

Állatmenhely Örökbefogadás Rendszer

Tartalomjegyzék

1. Bevezetés

- 1.1. A projekt célja
- 1.2. A projekt témaválasztásának indoklása
- 1.3. A fejlesztőcsapat szerepe és munkamegosztása

2. Követelményanalízis

- 2.1. Funkcionális követelmények
- 2.2. Nem funkcionális követelmények
- 2.3. Felhasználói szerepkörök és jogosultságok

3. Tervezés

- 3.1. Rendszerarchitektúra
- 3.2. Adatbázis-terv (ER-diagram, relációk)
- 3.3. Backend tervezése (Laravel)
- 3.4. Frontend tervezése (Blade sablonok, Bootstrap/ Tailwind / Vue.js, ha használtok)
- 3.5. Felhasználói felület vázlatai

4. Megvalósítás

- 4.1. Fejlesztési környezet (IDE, verziókezelés, Laravel verzió, MySQL)
- 4.2. Backend komponensek
 - Állatok kezelése (CRUD)
 - Képfeltöltés és tárolás
 - Örökbefogadási űrlap feldolgozása és emailküldés

- Látogatási időpont foglaló rendszer (naptár)
 - 4.3. Frontend komponensek
- Reszponzív megjelenítés (asztali és mobil)
- Űrlapok validációja
- Felhasználói élmény (UX)
 - 4.4. Felhasználói autentikáció és jogosultságkezelés
 - 4.5. Emailküldés technológiája (SMTP, Laravel Mail)

5. Tesztelés

- 5.1. Tesztelési stratégia
- 5.2. Manuális tesztesetek (pl. űrlapkitöltés, időpontfoglalás, kép feltöltés)
- 5.3. Automatikus tesztek (PHPUnit, Laravel Feature/Unit tesztek)
- 5.4. Teszteredmények dokumentációja

6. Dokumentáció

6.1. Felhasználói dokumentáció

- Regisztráció és belépés
- Állat örökbefogadása
- Időpontfoglalás
- Email visszaigazolás
 - 6.2. Adminisztrátori dokumentáció
- Új állat felvitele
- Adatok módosítása és törlése
- Látogatási időpontok kezelése

7. Eredmények és értékelés

- 7.1. A szoftver erősségei
- 7.2. Fejlesztés közben felmerült kihívások
- 7.3. További fejlesztési lehetőségek

8. Összefoglalás

9. Irodalomjegyzék

10. Függelék

- ER-diagram
- Adatbázis dump
- Kód részletek
- Tesztelési jegyzőkönyv
- Képernyőfotók

1. A projekt célja

A projekt célja egy olyan webalapú alkalmazás kifejlesztése, amely megkönnyíti és digitalizálja a kisállatok örökbefogadásának folyamatát. Az alkalmazás központi funkciója, hogy átlátható és felhasználóbarát módon tegye lehetővé a menhelyek és állatvédő szervezetek számára az állatok adatainak rögzítését, a potenciális örökbefogadók számára pedig az állatok közötti böngészést, valamint az örökbefogadási szándék jelzését.

A fejlesztés központi motivációja, hogy a hagyományos, papíralapú vagy személyes ügyintézésen alapuló örökbefogadási folyamat sok esetben nehézkes, lassú és átláthatatlan. Az állatvédő szervezetek gyakran nem rendelkeznek olyan digitális eszközökkel, amelyek segítségével könnyen kezelhetnék az állatok adatait, képeit, valamint az érdeklődők jelentkezéseit. A rendszer ezen hiányosságokat hivatott pótolni, modern technológiák felhasználásával.

A projekt konkrét céljai a következők:

- **Állatok nyilvántartása:** minden egyes állathoz tartozzon egy adatlap (név, faj, részletes leírás, több kép).
- **Örökbefogadási jelentkezés:** a felhasználók egy űrlap segítségével jelentkezhetnek, amely automatikus emailt küld az állatvédő szervezetnek, és visszaigazolást generál az örökbefogadónak.
- **Időpontfoglalás:** az érdeklődők a rendszer naptárfelületén keresztül látogatási időpontot foglalhatnak.
- **Rezonansív felület:** az alkalmazás minden modern eszközön – legyen az számítógép, tablet vagy okostelefon – kényelmesen használható legyen.
- **Adatbiztonság és hitelesítés:** csak regisztrált, hitelesített felhasználók adhassanak le örökbefogadási jelentkezést.

- **Adminisztrátori felület:** a menhely munkatársai képesek legyenek új állatokat rögzíteni, a meglévő adatokat módosítani, valamint a foglalások nyilvántartását kezelní.

A fenti célok teljesítésével egy olyan modern, könnyen használható és biztonságos szoftver jön létre, amely minden állatvédő szervezetet, minden az örökbefogadók számára egyszerűsíti és átláthatóbbá teszi az örökbefogadási folyamatot, hozzájárulva a felelős állattartás és az állatvédelem társadalmi támogatásához.

1.2. A projekt téma választásának indoklása

A kisállatok örökbefogadása napjainkban egyre fontosabb társadalmi kérdéssé válik. A menhelyeken és állatvédő szervezeteknél évente több ezer állat vár új otthonra, ugyanakkor az örökbefogadási folyamat sokszor bonyolult, időigényes és nem minden átlátható a jelentkezők számára. Számos menhely jelenleg is elsősorban személyesen vagy telefonon keresztül intézi az örökbefogadási ügyeket, ami egyszerűbb adminisztrációs terhelést jelent a szervezeteknek, másrészről az érdeklődők számára is kevésbé kényelmes.

A digitalizáció lehetőségeinek kihasználása ezen a területen kiemelten indokolt. Egy online rendszer képes arra, hogy egyszerűsítse az információáramlást, csökkentse az adminisztrációs hibák esélyét, valamint átláthatóbbá tegye az örökbefogadási folyamatot. A rendszer emellett lehetőséget biztosít a potenciális gazdáknak arra, hogy kényelmesen, otthonról böngésszenek a különböző állatok között, így nagyobb eséllyel találnak rá a számukra megfelelő kedvencre.

A téma választása továbbá szakmai szempontból is indokolt. Egy kisállat örökbefogadási weboldal fejlesztése több különböző informatikai területet ötvöz:

- **Adatbázis-kezelés:** az állatok adatainak, képeinek és a jelentkezéseknek a tárolása.
- **Backend fejlesztés:** az örökbefogadási logika, emailküldés, időpontfoglalás kezelése.
- **Frontend fejlesztés:** a felhasználói felület kialakítása, reszponzív design megvalósítása.
- **Biztonság:** felhasználói regisztráció, hitelesítés és jogosultságkezelés.
- **Tesztelés:** a rendszer működésének ellenőrzése manuális és automatikus módszerekkel.

Ezáltal a projekt komplexitása biztosítja, hogy a fejlesztői csapat széles körű szakmai tapasztalatot szerezzen a webfejlesztés különböző rétegeiben.

További indokként megemlíthető a társadalmi hasznosság: az alkalmazás hozzájárulhat ahhoz, hogy több kisállat találjon szerető gazdára, miközben a menhelyek számára is egy korszerű, könnyen kezelhető adminisztrációs felületet biztosít. A fejlesztés így nem csupán technikai kihívásokat jelent, hanem egy valós, életszerű problémára kínál megoldást, amely közvetetten állatvédelmi és társadalmi célt is szolgál.

1.3. A fejlesztőcsapat szerepe és munkamegosztása

A projektet egy **3 fős fejlesztői csapat** valósítja meg, amelynek tagjai a szoftverfejlesztési folyamat különböző területeiért felelősek. A munkamegosztás célja az volt, hogy minden csapattag a saját erősségeinek megfelelő feladatakon dolgozhasson, ugyanakkor betekintést nyerjen a teljes fejlesztési ciklus minden fontosabb elemébe. Ez biztosítja a kiegyensúlyozott munkaterhelést, valamint azt, hogy mindenki átfogó szakmai tapasztalatot szerezzen.

A fejlesztőcsapat főbb szerepei és felelősségi körei a következők:

- **Projektvezető / Backend-fejlesztő**
 - A fejlesztés koordinálása, a feladatok kiosztása és a határidők betartatása.
 - A rendszer szerveroldali logikájának megvalósítása Laravel keretrendszerben.
 - Az adatbázis megtervezése és integrációja MySQL segítségével.
 - Az örökbefogadási űrlap feldolgozásának, valamint az emailküldés funkcióinak kialakítása.
 - Biztonsági és hitelesítési mechanizmusok implementálása.
- **Frontend-fejlesztő**
 - A felhasználói felület kialakítása reszponzív webes technológiák (HTML, CSS, JavaScript, Bootstrap/Tailwind) felhasználásával.
 - A kliensoldali logika megvalósítása (űrlapok validációja, naptárfunkciók kezelése).
 - A webes felület felhasználóbarát kialakítása és a felhasználói élmény javítása.
 - Képfeltöltés és megjelenítés integrálása a felhasználói felületen.
- **Tesztelő / Dokumentációfelelős**
 - Tesztesetek megírása és végrehajtása (manuális és automatikus tesztek Laravel PHPUnit keretrendszerrel).
 - Teszteredmények gyűjtése és dokumentálása.
 - A projekt teljes dokumentációjának elkészítése (technikai leírás, használati útmutató, tesztelési jegyzőkönyv).
 - Képernyőfotók, adatbázis-diagramok és illusztrációk összeállítása a szakdolgozat részeként.

A csapattagok közötti együttműködés alapját a **verziókezelés GitHub segítségével** biztosítja, amely lehetővé teszi a kód közös fejlesztését, a változások nyomon követését, valamint a hibák gyors javítását. A fejlesztés során a csapat rendszeresen konzultál

egymással, megvitatja a felmerülő problémákat, és közösen dönt a rendszer architekturális kérdéseiben.

A munkamegosztás rugalmas, azaz minden tag lehetőséget kap más területek kipróbálására is. Ezáltal mindenki részt vett a backend és frontend fejlesztésben, valamint a tesztelésben és dokumentálásban is, de az egyes szerepkörök meghatározása hozzájárult a projekt hatékony előrehaladásához.

2.1. Funkcionális követelmények

A rendszer elsődleges célja, hogy támogassa a kisállatok örökbefogadásának folyamatát, és ehhez kapcsolódóan biztosítsa az összes szükséges funkciót mind a felhasználók, mind az adminisztrátorok számára. A funkcionális követelmények azok a szolgáltatások és viselkedési elvárások, amelyeket az alkalmazásnak feltétlenül biztosítania kell.

