

BZSH Külkereskedelmi Technikum



Mázli Állatmenhely

Témavezető: Józsa Béla

Készítette: Király Péter, Németh Angéla

Szoftverfejlesztő és -tesztelő

BUDAPEST 2024

Tartalom

| | |
|---|----|
| A projekt célja | 4 |
| A projekt beüzemelése | 4 |
| Adatbázis | 4 |
| Az adatbázis táblái és mezői | 5 |
| Admin tábla | 5 |
| Animal tábla | 5 |
| Enquery tábla | 6 |
| Kind tábla | 6 |
| Fejlesztői dokumentáció | 6 |
| Backend | 6 |
| Osztályok | 8 |
| Entity Models | 8 |
| RequestModel | 8 |
| ResponseModel | 8 |
| AdminController | 9 |
| AnimalController | 9 |
| EnqueryController | 10 |
| KindController | 10 |
| Helpers | 10 |
| HashHelper | 10 |
| JWTHelper | 11 |
| RemoveAuthorizeFilterOperationFilter | 11 |
| Frontend | 11 |
| Készítéshez használt alkalmazások, keretrendszerek. | 11 |
| A frontend főbb egységei | 12 |
| Az egyes egységek feladatai | 13 |
| Komponensek csoport | 13 |
| Modellek csoport | 17 |
| Szolgáltatások csoport | 18 |
| Könyvtárszerkezet | 19 |
| Teszt dokumentáció | 21 |
| ControllersUnitTests: | 21 |
| AdminControllerTests | 21 |
| AnimalControllerTests | 22 |
| EnqueryControllerTests | 22 |
| KindControllerTests | 23 |
| Front end teszt | 24 |
| Felhasználói dokumentáció | 24 |
| Főoldal | 25 |
| Menüpontok | 25 |
| Kutyák menüpont | 25 |
| Érdeklődés rögzítése | 26 |
| Macskák menüpont | 27 |
| Egyéb menüpont | 27 |
| Admin funkciók | 27 |

| | |
|------------------------|----|
| Bejelentkezés | 28 |
| Új állat felvétele | 28 |
| Érdeklődések listázása | 29 |
| Állat módosítása | 29 |
| Állat törlése | 30 |
| Összefoglalás | 30 |
| További célkitűzések | 31 |

A projekt célja

A projekt célja egy olyan webes alkalmazás létrehozása, amelyben egy állatmenhely állatait lehet megnézni és örökbe fogadni. A cél valós igényeken alapul, mert egyre több az elárvult, gazdátlanodott kisállat, aki szerető otthonra vágyik, és szerencsére egyre többen vannak azok, akik szívesen fogadnának örökbe kutyát, macskát, nyulat, ill. egyéb kis kedvencet.

A projekt beüzemelése

A projekt két fő komponense saját könyvtárban van: „backend” illetve „frontend”, és külön-külön el kell indítani:

Backend indítása

A „backend” könyvtárban terminál ablakban ki kell adni a **'dotnet watch'** parancsot. A sikeres indítás után egy swagger felület nyílik meg az alapértelmezett böngészőben.

Frontend indítása

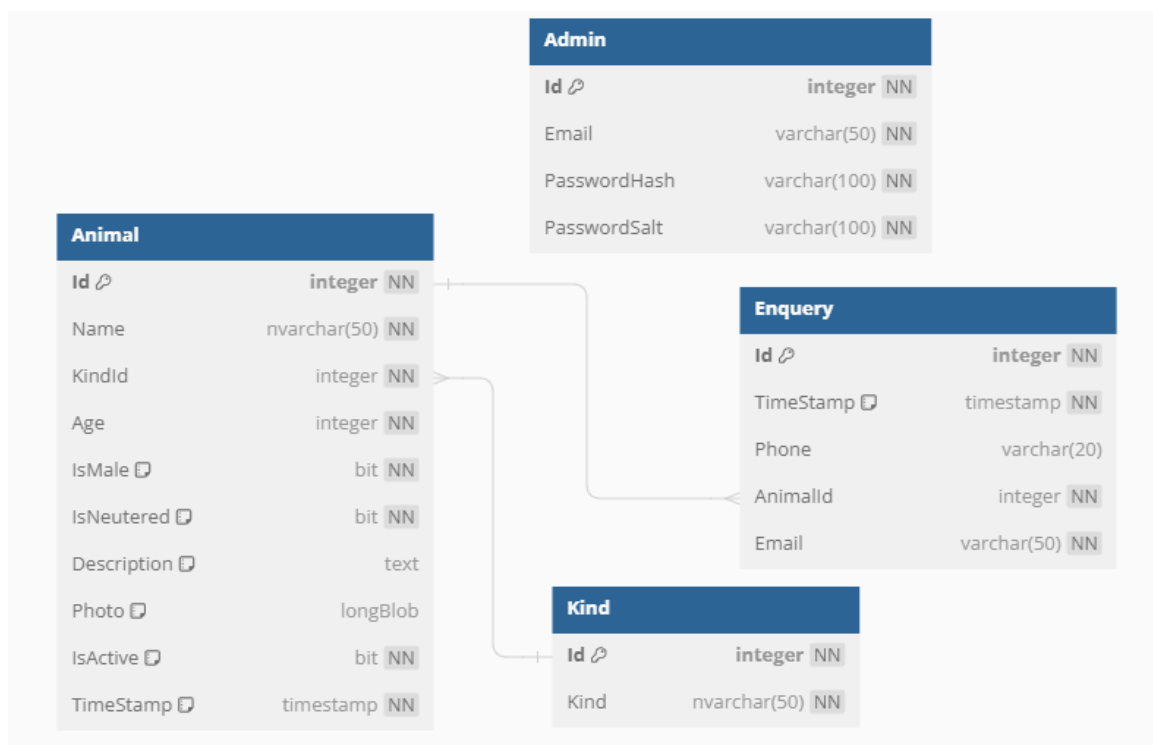
A „frontend” könyvtárban első lépésben a függőségeket kell telepíteni, mivel a „node_modules” könyvtár nem kerül eltávolításra GitHub-on.

