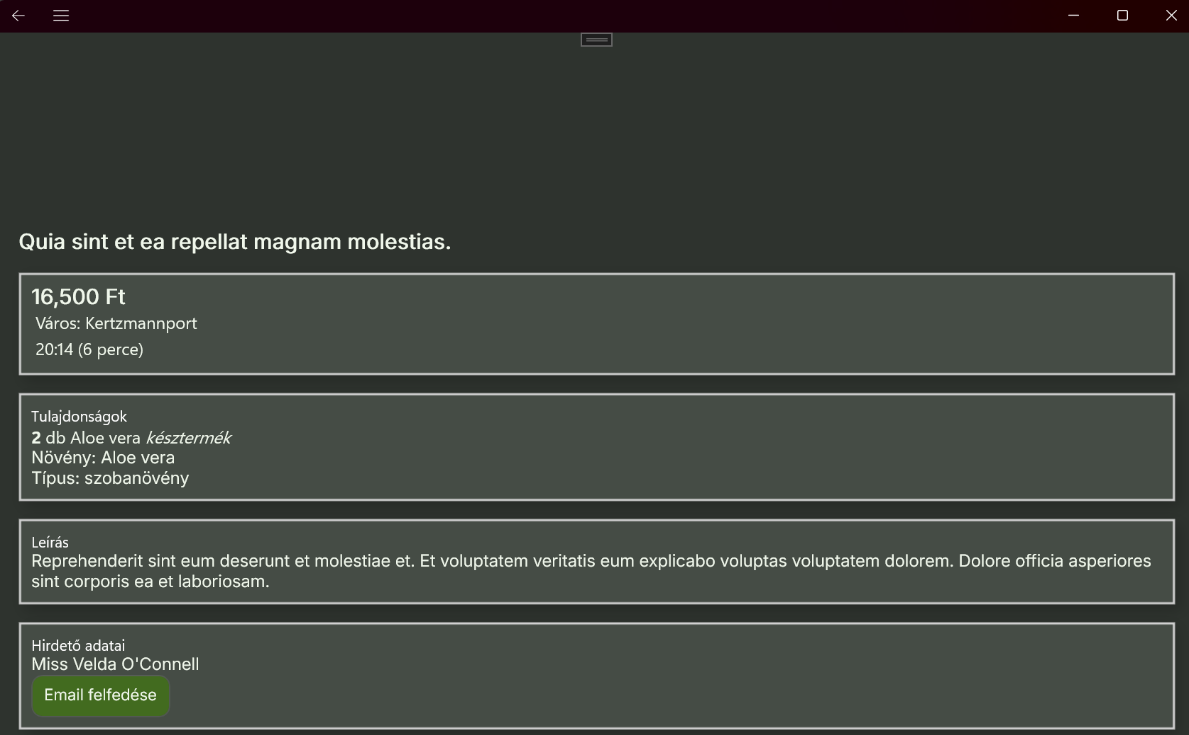
  
Marketplace fogadóoldal

  
Regisztráció

  
Bejelntkezés

  
Menü

  
Piac oldal, generált tesztadattokkal

  
Egy hirdetés adatai

  
Új hirdetés létrehozása

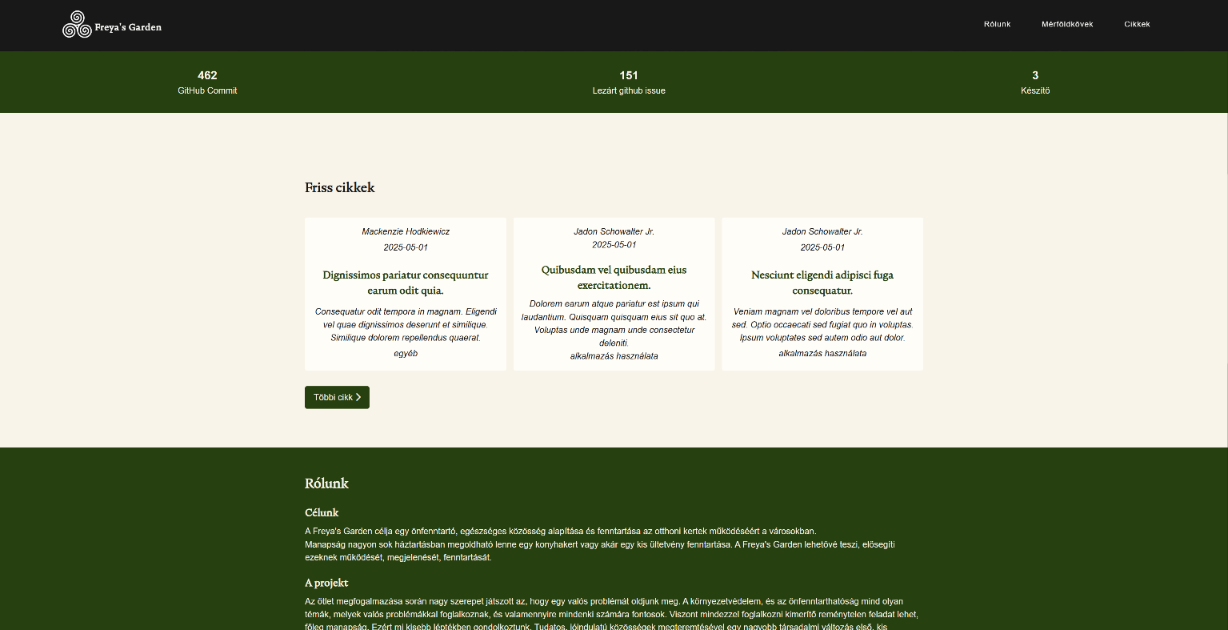