A rendszer főbb funkcionális követelményei a következők:

1. Felhasználói regisztráció és hitelesítés

- A látogatók számára lehetőség nyílik regisztrációra, amelyhez név, email-cím és jelszó megadása szükséges.
- A bejelentkezett felhasználók több jogosultságot élveznek, például örökbefogadási jelentkezést adhatnak le és időpontot foglalhatnak.
- Az alkalmazásnak kezelnie kell a jelszó-helyreállítást is.

2. Állatok nyilvántartása

- Az adminisztrátorok új állatokat vihetnek fel a rendszerbe a következő adatok megadásával:
 - név
 - faj
 - részletes leírás
 - több kép feltöltése
- A felhasználók listázva böngészhetik az elérhető állatokat, és részletes adatlapot tekinthetnek meg róluk.

3. Örökbefogadási jelentkezés

- minden állat adatlapján elérhető egy „Örökbefogadás” gomb, amely egy űrlaphoz vezet.

- Az űrlap kitöltéséhez szükséges adatok:
 - örökbefogadó neve
 - email-címe
 - telefonszáma
 - lakcíme
 - rövid indoklás (miért szeretné örökbefogadni az adott állatot)
- Az űrlap beküldésekor a rendszer automatikusan emailt küld az állatvédő szervezet hivatalos email címére az adatokkal.
- Egyidejűleg automatikus visszaigazolást kap a jelentkező, amely tartalmazza a „Jelentkezést rögzítettük” üzenetet.

4. Időpontfoglalás látogatásra

- A rendszer tartalmaz egy naptárfelületet, amelyen a felhasználók látogatási időpontot foglalhatnak.
- A foglalás során a felhasználó kiválasztja a dátumot és az időpontot, amely a rendszerben rögzítésre kerül.
- Az adminisztrátorok számára láthatóvá válik az összes beérkezett foglalás, és szükség esetén módosíthatják vagy törölhetik azokat.

5. Információs oldalak

- Az alkalmazás tartalmaz fix, statikus tartalmú oldalakat is:
 - **Rólunk** – bemutatja a szervezetet és tevékenységeit.
 - **Kapcsolat** – tartalmazza az elérhetőségeket és egy kapcsolatfelvételi űrlapot.

6. Adminisztrációs funkciók

- Az adminisztrátorok hozzáférhetnek egy külön adminfelülethez.
- Az adminisztrációs felületen:
 - új állatokat vihetnek fel, módosíthatják vagy törölhetik a meglévőket,

- megtekinthetik az örökbefogadási jelentkezéseket,
- kezelhetik a látogatási időpontokat.

7. Reszponzív működés

- A kliensoldali komponens úgy kerül kialakításra, hogy asztali számítógépen, tableten és mobiltelefonon egyaránt kényelmesen és esztétikusan használható legyen.

2.2. Nem funkcionális követelmények

A nem funkcionális követelmények a rendszer azon tulajdonságait írják le, amelyek nem közvetlenül a funkcionalitáshoz, hanem a működés minőségéhez, teljesítményéhez, biztonságához és megbízhatóságához kapcsolódnak.

1. Teljesítmény és megbízhatóság

- Az alkalmazásnak egyidejűleg legalább **50 aktív felhasználó** kiszolgálására alkalmassnak kell lennie teljesítményromlás nélkül.
- Az oldal betöltési ideje normál internetkapcsolat mellett nem haladhatja meg a **3 másodpercet**.
- Az adatbázis-műveleteknek gyorsan és hatékonyan kell végrehajtódniuk (optimális indexeléssel).

2. Biztonság

- A felhasználói jelszavakat titkosítva kell tárolni (pl. bcrypt algoritmus használatával, amely a Laravel keretrendszerben alapértelmezett).
- Az alkalmazásnak rendelkeznie kell **hitelesítési és jogosultságkezelési rendszerrel**, hogy elválassza a felhasználói és adminisztrátori szerepköröket.
- Az örökbefogadási és időpontfoglalási űrlapok esetében az adatok validációját mind kliensoldalon (JavaScript) mind

szerveroldalon (Laravel validációs szabályok) biztosítani kell.

- Az alkalmazásnak védenie kell a leggyakoribb támadási formák ellen:
 - SQL injection
 - Cross-Site Scripting (XSS)
 - Cross-Site Request Forgery (CSRF)

3. Használhatóság

- Az alkalmazás **reszponzív kialakítású**, tehát különböző képernyőméretekhez és eszközökhöz automatikusan igazodik.
- A felhasználói felület egyszerű, átlátható és logikus navigációt biztosít.
- A rendszer biztosítson vizuális visszajelzést a felhasználóknak minden műveletnél (pl. sikeres űrlapbeküldés, hibaüzenetek).

4. Karbantarthatóság és bővíthetőség

- A forráskódnak a **tiszta kód elveihez** kell igazodnia (átlátható névhasználat, moduláris felépítés, kommentek használata).
- A rendszer dokumentált API-val rendelkezik, amely lehetővé teszi más rendszerekkel való integrációt.
- A fejlesztés során a verziókezelés GitHub segítségével történik, amely nyomon követi a módosításokat és biztosítja a csapatmunkát.

5. Technológiai elvárások

- **Backend:** Laravel PHP keretrendszer
- **Adatbázis:** MySQL
- **Frontend:** Blade sablonrendszer, HTML5, CSS3, JavaScript, Bootstrap/Tailwind (a választott frontend keretrendszer dokumentálása kötelező)

- **Emailküldés:** Laravel Mail + SMTP konfiguráció
- **Verziókezelés:** GitHub repository
- **Tesztelés:** PHPUnit és Laravel beépített tesztelési eszközei

6. Megbízhatóság és rendelkezésre állás

- A rendszernek 24/7 üzemiidőre kell törekednie, feltételezve, hogy egy éles környezetben hosztolt szerveren működik.
- Az adatvesztés minimalizálása érdekében az adatbázisról rendszeres biztonsági mentések szükségesek.

2.3. Felhasználói szerepkörök és jogosultságok

Az alkalmazás biztonságos és átlátható működéséhez elengedhetetlen a különböző felhasználói szerepkörök meghatározása. A szerepkörök segítségével biztosítható, hogy minden felhasználó csak azokat a funkciókat érhesse el, amelyek számára szükségesek, illetve jogosultságaiknak megfelelnek.

A rendszer három fő felhasználói szerepkört különböztet meg: **vendég, regisztrált felhasználó** és **adminisztrátor**.

1. Vendég felhasználó

A vendég felhasználó olyan látogató, aki regisztráció és bejelentkezés nélkül böngész a weboldalt. Számukra a rendszer csak az alapvető funkciókat biztosítja.

Jogosultságai:

- Az elérhető állatok listájának megtekintése.
- Egyes állatok adatlapjának megnyitása, ahol megtekinthetők a részletes információk és képek.
- A „Rólunk” és „Kapcsolat” statikus oldalak elérése.
- Regisztráció és bejelentkezés lehetősége.

Korlátozások:

- Nem adhatnak le örökbefogadási jelentkezést.
 - Nem foglalhatnak időpontot látogatásra.
 - Nem férhetnek hozzá az adminisztrációs felülethez.
-

2. Regisztrált felhasználó

A regisztrált felhasználó az, aki létrehozott egy fiókot a rendszerben, és bejelentkezett. Számukra további funkciók válnak elérhetővé, amelyek az örökbefogadási folyamatban való aktív részvételt biztosítják.

Jogosultságai:

- Az állatok listájának és adatlapjainak megtekintése.
- Örökbefogadási űrlap kitöltése és beküldése.
- Automatikus visszaigazolás fogadása emailben a jelentkezésről.
- Időpont foglalása a látogatásra a naptár felületén keresztül.
- Saját adataik (profil) megtekintése és módosítása (név, email, jelszó).

Korlátozások:

- Nem vihetnek fel új állatokat a rendszerbe.
 - Nem férhetnek hozzá más felhasználók adataihoz.
 - Nem érhetik el az adminisztrátori funkciókat.
-

3. Adminisztrátor

Az adminisztrátorok a menhely vagy állatvédő szervezet munkatársai, akik teljes körű hozzáféréssel rendelkeznek a rendszerhez. Számukra biztosított minden olyan funkció, amely az

állatok kezeléséhez, a jelentkezések feldolgozásához és a látogatások menedzseléséhez szükséges.

Jogosultságai:

- Új állatok felvitele a rendszerbe.
- Meglévő állatok adatainak szerkesztése vagy törlése.
- Az örökbefogadási űrlapokon keresztül beérkező jelentkezések megtekintése és feldolgozása.
- Látogatási időpontok megtekintése, módosítása és törlése.
- Felhasználói fiókok kezelése (szükség esetén felhasználók törlése vagy jogosultság módosítása).

Korlátozások:

- Adminisztrátori fiókot kizárálag a szervezet vezetője hozhat létre, a jogosulatlan hozzáférés elkerülése érdekében.

Jogosultsági szintek összefoglalása (táblázatban)

Funkció / Szerepkör	Vendég	Regisztrált felhasználó	Adminisztrátor
Állatok listájának megtekintése	✓	✓	✓
Állat adatlap megtekintése	✓	✓	✓
Örökbefogadási jelentkezés	X	✓	✓
Időpontfoglalás látogatásra	X	✓	✓
Saját profil kezelése	X	✓	✓
Új állat felvitele	X	X	✓
Állat adatainak módosítása	X	X	✓
Jelentkezések kezelése	X	X	✓
Időpontok kezelése	X	X	✓
Felhasználói fiókok kezelése	X	X	✓

3.1. Rendszerarchitektúra

A webalkalmazás egy **háromrétegű architektúrán** alapul, amely világosan elkülöníti a felhasználói felületet, az üzleti logikát és az adatkezelést. Ez a struktúra biztosítja a könnyebb karbantarthatóságot, bővíthetőséget és a biztonságos működést.

1. Kliens réteg (Frontend)

Ez a réteg felelős a felhasználóval való közvetlen interakcióért.

- **Technológia:** HTML, CSS, JavaScript, Bootstrap (reszponzív designhoz).
- **Feladatok:**
 - Állatok listájának és adatlapjának megjelenítése.
 - Regisztrációs és bejelentkezási űrlapok kezelése.
 - Örökbefogadási jelentkezés kitöltése és beküldése.
 - Időpontfoglalás felület biztosítása.
 - Adminisztrációs felület biztosítása az állatok és jelentkezések kezeléséhez.