A függőséget telepítéséhez az **'npm install'** parancsot kell kiadni terminálból, majd a telepítés sikeres lefutása után el lehet indítani az Angular alapú frontendet az **'ng serve -o'** paranccsal. A parancs sikeres lefutásával a projekt felülete betöltődik az alapértelmezett böngészőben. Amennyiben nem történne meg az automatikus megnyitása a felületnek, akkor azt a **'http://localhost:4200/'** címen lehet elérni.

Adatbázis

Az állatok és a hozzájuk kapcsolódó műveleti adatok tárolásáért az adatbázis felelős, melyet MySQL nyelven hoztunk létre, az ingyenes Aiven szerveren tárolva. Ezen keresztül tudjuk kezelni az összes admin funkciót, mint állatok listázása, új állat felvétele, módosítása, inaktívvá tétele. Ezen felül minden user számára elérhető szolgáltatás, az állatok listázása, illetve az érdeklődés, mely gombra, ha rákattint a felhasználó, e-mailt küld az admin e-mail címére az érdeklődésről, amit később hasznos lehet visszakeresni, hogy mely állatra mikor volt érdeklődés milyen felhasználói e-mailről.

AZ ADATBÁZIS TÁBLÁI ÉS MEZŐI



ADMIN TÁBLA

| Admin | | |
|--------------|--------------|----|
| Id | integer | NN |
| Email | varchar(50) | NN |
| PasswordHash | varchar(100) | NN |
| PasswordSalt | varchar(100) | NN |

Az admin tábla tartalmaz egy Id (Primary Key) mezőt, egy e-mail mezőt, melyre megkapja az állatok iránti érdeklődéseket, a PasswordHash és PasswordSalt-al megoldjuk a biztonságos jelszókezelést, melynek során a szózott jelszót az MD5 algoritmus egy hash-elt karakterlánccá alakítja.

ANIMAL TÁBLA

Az animal tábla tartalmazza a menhely állataira vonatkozó tulajdonságokat. Az Id mezője a PK (és AUTO INCREMENT). A Name az állat nevét tárolja, a KindId idegenkulcs a Kind tábla Id-jára. Ezen kívül megadjuk az állat korát (Age), a neme vagy hím, vagy nőstény, így egy boolean flag (1-es vagy 0) határozza meg, hogy hím-e az adott állat: IsMale. Mivel lehet ivartalanított is, felvettünk egy IsNeutered mezőt is, ezt is boolean flag tárolja. A Description-nel a tulajdonságot szövegesen tároljuk.

| Animal | | |
|-------------|--------------|----|
| Id | integer | NN |
| Name | nvarchar(50) | NN |
| KindId | integer | NN |
| Age | integer | NN |
| IsMale | bit | NN |
| IsNeutered | bit | NN |
| Description | text | |
| Photo | longBlob | |
| IsActive | bit | NN |
| TimeStamp | timestamp | NN |

A Photo Base64-es kódolással tárolja az állatok fotóit (longBlob). Az IsActive mező boolean flag-el határozza meg, hogy a menhelyen van-e az állat, vagy elvitték már.

A TimeStamp az állat felvételi időpontját jelöli az adatbázisba.