### WEB

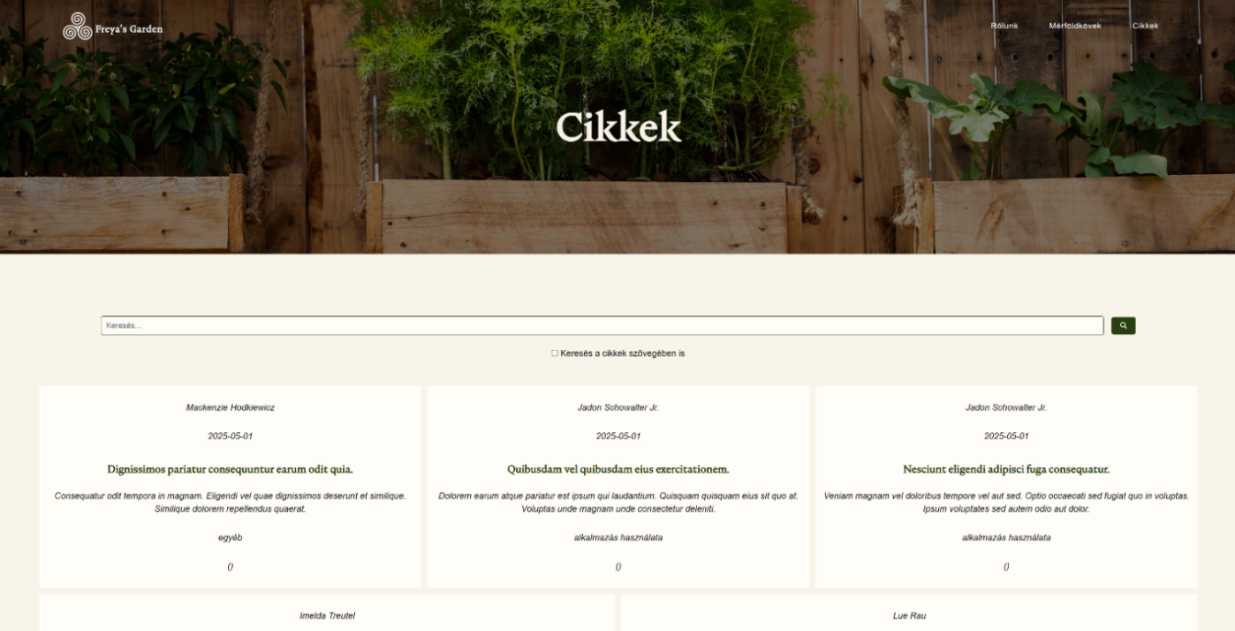
A weboldalunk minden oldalán megtalálható a navigációs sáv, aminek segítségével könnyen lehet navigálni az oldalak között. Regisztráció és bejelentkezés nélkül vehető igénybe.