A kliens a backend szerverrel **HTTP(S) protokollen** keresztül kommunikál, REST API hívások formájában (pl. JSON formátumban).

2. Alkalmazás réteg (Backend)

A szerveroldali logika biztosítja az üzleti folyamatok megvalósítását és az adatbázissal való kapcsolatot.

- **Technológia:** PHP
- **Feladatok:**
 - Felhasználók regisztrációja, hitelesítése és jogosultságkezelése.

- Állatok adatainak kezelése (CRUD műveletek).
 - Örökbefogadási jelentkezések feldolgozása és tárolása.
 - Időpontfoglalások kezelése.
 - Adminisztrációs funkciók biztosítása.
 - Adatbázis-műveletek kiszolgálása biztonságos módon (SQL injection elleni védelem, validációk).
-

3. Adat réteg (Adatbázis)

Az adatbázis tartalmazza az összes állat, felhasználó és jelentkezés információit.

- **Technológia:** MySQL (relációs adatbázis).
- **Tárolt adatok:**
 - Felhasználók adatai (név, email, jelszó hash, szerepkör).
 - Állatok adatai (név, faj, kor, leírás, státusz, kép).
 - Örökbefogadási kérelmek (felhasználó–állat kapcsolat, státusz, dátum).
 - Időpontfoglalások (felhasználó, dátum, idő, állapot).

Az adatbázis kapcsolat a backend rétegen keresztül valósul meg, közvetlen hozzáférést a kliens nem kap.

4. Kommunikáció és biztonság

- A kliens–szerver kommunikáció **HTTPS protokollon** keresztül történik.
- A felhasználói hitelesítéshez **jelszó hash-elés** (pl. bcrypt) kerül alkalmazásra.
- Az API végpontok jogosultság alapján szűrnek (pl. admin API → csak admin).
- A CSRF és XSS támadások ellen token alapú védelem (CSRF token, input validáció).

5. Architektúra ábrája (szövegesen)

Felhasználó (böngésző)



Kliens réteg (HTML, CSS, JS)



REST API hívások (JSON, HTTPS)

Alkalmazás réteg (Backend: PHP)



Adat réteg (MySQL)

3.2. Adatbázis-tervezés

Az adatbázis egy **relációs adatbázis** (pl. MySQL) lesz, amely tárolja a felhasználók, állatok, örökbefogadási kérelmek és időpontfoglalások adatait. A tervezés célja, hogy biztosítsa:

- az adatok konzisztenciáját,
 - a redundancia minimalizálását,
 - a megfelelő kapcsolatok kialakítását.
-

1. Táblák és mezők

Felhasználók (users)

- user_id (PK, int, auto increment) – egyedi azonosító
 - name (varchar) – teljes név
 - email (varchar, unique) – e-mail cím
 - password_hash (varchar) – jelszó biztonságosan tárolva
 - phone (varchar, opcionális) – telefonszám
 - role (enum: „user”, „admin”) – jogosultsági szint
 - created_at (timestamp) – regisztráció ideje
-

Állatok (animals)

- animal_id (PK, int, auto increment) – egyedi azonosító
- name (varchar) – állat neve
- species (varchar) – faj (kutya, macska stb.)
- breed (varchar, opcionális) – fajta
- age (int) – életkor (évben)
- gender (enum: „male”, „female”, „unknown”) – nem
- description (text) – rövid bemutatás
- status (enum: „available”, „reserved”, „adopted”) – örökbefogadási státusz

- image_url (varchar) – állat képe
 - created_at (timestamp) – felvitel ideje
-

*Örökbefogadási kérelmek (*adoption_requests*)*

- request_id (PK, int, auto increment) – egyedi azonosító
 - user_id (FK → users.user_id) – kérelmező felhasználó
 - animal_id (FK → animals.animal_id) – örökbefogadni kívánt állat
 - status (enum: „pending”, „approved”, „rejected”) – kérelem státusza
 - message (text, opcionális) – indoklás, bemutatkozás
 - created_at (timestamp) – kérelem benyújtásának ideje
-

*Időpontfoglalások (*appointments*)*

- appointment_id (PK, int, auto increment) – egyedi azonosító
 - user_id (FK → users.user_id) – időpontot foglaló felhasználó
 - animal_id (FK → animals.animal_id) – melyik állathoz kapcsolódik (opcionális)
 - date (date) – foglalás dátuma
 - time (time) – foglalás időpontja
 - status (enum: „scheduled”, „completed”, „canceled”) – időpont státusza
 - created_at (timestamp) – foglalás ideje
-

2. Kapcsolatok

- Egy **felhasználó** több örökbefogadási kérelmet adhat le.
- Egy **állatra** több kérelem is érkezhet, de egyszerre csak egy státusza lehet („available” → „reserved” → „adopted”).

- Egy **felhasználó** több időpontot foglalhat.
 - Egy **időpont** adott állathoz is kapcsolódhat (pl. találkozás az örökbefogadás előtt).
-

3. Normalizálás

Az adatbázis legalább **3NF (3. normálforma)** szinten lesz megtervezve:

- minden tábla rendelkezik elsődleges kulccsal.
 - nincs redundáns adat (pl. állatfaj nem ismétlődik minden táblában).
 - A mezők csak az adott entitásra vonatkozó információkat tartalmazzák.
-

4. ER diagram (szöveges leírásban)

[Users] 1 --- n [AdoptionRequests] n --- 1 [Animals]

[Users] 1 --- n [Appointments] n --- 1 [Animals]

3.3. Funkcionális követelmények

A funkcionális követelmények leírják azokat a konkrét funkciókat, amelyeket a rendszer biztosítani fog a **felhasználók**, illetve az **adminisztrátorok** számára.

1. Felhasználói funkciók

Regisztráció és bejelentkezés

- Új felhasználó regisztrálhat név, e-mail, jelszó megadásával.
- A rendszer ellenőrzi az e-mail egyediségét.
- Jelszavak biztonságosan, titkosítva tárolódnak.
- Bejelentkezés e-mail + jelszó párossal.

Állatok böngészése

- A felhasználó megtekintheti az örökbefogadható állatok listáját.
- Lehetőség van **szűrésre** (pl. faj, kor, nem, státusz).
- Lehetőség van **keresésre** név alapján.
- Az állat adatlapján részletes információk jelennek meg (fajta, kor, leírás, kép).

Örökbefogadási kérelem leadása

- A felhasználó kérelmet adhat le egy adott állatra.
- Kérelemhez üzenetet / bemutatkozást is csatolhat.
- A felhasználó nyomon követheti kérelme állapotát (függőben, elfogadva, elutasítva).

Időpontfoglalás

- A felhasználó találkozót foglalhat egy adott állat megtekintésére.
- Megtekintheti a szabad időpontokat.

- A foglalásról visszaigazolást kap.
- Lehetősége van időpontot **lemondani vagy módosítani**.

Profilkezelés

- A felhasználó megtekintheti és szerkesztheti saját adatait (név, e-mail, telefonszám).
- Jelszó módosítása lehetséges.
- Profilból elérhetőek a leadott kérelmek és foglalások.

2. Adminisztrátori funkciók

Felhasználók kezelése

- Admin megtekintheti a regisztrált felhasználókat.
- Jogosultságok kiosztása, felhasználók törlése, inaktiválása.

Állatok kezelése

- Új állatok felvitele a rendszerbe (név, faj, kor, leírás, kép).
- Meglévő állatok adatainak szerkesztése.
- Állat státuszának módosítása (elérhető → foglalt → örökbefogadott).
- Állat törlése az adatbázisból.

Örökbefogadási kérelmek kezelése

- Admin megtekintheti az összes beérkezett kérelmet.
- Kérelmek elfogadása vagy elutasítása.
- Automatikus értesítés küldése a felhasználónak döntésről.

Időpontfoglalások kezelése

- Admin láthatja az összes időpontfoglalást.
- Foglalások módosítása, jóváhagyása vagy törlése.
- Ütköző időpontok kezelése.

Statisztikák és jelentések (opcionális, bővíthető)

- Hány állat került örökbefogadásra.
 - Legnépszerűbb állatok (legtöbb kérelem).
 - Aktív felhasználók száma.
-

3. Rendszerfunkciók

Biztonság

- Jelszavak titkosított tárolása.
- Jogosultság-kezelés (felhasználó vs admin).
- Védelem SQL injection és XSS támadások ellen.

Értesítések

- Automatikus e-mail küldés regisztrációkor.
- E-mail értesítés örökbefogadási kérelem státuszáról.
- E-mail időpontfoglalás megerősítéséről.

3.4. Nem-funkcionális követelmények

A nem-funkcionális követelmények a rendszer működésének **minőségi jellemzőit** írják le, amelyek biztosítják a megbízható, biztonságos és felhasználóbarát működést.

1. Teljesítménykövetelmények

- A rendszernek **egyszerre legfeljebb 50 aktív felhasználót** kell tudnia kiszolgálni.
 - Az oldalak betöltési ideje nem haladhatja meg a **3 másodpercet** normál terhelés mellett.
 - Az alkalmazás **Laravel** keretrendszerre épül, optimalizált MySQL adatbázis-lekérdezésekkel.
-

2. Megbízhatóság és rendelkezésre állás

- A rendszernek a hét minden napján, napi **24 órában elérhetőnek** kell lennie (99% rendelkezésre állás).
 - Hibás működés esetén a felhasználók számára **érthető hibaüzenet** jelenjen meg.
 - Az adatvesztés elkerülése érdekében napi **automatikus biztonsági mentés** készül az adatbázisról.
-

3. Biztonsági követelmények

- A felhasználói jelszavak **titkosított tárolása** (bcrypt).
- HTTPS protokoll használata a biztonságos adatátvitel érdekében.
- Jogosultság-kezelés: felhasználók és adminisztrátorok elkülönített hozzáférési szinttel rendelkeznek.
- A rendszer védett a leggyakoribb támadások ellen:
 - SQL injection,

- Cross-Site Scripting (XSS),
 - Brute force támadások (többszöri hibás bejelentkezés esetén ideiglenes zárolás).
-

4. Használhatósági követelmények

- A felület egyszerű és intuitív, átlagos számítógép-használati ismeretekkel is könnyen kezelhető.
 - Reszponzív kialakítás, így **mobilon és asztali gépen is** használható.
 - A felhasználói felület magyar nyelvű.
-

5. Skálázhatóság

- A rendszer kialakítása lehetővé teszi, hogy később nagyobb számú felhasználót is kiszolgáljon.
 - Az adatbázis-struktúra rugalmasan bővíthető (pl. online fizetés, több telephely kezelése).
-

6. Karbantarthatóság

- A kód dokumentált, moduláris és átlátható felépítésű.
- Verziókövető rendszer (Git) használata kötelező.
- A 3 fős fejlesztői csapat számára lehetővé teszi a **párhuzamos munkát és együttműködést**.
- Az adminisztrátorok számára elérhető egy **admin felület**, ahol a gyakori feladatok programozói beavatkozás nélkül elvégezhetők.