ENQUERY TÁBLA

| Enquery | | |
|-----------|-------------|----|
| Id | integer | NN |
| TimeStamp | timestamp | NN |
| Phone | varchar(20) | |
| AnimalId | integer | NN |
| Email | varchar(50) | NN |

Az Enquery tábla tartalmazza az Id (PK) mezőt, a TimeStamp-et, mely rögzíti az érdeklődés időpontját, az érdeklődő (user) telefonszámát és e-mail címét, ill. az AnimalId-t, ami FK az Animal tábla Id-jára.

KIND TÁBLA

| Kind | | |
|------|--------------|----|
| Id | integer | NN |
| Kind | nvarchar(50) | NN |

A Kind tábla az állatok fajtáit tartalmazza az Id-val (PK), és a Kind mezővel, ami a fajta neve.

Fejlesztői dokumentáció

BACKEND

Készítéshez használt programok:

ASP .NET Framework 6.0

C#

Visual Studio 2022

A kódot git verziókövetővel használjuk.

Könyvtárszerkezet:

backend/

| -Controllers/

| | -AdminController.cs

| | -AnimalController.cs

| | -EnqueryController.cs

| | -KindContoller.cs

- | -Helpers/
 - | | -HashHelper.cs
 - | | -JWTHelper.cs
 - | | -RemoveAuthorizeFilterOperationFilter.cs
- | -Models/
 - | | -RequestModels/
 - | | | -LoginRequestModel.cs
 - | | -ResponseModels/
 - | | | -AnimalsResponseModel.cs
 - | | | -BaseResponseModel.cs
 - | | | -EnqueryResponseModel.cs
 - | | | -KindResponseModel.cs
 - | | | -LoginResponseModel.cs
 - | | -Admin.cs
 - | | -Animal.cs
 - | | -Enquery.cs
 - | | -Kiind.cs
- | -AllatmenhelyDbContext.cs
- | -appsettings.json
- | -Program.cs
- | -BackendTests/
 - | | -TestHelper.cs
 - | | -ControllersUnitTests/
 - | | | -AdminControllerTests.cs
 - | | | -AnimalContollerTests.cs
 - | | | -EnqueryControllerTests.cs
 - | | | -KiindControlleTests.cs
 - | | | -TestBase.cs

Végpontok:

| Végpont | Metódus | Azonosítás | Leírás |
|--------------------------------|---------|------------|---------------------------------------|
| /Admin/Login | POST | nem | Bejelentkezés (csak adminnak) |
| /Animal/GetAllAnimal | get | nem | Állatok lekérdezése. |
| /Animal/GetAnimalById | get | nem | Egy állat lekérdezése. |
| /Animal/GetAnimalByKindId | get | nem | Állatok lekérdezése fajta szerint. |
| /Animal/CreateAnimal | post | igen | Állat felvétele. |
| /Animal/UpdateAnimal | put | igen | Állat módosítása. |
| /Animal/DeleteAnimal | delete | igen | Állat törlése/inaktívvá tétele. |
| /Enquery/GetAllEnqueries | get | igen | Érdeklődések lekérdezése. |
| /EnqueryGetEnqueriesByAnimalId | get | igen | Érdeklődések lekérdezése egy állatra. |
| /Enquery/CreateEnquery | post | nem | Érdeklődés. |
| /Kind/GetAllKinds | get | nem | Fajták lekérdezése. |
| /Kind/GetKindById | get | nem | Fajta lekérdezése Id alapján. |
| /Kind/CreateKind | post | igen | Fajta felvétele. |
| /Kind/UpdateKind | put | igen | Fajta módosítása. |

| | | | |
|------------------|--------|------|---------------------|
| /Kind/DeleteKind | delete | igen | Fajta inaktíválása. |
|------------------|--------|------|---------------------|

Általános működés:

A vezérlést, az üzleti logikát kontrollerek valósítják meg, melyek felelősek az adatbázis műveletekért, valamint tartalmazzák a frontend által hívható végpontokat.

A kliens aszinkron kéréseket intéz a backend felé REST API végpontok hívásával, melyek tartalmazzák a műveletekhez szükséges adatokat. A backend – amennyiben szükséges – az adatbázisból kinyert adatokat feldolgozva a HTTP Response BODY-jában küldi vissza az eredményt.

Osztályok

Entity Models

Feladata az adatbázis tábláinak reprezentálása: Admin, Animal, Enquery, Kind.

RequestModel

LoginRequestModel

Ez az osztály a bejelentkezési kérés modelljét reprezentálja, e-mail címet és jelszót tárol. A frontend részéről elküldött adatokat a RequestModel osztály fogadja, ezeket az adatokat a backend feldolgozza (autentikáció).

ResponseModel

BaseResponseModel

Ez az osztály visszajelzést ad a frontendnek, hogy az adott művelet sikeres volt-e vagy sem.

IsError: logikai érték, mely ha true, akkor hibát jelez, ha false, akkor nincs hiba.