* Főoldal
  + GitHub elérése
  + Általános információk
    - bemutatkozás
    - elérhetőségek
    - érdekességek
  + Friss cikkek
  + Cikkek
    - keresés
    - szűrés
    - egy cikk megtekintése

A képen kültéri, kerítés, kert, épület látható

Előfordulhat, hogy a mesterséges intelligencia által létrehozott tartalom helytelen.  
A web főoldala

  
A cikkek ajánlásai megjelenési dátum alapján

   
Cikkek böngészése

  
Cikkek szűrése és keresése

  
Egy cikk megtekintése

### DEV

A DEV-app megnyitásakor, a Marketplace-hez hasonlóan bejelentkezés szükséges, itt viszont csak a Stats és az Admin jogosultsággal lehet az alkalmazást igénybe venni. (lásd 6.5 [Használati esetmodell, szerepkörök](#_Használati_esetmodell,_szerepkörök))

* Felhasználók megtekintése (legalább Stats jogosultság)
* Adatbázis kezelés (Admin jogosultság)
  + Felhasználók törlése

## A rendszer indítása

A rendszer összes eleme az API-on keresztül éri el az adatbázis adatai tehát azt akármelyik alkalmazás használata előtt ajánlott elindítani

### API

1. Töltsük le az api-t a [githubról](#_Forráskód_(github)) ...\xampp\htdocs\ mappába
2. Indítsuk el a XAMPP-ban az Apache-t és a MySQL-t
3. kommenteljük ki az Apache konfigurációs filejában php.ini-ben az ;extensions:gd-t ha képeket akarunk tárolni
4. Futtassuk a Laravel működéséhez szükséges parancsokat a projekt mappájában
   1. composer install (composer update)
   2. php artisan migrate –seed
   3. php artisan storage:link
   4. php artisan serve --port 8069
5. Az API útvonalak dokumentációja elérhető a <http://127.0.0.1:8069/apidoc/index.html> címen
6. Az API útvonalai elérhetőek a <http://127.0.0.1:8069/api/> kezdetű címeken
7. Ajánlott pl. Postman, EchoAPI használata teszteléshez

### WEB

1. Az API futtatása után a weboldal letöltése a [linkről] a ...\xampp\htdocs\ mappába.
2. Futtassuk a Laravel működéséhez szükséges parancsokat a projekt mappájában.
   1. composer install (composer update)
   2. php artisan serve
3. A weboldal megnyitása a <http://127.0.0.1:8000/> címen történik és az API-t próbálja használni a <http://127.0.0.1:8069/api/> címről

# Források

#### <https://en.wikipedia.org/wiki/Visual_Studio_Code>

* <https://visualstudio.microsoft.com/#vs-section>
* <https://www.phpmyadmin.net/>
* [https://www.echoapi.com/wiki/docs/introduct ion](https://www.echoapi.com/wiki/docs/introduct%20%20ion)
* <https://en.wikipedia.org/wiki/GIMP>
* <https://hu.wikipedia.org/wiki/HTML>
* <https://en.wikipedia.org/wiki/CSS>
* <https://en.wikipedia.org/wiki/JavaScript>
* <https://en.wikipedia.org/wiki/C_Sharp_(programming_language)>
* <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-9.0>
* <https://www.json.org/json-en.html>
* <https://www.rfc-editor.org/info/rfc7519>
* <https://dev.mysql.com/doc/refman/8.0/en/introduction.html>
* <https://www.geeksforgeeks.org/rest-api-introduction/>
* <https://laravel.com/docs/12.x#meet-laravel>
* <https://www.php.net/manual/en/preface.php>
* <https://inkscape.org/about/>
* <https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/#additional-resources>
* <https://getcomposer.org/doc/00-intro.md>
* <https://fakerphp.org/>
* <https://youtu.be/DuNLR_NJv8U?si=pc8pqLyUVaKG1k65>
* <https://github.com/features/copilot>
* <https://chat.deepseek.com/>
* <https://chatgpt.com/>
* <https://www.figma.com/>
* <https://github.com/dotnet-presentations/dotnet-maui-workshop?tab=readme-ov-file>
* <https://www.youtube.com/watch?v=DuNLR_NJv8U>

Az oldalakon az idegen nyelven megjelenő szövegeket a GoogleChrome automatikus fordítórendszerével fordítottuk magyarra.

# Mellékletek

## Forráskód (github)

* Közös helyen megtalálható linkek:  
  <https://github.com/stars/cerberus2477/lists/freya-s-garden>
* API:  
  <https://github.com/cerberus2477/freya-rest-kola>
* Web:  
  <https://github.com/cerberus2477/freya-client-kola>
* Marketplace:  
  <https://github.com/cerberus2477/freya-marketplace>
* Dev-app:  
  <https://github.com/cerberus2477/freya-dev>
* Általános:  
  <https://github.com/OOGGZZII/plamnt>