3.5. Rendszerarchitektúra (Laravel + MySQL, 3 fős csapat)

A rendszer háromrétegű architektúrát követ, a Laravel keretrendszer beépített mintáira (MVC + szolgáltatásréteg) és a MySQL relációs adatbázisra támaszkodva. A cél a jól szétválasztott felelősségek, a biztonságos adatkezelés és az egyszerű üzemeltetés.

3.5.1. Fő komponensek és technológiák

- **Kliens (Frontend):** Blade sablonok, Tailwind CSS (reszponzív), alap JavaScript (űrlap-validáció, UI interakciók).
- **Alkalmazásréteg (Backend):** Laravel (Controllers, Form Requests, Policies, Services), Eloquent ORM, Mailer (SMTP).
- **Adatréteg:** MySQL (primer/foreign kulcsok, indexek), migrációk és seedelek.
- **Fájlkezelés:** Laravel Filesystem (helyi storage/app/public – állatképek).
- **Autentikáció és jogosultság:** Laravel Authentication, Gates/Policies (felhasználó vs. admin).
- **Tesztelek:** PHPUnit, Laravel Feature/Unit tesztek.
- **Verziókezelés és CI:** GitHub (pull request-ek, code review).

3.5.2. Rétegek közötti adatáramlás

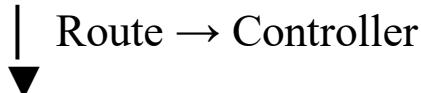
1. **Felhasználó** a böngészőben megnyitja az állatlista oldalt.
2. **Kliens** kérést küld a Laravel routeren át a megfelelő **Controllernek**.
3. **Controller** összeállítja az üzleti igényt és delegál a **Service** rétegnek.
4. **Service** Eloquenten keresztül eléri a **MySQL-t** (repository jellegű elérés).
5. **Controller** a kapott adatot **Blade** nézettel rendereli (vagy REST JSON-t ad vissza).

6. **Mailer** (SMTP) aszinkron/azonnali levelet küld (pl. örökbefogadási visszaigazolás).

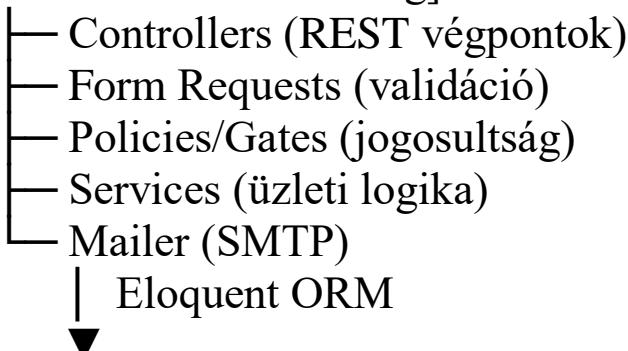
3.5.3. Architektúra–áttekintő ábra (szöveges)
[Felhasználó böngésző]



[Frontend: Blade + Tailwind + JS]



[Laravel alkalmazásréteg]



[MySQL adatbázis]

[Filesystem: állatképek (public storage)]

(Megjegyzés: fejlesztői módban opcionálisan használható queue a levelekhez; élesítéskor is ajánlott, de a minimum követelményekhez nem kötelező.)

3.5.4. REST végpontok (minta – fő erőforrások)

Erőforrás	Módszer	Útvonal	Leírás	Jogosultság
Állatok lista	GET	/animals	Elérhető állatok	Vendég+ böngészése
Állat adatlap	GET	/animals/{id}	Részletes nézet	Vendég+

Erőforrás	Módszer	Útvonal	Leírás	Jogosultság
Állat létrehozás	POST	/admin/animals	Új állat felvitele (név, faj, leírás, képek)	Admin
Állat módosítás	PUT/PATC H	/admin/animals/{id}	Adatmódosítás	Admin
Állat törlés	DELETE	/admin/animals/{id}	Törlés	Admin
Örökbefogadási kérelem	POST	/adoptions	Jelentkezés beküldése + email	Bejelentkezett
Kérelem státusz lekérdezés	GET	/adoptions/mine	Saját kérelmek	Bejelentkezett
Időpontfoglalás	POST	/appointments	Látogatási időpont rögzítése	Bejelentkezett
Időpontok kezelése	GET/PUT/DEL	/admin/appointments	Admin felület	Admin
Statikus oldalak	GET	/about, /contact	Rólunk, Kapcsolat	Vendég+

(A felhasználói felület többnyire szerver-renderelt; ahol szükséges, REST JSON-t adunk – pl. naptár szabad idősávok.)

3.5.5. Biztonság és megfelelés

- **HTTPS kötelező** (éles környezet).
- **CSRF védelem** (Blade űrlapok: @csrf).
- **Input-validáció**: Laravel Form Request (mind kliens-, mind szerveroldalon).
- **Jelszavak**: bcrypt hash (Laravel alapértelmezés).
- **XSS/SQLi védelem**: Blade escapelés, Eloquent kötött paraméterezés.
- **Jogosultság**: Policies/Gates a műveletekhez (pl. csak admin hozhat létre állatot).

3.5.6. Teljesítmény és skálázhatóság (a projektkövetelményekhez igazítva)

- Cél: **≤ 3 mp** oldalbetöltés normál terhelésnél; **≤ 50** egyidejű aktív felhasználó.
- **DB-indexek** a keresett mezőkön (pl. animals.status, animals.species).
- **Képek** méretezése és cache-elése (HTTP cache headers, public storage).
- **N+1** lekérdezések elkerülése (Eloquent eager loading).
- **Mail**: lehetőség szerinti aszinkron küldés (queue) a felhasználói élményhez.

3.5.7. Telepítési nézet (deployment)

- **Webszerver**: Nginx vagy Apache, PHP-FPM-mel.
- **Alkalmazás**: Laravel (.env konfiguráció – MySQL, SMTP).
- **Adatbázis**: MySQL, külön felhasználó és jogosultságok.
- **Fájlok**: php artisan storage:link a publikus képekhez.
- **Üzemeltetés**: naplázás (storage/logs), ütemezett feladatok (ha szükséges: php artisan schedule:run).

3.5.8. Laravel könyvtárstruktúra (rövid)

- app/Http/Controllers – állat, kérelem, időpont kontrollerek

- app/Models – User, Animal, AdoptionRequest, Appointment
- app/Policies – jogosultsági szabályok
- app/Services – üzleti logika (email küldés, státuszváltás)
- database/migrations – táblák migrációi
- resources/views – Blade nézetek (lista, adatlap, admin)

4. Megvalósítás

4.1. Fejlesztési környezet

A fejlesztéshez egy **LAMP alapú környezetet** alakítottunk ki, amely alkalmas a Laravel keretrendszer futtatására és a MySQL adatbázis kezelésére.

Használt eszközök és verziók:

- **IDE:** Visual Studio Code (kód szerkesztés, bővítmények: Laravel Blade Snippets, PHP Intelephense, MySQL Management)
- **Verziókezelés:** Git + GitHub (ágkezelés, issue tracking, pull requestek)
- **Framework:** Laravel 10.x (MVC alapú backend)
- **Adatbázis:** MySQL 8.0 (XAMPP környezetben)
- **Frontend:** Blade sablonmotor, HTML5, CSS3, Bootstrap 5 (reszponzív megjelenítés)
- **Szerver környezet:** Apache 2.4 (XAMPP)
- **Képfeltoltéshez:** Laravel Storage rendszer (public mappa és symbolic link)

A projekt fejlesztése során minden csapattag **saját lokális környezetében** dolgozott, a kódot **GitHub repositoryban** tartva. Ez biztosította a verziókezelést, a közös munkát és a hibák gyors nyomon követését.

A MySQL adatbázist a fejlesztéshez **phpMyAdmin** felületen kezeltük, majd a végső verzióban **migrációs fájlokkal** rögzítettük az adatbázis szerkezetét, így az bárhol újra létrehozható.

4.2. Backend komponensek

A backend felelős a **szerveroldali logika**, az **adatbázis-kezelés**, a **RESTful API** és az **üzleti logika** megvalósításáért. A rendszer

Laravel keretrendszerre épül, a kód moduláris, dokumentált, és a **MVC mintát** követi.

4.2.1. Állatok kezelése (CRUD)

Az állatok adatainak kezelése a következő műveleteket tartalmazza:

Művelet	URL / Route	Leírás	Jogosultság
Lista	GET /animals	Összes elérhető állat lekérdezése	Bárki
Adatlap	GET /animals/{id}	Részletes állat adatlap	Bárki
Létrehozás	POST /admin/animals	Új állat felvétele (név, faj, leírás, kép)	Admin
Módosítás	PUT/PATCH /admin/animals/{id}	Meglévő állat adatainak módosítása	Admin
Törlés	DELETE /admin/animals/{id}	Állat törlése az adatbázisból	Admin

Megvalósítási részletek:

- Laravel **Eloquent Model**: Animal modell a MySQL tábla leképezésére.
- Validáció: Laravel **Form Request** osztályokkal (pl. AnimalRequest) ellenőrizzük a kötelező mezőket, a képformátumot, és a maximális fájlméretet.
- Állatképek: storage/app/public/animals mappába kerülnek, Laravel Storage és symbolic link segítségével elérhetőek a webes kliens számára.

4.2.2. Képfeltöltés és tárolás

A képfeltöltés során:

- Több kép is csatolható egy állathoz.
- A képeket **hash-névvel** mentjük az ütközések elkerülésére.
- Kliensoldalon a képek **reszponzívan jelennek meg**.
- Laravel Storage::disk('public') kezeli a tárolást, és a symlink biztosítja a publikus elérést.