ErrorMessage: string típusú, mely ha true, akkor tartalmazza a hiba leírását, vagy okát.

Exception: ha true, akkor az esetlegesen keletkezett kivételt tárolja.

Az összes többi modell gyakorlatilag ennek a leszármazottja, így minden modell képes arra, hogy hiba esetén annak részleteiről tájékoztatást küldjön a kliensnek.

AnimalResponseModel

Egy állatot képes visszaadni (pl Id alapján).

AnimalsResponseModel

Az osztály az állatokkal kapcsolatos adatok visszaadására szolgál a frontend felé.

EnqueryResponseModel

Egy állatra történő érdeklődést képes visszaadni.

EnquiriesResponseModel

Több érdeklődést képes visszaadni.

KindResponseModel

Egy fajtára történő érdeklődést képes visszaadni.

KindsResponseModel

Több fajtára történő érdeklődést képes visszaadni.

LoginResponseModel

Ez az osztály a bejelentkezés utáni válasz adatait tartalmazza, amelyeket a frontend megjelenít vagy felhasznál a további műveletekhez. A BaseResponseModel osztályból örököelve további információkat is tartalmaz, mint például a token és hibakezelési adatokat.

Controllers

A vezérlők feladata a weboldal backend-jén az üzleti logika kezelése, a HTTP kérések fogadása és feldolgozása (GET, POST, PUT, DELETE), validálása.

AdminController

Ez a kontroller felelős az adminisztrátor bejelentkezésének kezeléséért. Sikeres hitelesítés esetén JWT token generál, visszaküldi, ellenkező esetben hibaüzenetet küld.

AnimalController

Feladata az állatokkal kapcsolatos műveletek, és ha a hívás, vagy adatfeldolgozás során valamilyen problémába ütközünk, akkor egy megfelelő objektumban tájékoztatjuk a klienst.

Metódusok:

GetAllAnimals()

Az összes állat lekérdezéséért felelős az adatbázisból.

GetAnimalById(int id)

Egy bizonyos állat lekérdezését végzi, visszaadja annak adatait.

GetAnimalsByKindId(int kindId)

Adott fajtához tartozó állatot kérdezhetünk le ezzel a metódussal.

CreateAnimal([FromBody] Animal animal)

Új állat létrehozásáért felelős.

UpdateAnimal([FromBody] Animal newAnimal)
A metódus az állat módosítását végzi.

DeleteAnimal(int id)
Esetünkben az állat inaktívvá, ill. aktívvá tételét tehetjük meg a Delete-el.

EnqueryController

Feladata a felhasználói érdeklődésekkel kapcsolatos műveletek kezelése. Minden művelet biztosítja a hibák kezelését.

Metódusok:

GetAllEnqueries()
Az összes érdeklődés lekérdezését végzi.

GetEnqueriesByAnimalId(int animalId)
Azon érdeklődések lekérdezését végzi, melyek egy adott állathoz tartoznak.

CreateEnquery([FromBody] Enquery newEnquery)
Új érdeklődés felvétele.

KindController

Metódusok:

Feladata az állatfajtákkal kapcsolatos műveletek kezelése.

GetKindById(int id)
Összes állatfajta listázása.

CreateKind([FromBody] Kind newKind)
Új fajta létrehozása.

UpdateKind([FromBody] Kind newKind)
Fajta módosítása.

DeleteKind(int id)
Állatfajta inaktíválása.

Helpers

HashHelper

A HashHelper osztály a jelszavak generálásáért és ellenőrzéséért felelős. MD5 algoritmust használ.

GenerateSalt(int size = 32)
Ez a metódus véletlenszerűen generál egy sót, amely egy véletlenszerű string.

GenerateMD5Hash(string input, string salt)

Létrehoz egy MD5 hash-t az adott bemeneti adat és a só kombinációjából. Az inputot és a sót UTF-8 kódolásban átalakítja bájtokká, majd az MD5 hash-elés algoritmusát alkalmazza rájuk. A hash-t Base64-es kódolásban adja vissza.

VerifyMD5Hash(string input, string salt, string storedHash)

Ellenőrzi, hogy a megadott bemeneti adat és só kombinációjából származó hash megegyezik-e a tárolt PasswordHash-el.

JWTHelper

Az osztály segíti a JSON Web Token (JWT) kezelését a backend alkalmazásban.

GenerateToken(Admin adminUser, IConfiguration configuration)

A metódus egy JWT-t generál az admin számára a bejelentkezéskor. A token tartalmazza az adminisztrátor e-mail címét (mint claim), valamint a token lejáratí idejét is beállítja az alkalmazás konfigurációjából származó érték alapján.