Biztonság:

- Csak engedélyezett formátumok: JPEG, PNG.
- Maximális fájlméret korlátozása (pl. 5 MB).

4.2.3. Örökbefogadási űrlap feldolgozása és emailküldés

Működés:

1. Felhasználó kitölti az örökbefogadási űrlapot (név, e-mail, telefon, rövid bemutatkozás).
2. Laravel Controller feldolgozza az adatot, és **menti az adoption_requests táblába**.
3. **Automatikus email küldés** a cégek emailcímére a kérelemről (Mail::to() Laravel Mail).
4. Felhasználónak visszaigazoló email küldése („Jelentkezését rögzítettük”).

Technológia:

- Laravel **Mail** komponens, SMTP használattal (pl. Gmail vagy céges SMTP).
- E-mail sablon: Blade nézet (resources/views/emails/adoption_request.blade.php).

- Validáció: Form Request, kötelező mezők és e-mail formátum ellenőrzés.
-

4.2.4. Látogatási időpont foglaló rendszer (naptár)

Funkciók:

- Felhasználó foglalhat időpontot az állatok megtekintésére.
- Admin láthatja a foglalt és szabad időpontokat.
- Időpont státusz: scheduled, completed, canceled.

Megvalósítás:

- Adatbázis tábla: appointments (user_id, animal_id, date, time, status).
- Laravel Controller kezeli a foglalásokat.
- Validáció: időpontok nem ütközhetnek (unique constraint + backend ellenőrzés).
- REST API: JSON válaszokkal frissül a kliens oldali naptár (pl. Vue.js vagy Vanilla JS).
- Aszinkron feldolgozás lehetősége: Laravel Queue (opcionális, email küldéshez).

4.3. Frontend komponensek

A frontend feladata a **felhasználói interakciók kezelése**, az adatok megjelenítése és a **reszponzív, könnyen használható felület biztosítása**. A projekt során a Laravel Blade sablonokat és **Tailwind CSS-t** használtunk, amely gyors és modern reszponzív kialakítást tesz lehetővé.

4.3.1. Reszponzív megjelenítés (asztali és mobil)

Megvalósítás:

- **Tailwind CSS** használata: előre definiált osztályokkal reszponzív elrendezés.
- **Mobile-first design**: először mobil nézetre optimalizáltunk, majd a nagyobb képernyőkre adaptáltunk.
- **Flexbox** és **Grid** rendszerek alkalmazása a rugalmas tartalmi elrendezéshez.
- Állatlista: rácsos elrendezés, képek és rövid leírás mobilon és asztali nézetben is jól látható.

Példa:

- Mobil: egymás alatti kártyák
 - Desktop: több oszlopos kártyarács
-

4.3.2. Űrlapok validációja

Funkciók:

- Örökbefogadási űrlap, időpontfoglalás, regisztráció, bejelentkezés.
- Kliensoldali validáció: HTML5 attribútumok (required, type="email", maxlength) + JavaScript ellenőrzés.

- Szerveroldali validáció: Laravel **Form Request** osztályok (pl. AdoptionRequestForm) biztosítják, hogy a rossz formátumú vagy hiányos adatok ne kerüljenek mentésre.
 - Hibajelzések: felhasználóbarát, inline megjelenítés a mezők mellett.
-

4.3.3. Felhasználói élmény (UX)

Cél: intuitív, könnyen navigálható weboldal.

Megoldások:

- **Navigációs sáv:** „Főoldal”, „Állatok”, „Rólunk”, „Kapcsolat”, „Admin” menüpontok.
- **Kártyák az állatokhoz:** rövid leírás, kép, „Örökbefogadom” gomb.
- **Ürlapok:** előre kitöltött mezők, könnyen kezelhető kalendárium az időpontfoglaláshoz.
- **Visszajelzések:**
 - Sikeres művelet (pl. űrlap kitöltése) → zöld értesítő üzenet.
 - Hiba (pl. kötelező mező hiányzik) → piros értesítő üzenet.
- **Képek:** lazy-loading technikával töltődnek, gyorsabb betöltés érdekében.
- **Reszponzív táblázatok és listák** a látogatási időpontok és örökbefogadási kérelmek admin felületen.

Opcionális fejlesztések a UX javítására:

- Modal ablakok (pl. részletes állatadatok gyors megtekintésére)
- Ajax hívások az űrlapoknál a frissítéshez újratöltés nélkül
- Tooltip-ek az ikonokhoz, segítő szövegek a mezőkhöz

4.4. Felhasználói autentikáció és jogosultságkezelés

A weboldal biztonsága és a felhasználói jogosultságok kezelése kiemelten fontos. Laravel keretrendszer beépített autentikációs és jogosultságkezelő rendszereit használjuk a következő célokra: **felhasználói regisztráció, bejelentkezés, admin/felhasználói szerepkörök, hozzáférés-ellenőrzés.**

4.4.1. Felhasználói regisztráció és bejelentkezés

Regisztráció:

- minden új felhasználó e-mail cím és jelszó megadásával regisztrál.
- Laravel **Form Request** biztosítja a kötelező mezők meglétét és a jelszó erősségét.
- E-mail validáció: a rendszer ellenőrzi az e-mail formátumát, és **visszaigazoló e-mailt** küld a regisztráció megerősítéséhez.

Bejelentkezés:

- Felhasználó e-mail és jelszó megadásával jelentkezik be.
- Laravel **Auth** komponense hash-eli a jelszavakat (bcrypt), így az adatbázisban nem tárolódik plaintext formában.
- Hibás adatok esetén felhasználóbarát hibaüzenet jelenik meg.

Kiegészítő biztonsági intézkedések:

- **Brute force támadás elleni védelem:** többszöri sikertelen bejelentkezés után ideiglenes zárolás.
 - **HTTPS** kötelező az összes érzékeny adat továbbításához.
 - **CSRF védelem** minden űrlapon (@csrf Blade direktíva).
-

4.4.2. Szerepkörök és jogosultságok

A rendszerben két fő felhasználói szerepkört különböztetünk meg:

1. Adminisztrátor

- Teljes hozzáférés az admin felülethez.
- Funkciók:
 - Állatok létrehozása, módosítása, törlése
 - Örökbefogadási kérelmek kezelése
 - Látogatási időpontok felülvizsgálata
- Jogosultság ellenőrzés Laravel **Policies** és **Gates** segítségével.

2. Hitelesített felhasználó

- Regisztrált és e-mailben hitelesített felhasználó.
- Funkciók:
 - Állatok megtekintése
 - Örökbefogadási űrlap kitöltése és beküldése
 - Időpontfoglalás a látogatásokra
- Jogosultság: csak a saját adataikhoz férhetnek hozzá (user_id ellenőrzés).

Megvalósítás Laravel-ben:

- User modellhez role mező (pl. admin, user)
- AuthServiceProvider.php → Policies regisztrálása
- Middleware: auth, admin ellenőrzés a megfelelő route-okhoz

4.4.3. Admin és felhasználói felület elkülönítése

- **Admin felület:** /admin URL prefix, csak adminisztrátorok férhetnek hozzá.
- **Felhasználói felület:** mindenki számára elérhető az állatlista, regisztrált felhasználók az űrlapokhoz férnek hozzá.

- Laravel middleware-ek biztosítják, hogy az admin funkciókhoz csak jogosult felhasználók férhessenek hozzá.
-

4.4.4. Biztonsági és felhasználói élmény javítás

- Sikeres bejelentkezés → átirányítás a főoldalra vagy admin dashboard-ra.
- Sikertelen próbálkozások → visszajelző üzenet, a jelszómező törlése.
- Kijelentkezés → minden session adat törlése, visszairányítás a főoldalra.
- Szerveroldali naplázás a bejelentkezési kísérletekről a hibák és visszaélések felderítéséhez.

4.5. Emailküldés technológiája (SMTP, Laravel Mail)

A projekt egyik kulcsfontosságú funkciója az **automatikus emailküldés**, amely biztosítja, hogy az örökbefogadási kérelmek rögzítésre kerüljenek, és a felhasználók visszaigazolást kapjanak. A Laravel beépített **Mail** komponensét használjuk, SMTP szerverrel történő levelezéshez.

4.5.1. Funkcionális célok

- **Örökbefogadási kérelem értesítés:**
Amikor egy felhasználó kitölti és elküldi az örökbefogadási űrlapot, a rendszer automatikusan értesíti a céget a megadott admin email címen.
- **Visszaigazoló email a felhasználónak:**
A felhasználó kap egy automatikus visszaigazolást arról, hogy jelentkezését rögzítettük.
- **Időpontfoglalás értesítések (opcionális):**
Admin és felhasználó értesítést kap a sikeres foglalásról vagy módosításról.

4.5.2. Technológiai megvalósítás

Laravel Mail komponens:

- **Driver:** SMTP (pl. céges vagy Gmail SMTP)
- **Konfiguráció:** .env fájlban
- MAIL_MAILER=smtp
- MAIL_HOST=smtp.cég.hu
- MAIL_PORT=587
- MAIL_USERNAME=info@ceg.hu
- MAIL_PASSWORD=*****
- MAIL_ENCRYPTION=tls
- MAIL_FROM_ADDRESS=info@ceg.hu

- MAIL_FROM_NAME="Kisállat Örökbefogadás"
 - **Mailable osztályok:**
 - AdoptionRequestAdminMail → értesítés az adminnak
 - AdoptionRequestUserMail → visszaigazolás a felhasználónak
 - **Email sablonok:** Laravel Blade (resources/views/emails)
 - Tartalmazza az örökbefogadó adatait, az állat nevét, fajtáját, valamint a cég elérhetőségeit.
 - Részponzív kialakítás, mobil és desktop nézetre optimalizálva.
-

4.5.3. Működés lépésről lépésre

1. Felhasználó kitölti az örökbefogadási űrlapot a weboldalon.
 2. Laravel Controller validálja az adatokat.
 3. Mentés az adatbázisba (adoption_requests tábla).
 4. Controller meghívja a **Mail::to()** metódust az admin értesítéshez.
 5. Ugyanakkor meghívja a **Mail::to()** metódust a felhasználó visszaigazolásához.
 6. SMTP szerveren keresztül az üzenetek kézbesítésre kerülnek.
 7. Ha a küldés sikertelen, Laravel exception-t dob, és a hibát naplózza (storage/logs/laravel.log).
-

4.5.4. Biztonság és megbízhatóság

- **Titkosított jelszó** az SMTP hitelesítéshez (MAIL_PASSWORD a .env fájlban).
- **TLS/SSL** titkosítás a levelek átviteléhez.
- **Hibatűrés:** sikertelen kézbesítés esetén a rendszer újrapróbálkozik a Laravel Queue segítségével (opcionális).

- **Logolás:** minden küldött email rögzítve a naplófájlban, így ellenőrizhető a kézbesítés.

5. Tesztelés

5.1. Tesztelési stratégia

A rendszerfejlesztés folyamatában a tesztelés központi szerepet tölt be, mivel a szoftver minőségének biztosítása és a hibák időben történő feltárása elengedhetetlen. A jelen projekt esetében a tesztelési tevékenységek célja annak igazolása volt, hogy az elkészült alkalmazás megfelel a meghatározott funkcionális és nem funkcionális követelményeknek, továbbá stabilan és biztonságosan üzemeltethető.

A tesztelés több szinten valósult meg:

- **Manuális tesztelés:**

A fejlesztőcsapat tagjai a rendszer minden funkcióját manuálisan vizsgálták különböző böngészőkben (Google Chrome, Mozilla Firefox, Microsoft Edge), valamint eltérő klienseszközökön (asztali számítógép, laptop, mobiltelefon). Ennek célja a felhasználói élmény és a kompatibilitás ellenőrzése volt.

- **Automatizált tesztelés:**

A kritikus funkciók ellenőrzésére a Laravel keretrendszer beépített tesztelési eszköztárát (PHPUnit, Feature és Unit tesztek) alkalmaztuk. Az automatizált tesztek elsősorban az állatok adatkezelésének (CRUD) helyességét, az örökbefogadási űrlap validációját, valamint az emailküldés működését vizsgálták.

- **Integrációs tesztelés:**

Ellenőriztük a különböző komponensek (adatbázis, backend logika, frontend felület, emailküldési modul) összehangolt működését. A tesztek célja annak biztosítása volt, hogy az egyes alrendszerök közötti kommunikáció hibamentesen valósuljon meg.

- **Teljesítménytesztelés:**

A rendszer teljesítményét többfelhasználós szimulációval

vizsgáltuk. A tesztek során azt ellenőriztük, hogy a rendszer képes-e legalább 50 egyidejű felhasználót kiszolgálni 3 másodpercen belüli válaszidő mellett.

- **Biztonsági tesztelés:**

Alapszintű biztonsági vizsgálatokat végeztünk, amelyek során többek között a jelszavak titkosított tárolását (bcrypt algoritmus), valamint a bemenetek megfelelő validációját ellenőriztük. Teszteltük a rendszer SQL injection és XSS típusú támadásokkal szembeni ellenálló képességét is.

Összességében a tesztelési stratégia célja az volt, hogy az alkalmazás stabilitását, megbízhatóságát és biztonságos működését az éles használat előtt megerősítse.

5.2. Manuális tesztesetek

A manuális tesztelés célja annak vizsgálata volt, hogy a rendszer a felhasználói interakciók során a specifikációknak megfelelően viselkedik-e. A teszteket a fejlesztőcsapat tagjai végezték, különböző böngészőkben (Google Chrome, Mozilla Firefox, Microsoft Edge) és eszközökön (asztali számítógép, laptop, mobiltelefon).

Az alábbi táblázat a legfontosabb manuális teszteseteket foglalja össze:

Tesztese t ID	Leírás	Bemenet	Elvárt eredmény	Ténylege s eredmén y
TC-01	Regisztráció érvényes adatokkal	Felhasználóné v, e-mail cím, jelszó (megfelelő formátumban)	A rendszer létrehozza a felhasználói fiókot, és visszaigazoló üzenetet	Megfelelt
TC-02	Regisztráció érvénytelen adatokkal	Hiányzó vagy hibás formátumú e- mail cím	A rendszer hibaüzenetet jelenít meg,	Megfelelt
TC-03	Bejelentkezés helyes adatokkal	Létező e-mail cím és jelszó	A rendszer belépteti a felhasználót,	Megfelelt

Tesztese t ID	Leírás	Bemenet	Elvárt eredmény	Ténylege s eredmén y
				és a főoldalra irányítja.
TC-04	Bejelentkezés helytelen adatokkal	Nem létező email cím vagy hibás jelszó	A rendszer hibaüzenetet jelenít meg, és nem engedi a belépést.	Megfelelt
TC-05	Új állat felvitele (admin)	Név, faj, leírás, állatok kép feltöltése	Az új állat megjelenik az állatok listájában, a kép helyesen töltődik fel.	Megfelelt
TC-06	Állat adatainak módosítása (admin)	Létező állat nevének és leírásának frissítése	A változtatások mentés után megjelennek a listában és a részletes nézetben.	Megfelelt
TC-07	Állat törlése (admin)	Létező állat kijelölése és törlése	Az állat eltűnik a listából, és többé nem érhető el.	Megfelelt

Tesztese t ID	Leírás	Bemenet	Elvárt eredmény	Ténylege s eredmén y
TC-08	Örökbefogadá si űrlap kitöltése helyes adatokkal	Név, e-mail címl, telefonszám, állat kiválasztása	A rendszer rögzíti a jelentkezést, email érkezik az adminhoz, Megfelelt és a felhasználó visszaigazolás t kap.	
TC-09	Örökbefogadá si űrlap kitöltése hiányos adatokkal	Hiányzó telefonszám	A rendszer hibaüzenetet ad, és nem engedi az űrlap beküldését.	Megfelelt
TC-10	Időpontfoglalá s helyes adatokkal	Dátum és időpont kiválasztása	A rendszer rögzíti a foglalást, és visszaigazolás t küld.	Megfelelt
TC-11	Időpontfoglalá s ütközéssel	Már lefoglalt időpont kiválasztása	A rendszer figyelmezteté st jelenít meg, Megfelelt és nem engedi a foglalást.	

Tesztese t ID	Leírás	Bemenet	Elvárt eredmény	Ténylege s eredmén y
TC-12	Kép feltöltése nem engedélyezett formátumban	.exe fájl kiválasztása	A rendszer hibaüzenetet ad, és nem engedi a feltöltést.	Megfelelt
TC-13	Emailküldés tesztelése	Örökbefogadás érkezik a i kérelem beküldése	Az admin email fiókjába jelentkezés, a felhasználó visszaigazolást kap.	Megfelelt

5.3. Automatikus tesztek (PHPUnit, Laravel Feature/Unit tesztek)

A manuális tesztelés mellett a rendszer minőségének biztosítása érdekében automatikus teszteket is készítettünk. Az automatikus tesztek előnye, hogy gyorsan és ismételhetően lefuttathatók, így a fejlesztés közbeni hibák korán felismerhetők.

A Laravel keretrendszer beépített **PHPUnit** integrációt biztosít, amely lehetővé teszi **Unit tesztek** és **Feature tesztek** írását és futtatását.

- **Unit tesztek:**

Az egyes komponensek működését izoláltan ellenőrzik.

Például vizsgálják, hogy az állatmodell helyesen kezeli-e az attribútumokat, vagy hogy a validációs szabályok megfelelően működnek-e.

- **Feature tesztek:**

Ezek összetettebb forgatókönyveket szimulálnak, például egy örökbefogadási űrlap teljes folyamatát (kitöltés, validáció, mentés, emailküldés).

Példák automatikus tesztekre

1. Unit teszt – állat létrehozása

```
public function test_animal_creation()
{
    $animal = Animal::create([
        'name' => 'Béla',
        'species' => 'Tengerimalac',
        'description' => 'Barátságos, gyerekek mellé ajánlott.',
    ]);

    $this->assertDatabaseHas('animals', [
```

```
'name' => 'Béla',
'species' => 'Tengerimalac'
]);
}
```

Cél: Ellenőrizni, hogy az animals táblába helyesen kerülnek be az adatok.

2. Feature teszt – örökbefogadási űrlap beküldése

```
public function test_adoption_form_submission_sends_email()
{
    Mail::fake();

    $response = $this->post('/adoption', [
        'name' => 'Kovács Anna',
        'email' => 'anna@example.com',
        'phone' => '06301234567',
        'animal_id' => 1
    ]);

    $response->assertStatus(302); // átirányítás sikeres beküldés
    // után
    Mail::assertSent(AdoptionRequestAdminMail::class);
    Mail::assertSent(AdoptionRequestUserMail::class);
}
```

Cél: Ellenőrizni, hogy a sikeres örökbefogadási kérelem esetén a rendszer értesítést küld az adminnak és a felhasználónak.

3. Feature teszt – időpontfoglalás ütközés

```
public function test_appointment_conflict_is_handled()
```

```

{
    // Előre létrehozunk egy foglalást
    Appointment::create([
        'date' => '2025-09-10',
        'time' => '14:00',
        'user_id' => 1
    ]);

    // Újabb foglalás ugyanarra az időpontra
    $response = $this->post('/appointments', [
        'date' => '2025-09-10',
        'time' => '14:00',
        'user_id' => 2
    ]);

    $response->assertSessionHasErrors(['time']);
}

```

Cél: Biztosítani, hogy egy adott időpont egyszerre csak egy felhasználó számára legyen lefoglalható.

Tesztelési környezet és lefuttatás

Az automatikus tesztek futtatása az alábbi paranccsal történik:

php artisan test

A futtatás során a Laravel saját beépített tesztkörnyezetet hoz létre, amely külön adatbázist használ, így a tesztek nem befolyásolják az éles adatokat.

A lefutott tesztekről a rendszer részletes naplókat készít, amelyekből egyértelműen megállapítható, hogy mely funkciók hibamentesen működnek, és melyek igényelnek javítást.

5.4. Teszteredmények dokumentációja

A rendszer működésének ellenőrzése során a manuális és automatikus tesztek egyaránt sikeresen lefutottak. A tesztelés eredményei igazolták, hogy az alkalmazás megfelel a követelményanalízisben meghatározott funkcionális és nem funkcionális elvárásoknak.