ValidateToken(string token, string email, HttpContext context, IConfiguration configuration)

A függvény ellenőrzi a beérkező JWT érvényességét. Ellenőrzi, hogy a token aláírása megfelelő-e, és hogy a token érvényes időkorlátokkal rendelkezik-e. Ha a token érvénytelen vagy lejárt, a metódus visszautasítja a hozzáférést, és 401-es hibakódot küld a kliensnek. Ha a token érvényes, a metódus visszatérési értéke igaz lesz.

RemoveAuthorizeFilterOperationFilter

Ez az osztály egy Swagger szűrő, amely segít eltávolítani az autorizációt a dokumentált végpontokból, ha azokat megjelölték az [AllowAnonymous] attribútummal.

Az Apply metódus megvizsgálja a dokumentálni kívánt műveletet, és ellenőrzi, hogy az attribútum be van-e állítva a metódusra vagy a kontrollerre. Ha igen, akkor az összes válasz és biztonsági definíció eltávolításra kerül az OpenAPI műveletből (operation). Ez azt jelenti, hogy a Swagger-ben ezek a műveletek nem lesznek autorizációval ellátva. Ez különösen hasznos lehet olyan műveleteknél, amelyek publikusak és nem igényelnek bejelentkezést.

FRONTEND

A projekt frontend része valósítja meg a felhasználói felület megjelenítését, az adatok megjelenítését és bekérését felhasználói részre, illetve az adatok beszerzését és átadását a back-end részére.

Készítéshez használt alkalmazások, keretrendszerek.

- Szerkesztő program: Visual Studio Code
- Keretrendszerek:

- Angular, 16.2-es verzió
- Angular Material
- Bootstrap, 5.3 verzió
- Programozási nyelv: JavaScript, TypeScript
- Verzió követő rendszer: Git / GitHub

A frontend főbb egységei

A projekt frontend részét biztosító Angular keretrendszer három főbb egységre bontható, melyek tükrözik az MVC (Model-View-Controller) tervezés sajátosságait.

- Komponensek (components) [View és Controller]
- Modellek (models) [Model] és
- Szolgáltatások (services) [Controller]

A **Komponensek** az alábbiak:

- Animal-card
- Dialogs
- Pages

A **Modellek** csoport az alábbi elemekből áll:

- admin.model
- animal.model
- animalresponsemodel.model
- animalsresponsemodel.model
- baseresponsemodel.model
- enqueriesresponsemodel.model
- enquiry.model
- enquiryresponsemodel.model
- kind.model
- kindresponsemodel.model
- kindsresponsemodel.model
- loginrequestmodel.model
- loginresponsemodel.model

Végül a **Szolgáltatások** listsája:

- admin.service
- animal.service
- auth.interceptor

- auth.service
- enquiry.service
- kind.service

Az egyes egységek feladatai

Komponensek csoport

A komponensek valósítják meg az alkalmazás kinézetét, illetve a nézetekhez tartozó egyedi vezérléseket.

Pages

Egyetlen komponenst tartalmaz (allatlista.component), ami a webes alkalmazás egyetlen, dinamikus oldalát képviseli.

A vezérlő modul rész (allatlista.component.ts) az alábbi metódusokat tartalmazza:

- refreshAnimalList()
 - Az oldal megnyitásakor lekérdezi és eltárolja fajtánként az állatokat, hogy ne kelljen folyamatosan a backendet hívni.
- onLoginClick()
 - A bejelentkezéshez szükséges 'LoginDialog' dialógust hívja meg.
- onLogoutClick()
 - A bejelentkeztetett állapotot szünteti meg, azaz kilépteti a felhasználót.
- onCreateAnimalClick()
 - Bejelentkezett adminisztrátor felhasználó esetében a 'CreateanimalComponent' dialógust hívja meg, új állat felvételéhez.
- onlistOfEnqueries()
 - Bejelentkezett adminisztrátor felhasználó esetében az állatokra történt jelentkezések listáját nyitja meg a 'GetAllEnqueriesComponent' dialógus meghívásával.

Animal-card

Az Animal-card tartalmazza és jeleníti meg a backendtől lekért állatokat és azok adatát. A kapcsolódó vezérlő modul (animal-card.component-ts) valósítja meg a felületi vezérlést:

- showDetails()
 - A megjelenített állat képre kattintva megnyitja az 'AnimalDetails' dialógust.
- OnCreateEnqueryClick()
 - A nem bejelentkezett felhasználó esetén megjelenő „Gazdája szeretnék lenni” gomb mögötti vezérlés, ami a 'CreateEnqueryComponent' dialógus komponenst hívja meg.