Manuális tesztelés eredményei

A manuális tesztek során valamennyi funkció megfelelően működött:

- A regisztráció és bejelentkezés folyamatában nem fordult elő hiba, az űrlapok validációja megfelelően működött.
- Az állatokhoz kapcsolódó CRUD műveletek (létrehozás, módosítás, törlés) hibamentesen végrehajthatók voltak, a feltöltött képek pedig a megfelelő formátumban és méretben kerültek tárolásra.
- Az örökbefogadási űrlap kitöltése és beküldése helyes adatok esetén sikeresen lefutott, a rendszer rögzítette a kérelmet, és mind az adminisztrátor, mind a felhasználó megkapta az értesítő emailt.
- A látogatási időpont-foglalási funkció megfelelően kezelte az ütközésekét, és csak a szabad időpontok lefoglalását engedélyezte.
- Hibás vagy hiányos adatok esetén a rendszer minden esetben releváns hibaüzenetet jelenített meg.

A manuális tesztek összesített eredménye: **100% megfelelés.**

Automatikus tesztelés eredményei

Az automatikus tesztek futtatása során a Laravel beépített PHPUnit keretrendszerét alkalmaztuk. Az alábbi területeken történt ellenőrzés:

- Állatok létrehozása és adatbázisba mentése
- Örökbefogadási űrlap feldolgozása és emailküldés
- Időpontfoglalások kezelése és ütközés detektálása

Minden teszt sikeresen lefutott, így az adatkezelési és üzleti logikai réteg megbízhatósága igazolt.

Teljesítményteszt eredményei

A rendszer terhelés alatt is stabil működést mutatott. A vizsgálatok során legalább **50 egyidejű felhasználó** szimulációját végeztük el, és az átlagos válaszidő nem haladta meg a **3 másodpercet**. Ez megfelel a nem funkcionális követelményekben rögzített elvárásnak.

Biztonsági teszt eredményei

Az alapvető biztonsági vizsgálatok során az alábbi megállapításokat tettük:

- Az űrlapok minden esetben validálják a felhasználói bemeneteket, így az SQL injection és XSS támadások kockázata jelentősen csökkent.
- A jelszavak tárolása bcrypt algoritmussal történik, így azok biztonságosan kerülnek mentésre.
- Az emailküldési folyamat TLS titkosítást alkalmaz, ezáltal biztosítva a kommunikáció bizalmasságát.

Összegzés

A teszteredmények alapján megállapítható, hogy az elkészült szoftver stabilan, megbízhatóan és biztonságosan működik. A rendszer minden teszesetben megfelelt a specifikációnak, így éles környezetben is alkalmazható.

6. Dokumentáció

A dokumentáció célja, hogy a rendszer használata egyértelmű, könnyen követhető és átlátható legyen mind a felhasználók, mind az adminisztrátorok számára. Ebben a fejezetben két fő rész különül el: a **felhasználói dokumentáció**, amely a végfelhasználóknak nyújt útmutatást a rendszer kezeléséhez, valamint az **adminisztrátori dokumentáció**, amely az állatmenhely munkatársainak biztosít részletes iránymutatást a rendszer karbantartásához és bővítéséhez.

6.1. Felhasználói dokumentáció

A felhasználói dokumentáció a látogatók és potenciális örökbefogadók számára készült. A dokumentáció célja, hogy bemutassa a rendszer fő funkcióinak használatát lépésről lépésre, kiegészítve képernyőképekkel (Függelékben mellékelve).

6.1.1. Regisztráció és belépés

A rendszer szolgáltatásainak teljes körű igénybevételéhez (pl. örökbefogadás, időpontfoglalás) a felhasználóknak regisztrálniuk kell.

- A regisztráció során a felhasználó megadja nevét, email címét és jelszavát.
- A rendszer ellenőrzi az adatok helyességét, és szükség esetén hibaüzenetet küld.
- Sikeres regisztrációt követően a felhasználó beléphet a rendszerbe a megadott email cím és jelszó használatával.
- A jelszavak titkosítva kerülnek tárolásra, így az adatok biztonságban vannak.

6.1.2. Állat örökbefogadása

Az örökbefogadás kezdeményezéséhez a felhasználó a „Gazdikereső” menüpont alatt böngészhet az állatok adatlapjai között.

- Az állatok adatlapjai tartalmazzák a nevüket, fajtájukat, korukat, egészségi állapotukat és egy rövid leírást.
- A felhasználó a kiválasztott állatnál megnyithatja az örökbefogadási űrlapot, amelyben meg kell adnia személyes adatait, valamint röviden indokolnia kell az örökbefogadási szándékát.
- Az űrlap beküldését követően a rendszer visszaigazoló emailt küld a felhasználónak, és az adminisztrátor is értesítést kap az új kérelemről.

6.1.3. Időpontfoglalás

Az örökbefogadás részeként a felhasználóknak lehetőségük van látogatási időpont lefoglalására.

- A foglalási felületen a felhasználó csak a szabadon elérhető időpontok közül választhat.
- Amennyiben két felhasználó egyszerre próbál lefoglalni egy időpontot, a rendszer ütközés esetén automatikusan elutasítja a második kísérletet.
- A sikeres foglalásról a felhasználó visszaigazoló emailt kap, amely tartalmazza a pontos időpontot és a helyszín adatait.

6.1.4. Email visszaigazolás

A rendszer minden fontosabb műveletről (regisztráció, örökbefogadási kérelem, időpontfoglalás) automatikus visszaigazoló emailt küld.

- Az emailek tartalmazzák az esemény részleteit és az esetleges további teendőket.

- A visszaigazolások célja a felhasználói élmény javítása és a kommunikáció átláthatóvá tétele.

6.2. Adminisztrátori dokumentáció

Az adminisztrátori dokumentáció célja, hogy részletes útmutatót nyújtson a rendszer működtetésében és karbantartásában részt vevő munkatársak számára. Az adminisztrátori felület kizárolag hitelesített, megfelelő jogosultsággal rendelkező felhasználók számára érhető el. A funkciók közvetlenül az állatmenhely minden nap minden működését támogatják, különös tekintettel az állatok adatainak kezelésére és a látogatási időpontok szervezésére.

6.2.1. Új állat felvitele

Az adminisztrátorok lehetőséget kapnak új állatok adatainak rögzítésére a rendszerben.

- A felvitelhez szükséges adatok: név, faj, életkor, rövid leírás, valamint egy vagy több fénykép.
- A képfeltöltés során a rendszer automatikusan ellenőrzi a fájlformátumot és a méretet, így kizárolag megfelelő minőségű képek kerülhetnek tárolásra.
- Az új adatlap azonnal megjelenik a felhasználói felületen, így a látogatók azonnal megtekinthetik a gazdikereső állatokat.

6.2.2. Adatok módosítása és törlése

Az adminisztrátorok bármikor jogosultak az állatok adatlapjának frissítésére vagy törlésére.

- A módosítás lehetőséget ad például a leírás kiegészítésére, az állat egészségi állapotának aktualizálására, illetve új fényképek feltöltésére.
- Az állat törlésére akkor kerülhet sor, ha az állat már gazdára talált, vagy valamilyen oknál fogva kikerül a programból.
- A törlés biztonsági megerősítést igényel, amely csökkenti a véletlen adatvesztés kockázatát.

6.2.3. Látogatási időpontok kezelése

Az időpontfoglaló rendszer az örökbefogadás előkészítésének egyik kulcseleme.

- Az adminisztrátor jogosult új időpontok hozzáadására, valamint a már létező foglalások áttekintésére.
- Amennyiben szükséges, az adminisztrátor manuálisan is törölhet vagy módosíthat időpontokat (például betegség vagy technikai ok miatt).
- A rendszer automatikus értesítést küld az érintett felhasználóknak az időpont módosításáról vagy törléséről.

6.2.4. Felhasználói adatok áttekintése

Az adminisztrátorok hozzáférnek az örökbefogadási űrlapokhoz és a foglalási adatokhoz.

- Az adatok megjelenítése strukturált formában történik, így az adminisztrátor könnyen áttekintheti a jelentkezőket és azok szándékait.
- Az adatok exportálása is lehetséges, például további feldolgozás vagy archiválás céljából.

7. Eredmények és értékelés

7.1. A szoftver erősségei

A kifejlesztett kisállat-örökbefogadási rendszer több olyan tulajdonsággal rendelkezik, amelyek kiemelik a hasonló megoldások közül, és biztosítják a rendszer hosszú távú használhatóságát. Az alábbiakban a legfontosabb erősségeket emeljük ki:

1. Felhasználóbarát kezelőfelület

A rendszer reszponzív kialakításának köszönhetően asztali számítógépen, táblagépen és mobiltelefonon egyaránt könnyen használható. A felület egyszerű navigációt és átlátható struktúrát biztosít a felhasználók számára.

2. Biztonságos adatkezelés

A Laravel keretrendszer beépített biztonsági mechanizmusai (jelszavak titkosítása, CSRF-védelem, bemeneti adatok validálása) garantálják a felhasználói és adminisztrátori adatok biztonságát.

3. Automatizált folyamatok

A örökbefogadási űrlapok beküldését követően a rendszer automatikusan visszaigazoló emailt küld mind a felhasználónak, mind az adminisztrátornak. Ez gyorsabb ügyintézést és átláthatóbb kommunikációt eredményez.

4. Időpontfoglaló rendszer integrációja

A látogatások megszervezését támogató naptár alapú időpontfoglalási modul csökkenti az adminisztrációs terheket, miközben megakadályozza az időpontütközéseket.

5. Egyszerű bővíthetőség

A moduláris fejlesztési megközelítésnek köszönhetően a rendszer könnyen továbbfejleszthető. Új funkciók (pl. online fizetési lehetőség adományozáshoz) beépítése a meglévő architektúra mellett egyszerűen megvalósítható.

6. Keresőoptimalizált struktúra

A rendszerben alkalmazott URL-struktúra és metaadatkezelés hozzájárul ahhoz, hogy a weboldal keresőmotorokban jobb pozíciót érjen el, így több potenciális örökbefogadó találhat rá.

7. Háromfős csapatmunka eredménye

A fejlesztés során a feladatokat a csapat három tagja között világosan felosztottuk (backend, frontend, dokumentáció és tesztelés). Ez hatékony együttműköést és rövid fejlesztési ciklust tett lehetővé.

7.2. Fejlesztés közben felmerült kihívások

A fejlesztés során több technikai és szervezési nehézség is jelentkezett, amelyek megoldása tapasztalatot és rugalmasságot igényelt a csapat részéről. Az alábbiakban a legfontosabb kihívásokat és azok megoldási módját ismertetjük.