- `OnUpdateAnimalClick()`
 - A bejelentkezett adminisztrátor felhasználó esetében megjelenő „Módosítás” gomb mögötti vezérlést látja el. Az `'UpdateanimalComponent'` dialógus komponenst hívja meg.
- `DeleteAnimal()`
 - A bejelentkezett adminisztrátor felhasználó esetében megjelenő „Törlés” gomb mögötti vezérlést látja el. Az `'DeleteanimalComponent'` dialógus komponenst hívja meg.

Dialogs

Angular Material Dialógus ablakok csoportja, melyek adatbeviteli és adatmegjelenítési feladatokat látnak el. Ezek az ablakok valósítják meg az oldal dinamikus jellegét.

animaldetails

- Az `Animal-card showDetails()` metódusától megkapja az aktuális állat példányt.
- A felületen kattintással kiválasztott állat adatait jeleníti meg. Kettő metódust tartalmaz:
 - `ngOnInit()`: Angular inicializációs metódus, a dialógus megnyitásakor fut le
 - `onNoClick()`: A dialógust zárja be

createanimal

- Dialógus ablak, melyben új állatot lehet rögzíteni
- Material form-field elemeket tartalmaz
- Vezérlő modulja (`createanimal.component.ts`) az alábbi metódusokat tartalmazza:
 - `onSaveAnimalClick()`
 - A mentés gombra kattintva feltölti az újonnan létrehozott `newAnimal`: `Animal` objektumot
 - A rögzítéskor megadott képet Base64-re kódolja a privát `convertImageToBase64()` metódussal
 - A `newAnimal` objektumot átadja a `animalService.createAnimal` részére
 - `onFileSelected()`
 - Az állat képét lehet vele kiválasztani az operációs rendszer dialógusának segítségével
 - `private convertImageToBase64()`
 - A megadott képet Base64-be átkódolja, aminek az eredmény `String` típusú adat.
 - `onCancelClick()`
 - A „Mégsem” gombhoz van rendelve és a feladata bezárni a dialógus ablakot, az állat mentése nélkül.

createEnquery

- Az állatokra történő jelentkezést valósítja meg a dialógus.
- A felület az állat képe mellett kettő beviteli mezőt tartalmaz.
- Vezérlő modulja (createEnquery.component.ts) az alábbi metódusokat tartalmazza:
 - o ngOnInit()
 - Angular inicializációs metódus, a dialógus megnyitásakor fut le.
 - o onSaveEnqueryClick()
 - Az „Elküld” gomb hatására létrehozza az új jelentkezést (newEnquery) és átadja azt a enqueryService szolgáltatás createEnquery metódusa részére
 - o onCancelClick()
 - A „Mégsem” gombra kattintva bezárja a dialógust
 - o getErrorMessage()
 - A felületi bevitel alapján hibaüzeneteket tud generálni az emial címet illetően
 - A nem kötelezően elvárt telefonszám ellenőrzése az Angular Validators segítségével történik

deleteanimal

- Az állat felületen megjelenő ikonja alatti „Törlés” gomb funkciója.
- Megjeleníti az állat adatait és megerősítést kér a törléshez
- Az alábbi funkciókat tartalmazza:
 - o ngOnInit()
 - Angular inicializációs metódus, a dialógus megnyitásakor fut le.
 - o onDeleteAnimalClick()
 - A törlésre jelölt állat azonosítóját adja át az animalService szolgáltatás deleteAnimal() funkciójának.
 - o onCancelClick()
 - A „Mégsem” gombra kattintva bezárja a dialógust a törlés megtörténte nélkül.

getAllEnqueries

- A rögzített jelentkezések listáját jeleníti meg Angular Material táblázat segítségével
- Az alábbi metódusokat tartalmazza:
 - o ngAfterViewInit()
 - Angular táblázathoz szedi össze az adatokat az enqueryService szolgáltatás getAllEnqueries funkciójával

- o `applyFilter()`
 - Angulat szűréshez szükséges metódus
- o `onCancelClick()`
 - A „Mégsem” gombra kattintva bezárja a dialógust

login

- Adminisztrátor felhasználó bejelentkeztetésére szolgál
- Bekéri az email címet és a jelszót
- Az alábbi funkciókból áll:
 - o `onNoClick()`
 - Bezárja a dialógus ablakot
 - o `getErrorMessage()`
 - Email címen végzett ellenőrzés hibaágát kezeli
 - o `onLoginClick()`
 - A „Bejelentkezés” gombra az `adminService` szolgáltatás login funkciójával bejelentkezteti a felhasználót, illetve hibakezelést végez.

updateanimal

- Meglévő állatok módosítására szolgál
- Megjeleníti az állat adatait, melyet az adminisztrátor felhasználó akár külön-külön is módosíthat
- Az alábbi funkciókból áll:
 - o `ngOnInit()`
 - Angular inicializációs metódus, a dialógus megnyitásakor fut le. Beállítja az egyes mezők értékét a szerkesztéshez.
 - o `onSaveAnimalClick()`
 - A frissített adatokat rögzíti az `updatedAnimal` objektumban és átadja azt az `animalService` szolgáltatás `updateAnimal` funkciójának.
 - o `onFileSelected()`
 - Fájlok beolvasását teszi lehetővé az aktuális állathoz.
 - o `private convertImageToBase64()`
 - A megadott képet konvertálja Base64 stringgé.

Modellek csoport

A modellek a projektben használt objektumokat írják le, reprezentálva a backend oldalon meglévő objektumokat.

admin.model

Adminisztrátor felhasználó modellje.

animal.model

Az adatbázisban lévő 'Animal' objektum leírója.

animalresponsemodel.model

A backendtől kapott Animal model. A BaseResponseModel-t örökli.

animalsresponsemodel.model

A backendtől kapott Animal model tömb. A BaseResponseModel-t örökli.

baseresponsemodel.model

A backendtől kapott válaszok sikerességének modellje. Szintén a BaseResponseModel-t örökli.

enqueriesresponsemodel.model

A foglalások lekérdezésének tömbje. A BaseResponseModel-t örökli.

enquery.model

A foglalás sémáját tartalmazza.

enqueryresponsemodel.model

A foglalási műveletre adott válasz modellje.

kind.model

Az adatbázisban lévő 'Kind' tábla reprezentációja.

kindresponsemodel.model

Az állat fajtájának meghatározására szolgáló modell

kindsresponsemodel.model

Több állat (tömb) fajtájának meghatározására szolgáló modell

loginrequestmodel.model

Az adminisztrátor felhasználó belépéséhez szükséges adatok modellje

loginresponsemodel.model

A bejelentkezésre adott válasz sémája

Szolgáltatások csoport

admin.service

Az adminisztrátor felhasználó bejelentkeztetését végzi a backend Admin/Login szolgáltatásával. Visszaadja a bejelentkezés eredményét. Csak a login() metódusa van.

animal.service

A Backend Animal szolgáltatása alatt lévő végpontok segítségével hajtja végre a CRUD műveleteket, az alábbi metódusok segítségével:

- getAnimalsByKindId() – GET
 - A tárolt állatokat fajta azonosító alapján kérdezi le a backend /GetAnimalsByKindId szolgáltatásával. Animal[] tömböt ad vissza.
- getAnimalById() - GET
 - Állatot azonosító alapján kérdez le, a backend /GetAnimalById segítségével. Egy konkrét Animal objektumot ad vissza.
- createAnimal() - POST
 - Új állat létrehozását végzi a backend /CreateAnimal végpontját meghívva. A létrehozás kérésre adott választ adja vissza.
- updateAnimal() - PUT
 - Meglévő állat adatait frissíti a backend /UpdateAnimal szolgáltatásával. Visszaadja az aktuálisan módosított állatot
- deleteAnimal() – DELETE
 - Meglévő állat törlését végzi, a backend /DeleteAnimal végpontjának segítségével, a törölni kívánt állat azonosítójával kiegészítve.

auth.interceptor

Speciális szolgáltatás. Minden HTTP kérést elfog és hozzáteszi a BEARER token. Így adjuk át a JWT token minden híváskor.

auth.service

A felhasználó beléptetését végzi, és tárolja el a böngésző lokális tárolójában.

enquery.service

Az állatokra való jelentkezésekkel kapcsolatos metódusokat tartalmaz.

- createEnquery()
 - A jelentkezéseket adja át a backend Enquery/CreateEnquery végpontjára. A backend választ adja vissza.
- getAllEnqueries()
 - A foglalásokat kérdezi le a backend Enquery/GetAllEnqueries funkciójával. A lekérdezés választ adja vissza.