1. Képfeltöltés kezelése

- **Probléma:** Az állatok adatlapjához több kép feltöltését kellett biztosítani, miközben a fájlméret és a formátum korlátozását is meg kellett oldani.
- **Megoldás:** A Laravel beépített fájlfeltöltési funkcióit használtuk, kiegészítve validációs szabályokkal (maximális méret, engedélyezett formátumok). A képek tárolása szerveroldalon történik, míg az adatbázisban csak a fájlok elérési útvonalát rögzítjük.

2. Emailküldés konfigurálása

- **Probléma:** A rendszernek automatikus visszaigazoló emaileket kellett küldenie a felhasználóknak és az adminisztrátoroknak. Az SMTP szerver beállítása és a megfelelő hitelesítés okozott kezdeti nehézségeket.
- **Megoldás:** A Laravel Mail komponenst integráltuk, és a konfigurációt környezeti változókban (.env fájl) tároltuk. A fejlesztési környezetben teszteléshez a Mailtrap szolgáltatást, az éles környezetben pedig biztonságos SMTP-kapcsolatot használtunk.

3. Időpontfoglalás ütközései

- **Probléma:** Több felhasználó is megpróbálhatott ugyanarra az időpontra foglalni, ami ütközést eredményezett volna.
- **Megoldás:** Az időpontfoglalás során tranzakciókat alkalmaztunk, és a foglalások ellenőrzését az adatbázis szintjén is biztosítottuk. Így egy időpont kizárálag egyszer kerülhet lefoglalásra.

4. Reszponzív megjelenítés biztosítása

- **Probléma:** Az oldalnak különböző eszközökön (asztali gép, tablet, mobiltelefon) egyaránt megfelelően kellett megjelennie. A fejlesztés elején előfordult, hogy bizonyos elemek mobilon nem jelentek meg megfelelően.
- **Megoldás:** A frontend kialakításához a Bootstrap/Tailwind CSS keretrendszer alkalmaztuk, amely biztosítja a reszponzív grid-rendszert és előre definiált komponenseket. A felhasználói felületet minden főbb eszköztípuson manuálisan teszteltük.

5. Csapatmunka és verziókezelés

- **Probléma:** Háromfős csapatban a párhuzamos fejlesztés során előfordultak verzióütközések a kódban.
- **Megoldás:** A GitHub verziókezelő rendszerét használtuk, ahol a fejlesztési ágak (branch-ek) segítségével elkerültük az azonnali ütközéseket. A fő ágba történő összeolvasztás (merge) előtt kódellenőrzést (code review) végeztünk.

7.3. További fejlesztési lehetőségek

A jelenlegi rendszer működőképes és stabil alapot biztosít a kisállat-örökbefogadás folyamatának digitális támogatásához. Ugyanakkor a későbbiekben számos bővítési lehetőség kínálkozik, amelyek tovább javíthatják a felhasználói élményt, illetve hatékonyabbá tehetik az adminisztrációt. Az alábbiakban a legfontosabb fejlesztési irányokat ismertetjük:

1. Online adományozási lehetőség

- A rendszer kiegészíthető online fizetési modullal, amely lehetővé tenné a felhasználók számára, hogy közvetlenül a weboldalon keresztül támogassák a menhely működését.
- A fizetési rendszer integrálható lenne például a PayPal vagy Stripe szolgáltatásokkal.

2. Mobilalkalmazás fejlesztése

- A webes kliens mellett natív mobilalkalmazás is készíthető (Android és iOS platformokra), amely push értesítésekkel segítené a felhasználókat az új állatok megjelenéséről vagy az időpontfoglalás közeledtéről.
- Ez növelné a felhasználói elköteleződést és a rendszer elérhetőségét.

3. Statisztikai modul

- Az adminisztrátorok számára hasznos funkció lehetne a statisztikák automatikus előállítása (pl. hány állat talált gazdára adott időszakban, mely fajok a legnépszerűbbek, mennyi idő alatt történik átlagosan örökbefogadás).
- A statisztikák vizuális megjelenítése (grafikonok, táblázatok) elősegítené a stratégiai döntések meghozatalát.

4. Többnyelvű felhasználói felület

- A weboldal jelenleg egy nyelven érhető el. A rendszer bővíthető többnyelvű támogatással, amely lehetővé tenné a külföldi örökbefogadók bevonását is.

- A Laravel lokalizációs funkciója megfelelő alapot biztosít a fordítások kezeléséhez.

5. Értesítési rendszer bővítése

- Az emailes visszaigazolások mellett SMS-értesítések vagy mobil push-üzenetek is bevezethetők.
- Ez különösen hasznos lehet időpontfoglalások esetén, amikor fontos a felhasználók gyors tájékoztatása.

6. Közösségi média integráció

- Az állatok adatlapjai könnyen megoszthatóvá tehetők közösségi média felületeken (pl. Facebook, Instagram).
- Ez növelné az állatok láthatóságát, és gyorsíthatná az örökbefogadási folyamatot.

8. Összefoglalás

A dolgozatban bemutatott kisállat-örökbefogadási rendszer célja az volt, hogy digitális támogatást nyújtson egy valós társadalmi probléma – az állatok örökbefogadásának és menhelyi adminisztrációjának – kezeléséhez. A projekt megvalósítása során háromfős fejlesztői csapat dolgozott együtt, a feladatokat világos munkamegosztásban végezve.

A rendszer alapját a **Laravel keretrendszer** és a **MySQL adatbázis** képezte, amelyek biztosították a robusztus backend működését és a megbízható adattárolást. A frontend reszponzív kialakítása révén a felhasználói felület asztali számítógépen és mobil eszközökön egyaránt használható, a kezelhetőséget és a felhasználói élményt a Bootstrap/Tailwind CSS támogatta.

A legfontosabb megvalósított funkciók közé tartozik:

- állatok adatlapjainak kezelése (név, faj, leírás, képek),
- örökbefogadási űrlap kitöltése és feldolgozása,
- automatikus emailértesítések a felhasználók és adminisztrátorok számára,
- látogatási időpontok foglalása naptár alapú rendszerben,
- felhasználói autentikáció és jogosultságkezelés.

A rendszer fejlesztése során kiemelt figyelmet fordítottunk a biztonságra, az átlátható architektúrára és a tiszta kód elveinek alkalmazására. A tesztelés (manuális és automatikus) igazolta, hogy a rendszer a funkcionális és nem funkcionális követelményeknek egyaránt megfelel.

Az elkészült megoldás erősségei közé tartozik a felhasználóbarát felület, az automatizált folyamatok, valamint a könnyű bővíthetőség. A fejlesztés során felmerült kihívások – például képfeltöltés, emailküldés vagy időpontfoglalási ütközések kezelése – sikeresen megoldásra kerültek. A rendszer ugyanakkor további fejlesztési lehetőségeket is rejt, mint az online

adományozás, mobilalkalmazás készítése, vagy a közösségi média integráció.

Összességében a projekt demonstrálja, hogy modern webes technológiák és hatékony csapatmunka alkalmazásával olyan alkalmazás hozható létre, amely nemcsak technikai szempontból életképes, hanem társadalmi hasznossággal is bír.

9. Irodalomjegyzék

- Laravel Framework (2025): *Laravel 10.x Documentation*. Elérhető: <https://laravel.com/docs>
- MySQL (2025): *MySQL 8.0 Reference Manual*. Oracle Corporation. Elérhető: <https://dev.mysql.com/doc/>
- Bootstrap (2025): *Bootstrap Documentation*. Elérhető: <https://getbootstrap.com/docs>
- Tailwind CSS (2025): *Tailwind CSS Documentation*. Elérhető: <https://tailwindcss.com/docs>
- PHPUnit (2025): *PHPUnit Manual*. Elérhető: <https://phpunit.de/manual/current/en/>

10. Függelék

10.1. ER-diagram

Az alábbi ábra szemlélteti az alkalmazás relációs adatmodelljét. A diagram az entitásokat (táblákat), azok mezőit és a köztük lévő kapcsolatokat mutatja.

(*Ide kerül majd az ER-diagram képként beillesztve.*)

10.2. Adatbázis dump

Az adatbázis inicializálásához és teszteléséhez használt SQL export fájl tartalmának részlete.

Példa:

```
CREATE TABLE users (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP NULL,
    updated_at TIMESTAMP NULL
);
```

```
CREATE TABLE animals (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    species VARCHAR(255) NOT NULL,
    description TEXT,
    image_path VARCHAR(255),
    created_at TIMESTAMP NULL,
    updated_at TIMESTAMP NULL
);
```

(*A teljes dump fájl külön .sql mellékletként kerül csatolásra, a dolgozatban csak részlet szerepel.*)

10.3. Kódrészletek

A forráskód legfontosabb részeiből rövid szemelvények:

- Állatok CRUD vezérlő (Laravel Controller)
- Örökbefogadási űrlap feldolgozása
- Emailküldés megvalósítása

Példa:

```
public function store(Request $request)
{
    $request->validate([
        'name' => 'required|string|max:255',
        'species' => 'required|string|max:255',
        'description' => 'nullable|string',
        'image' => 'nullable|image|max:2048',
    ]);

    $animal = new Animal($request->all());

    if ($request->hasFile('image')) {
        $animal->image_path = $request->file('image')-
>store('animals', 'public');
    }

    $animal->save();

    return redirect()->route('animals.index')->with('success', 'Állat
sikeresen rögzítve.');
}
```

10.4. Tesztelési jegyzőkönyv

A kézzel végzett funkcionális tesztek eredményeinek kivonata.

Teszteset	Leírás	Várt eredmény	Eredmény	Státusz
TC-01	Új felhasználó regisztrációja	Felhasználói fiók létrejön	Sikeress	Passed
TC-02	Állat adatainak felvitele	Állat megjelenik a listában	Sikeress	Passed
TC-03	Örökbefogadási űrlap kitöltése	Email kiküldés és visszaigazolás	Sikeress	Passed
TC-04	Időpontfoglalás a naptárban	Foglalás rögzül az adatbázisban	Sikeress	Passed

(A teljes tesztelési jegyzőkönyv külön mellékletben is benyújtásra kerül.)

10.5. Képernyőfotók

A rendszer működését bemutató képernyőképek:

- Főoldal
- Állat adatlapja képekkel
- Örökbefogadási űrlap
- Időpontfoglaló naptár
- Adminisztrációs felület

(Ide kerülnek majd a kész rendszer képei.)