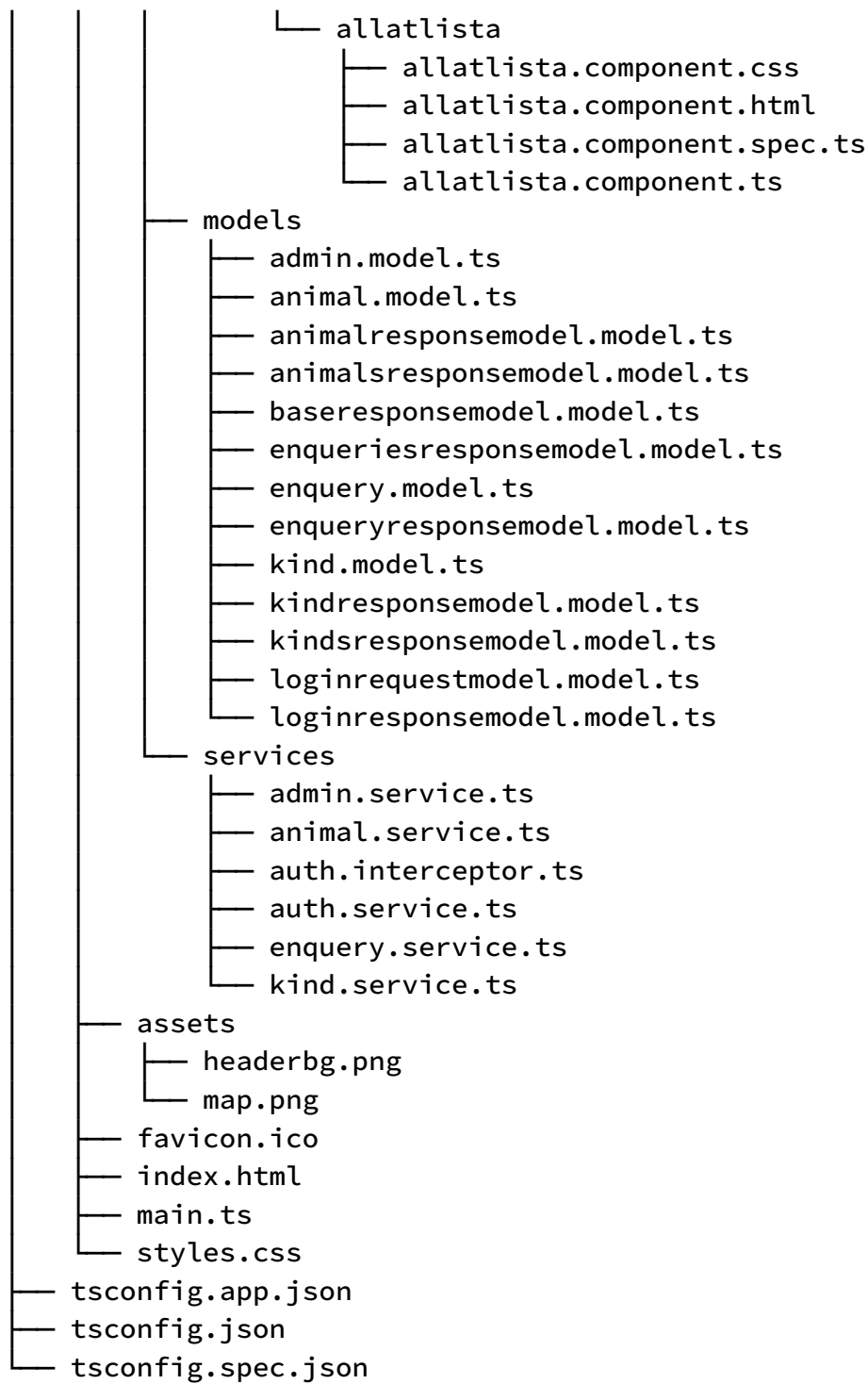
kind.service

Az állatok fajtáját kérdezi le a backend Kind/GetAllKinds végpontjáról. Az állatok fajtáját adja vissza a KindsResponseModel-ként.

Könyvtárszerkezet

A frontend könyvtárszerkezete az alábbi hierarchia szerinti:

```
frontend/
├── angular.json
├── package.json
├── package-lock.json
├── README.md
├── src
│   ├── app
│   │   ├── app.module.ts
│   │   ├── app-routing.module.ts
│   │   └── components
│   │       ├── animal-card
│   │       │   ├── animal-card.component.css
│   │       │   ├── animal-card.component.html
│   │       │   ├── animal-card.component.spec.ts
│   │       │   └── animal-card.component.ts
│   │       ├── dialogs
│   │       │   ├── animaldetails
│   │       │   │   ├── animal.details.html
│   │       │   │   └── animal.details.ts
│   │       │   ├── createanimal
│   │       │   │   ├── createanimal.component.html
│   │       │   │   └── createanimal.component.ts
│   │       │   ├── createEnquery
│   │       │   │   ├── createEnquery.component.html
│   │       │   │   └── createEnquery.component.ts
│   │       │   ├── deleteanimal
│   │       │   │   ├── deleteanimal.component.html
│   │       │   │   └── deleteanimal.component.ts
│   │       │   ├── getAllEnqueries
│   │       │   │   ├── getAllEnqueries.component.html
│   │       │   │   └── getAllEnqueries.component.ts
│   │       │   ├── login
│   │       │   │   ├── logindialog.html
│   │       │   │   └── logindialog.ts
│   │       │   └── updateanimal
│   │       │       ├── updateanimal.component.html
│   │       │       └── updateanimal.component.ts
│   │       └── pages
```



Teszt dokumentáció

A backend teszteléséhez unit teszteket használtunk, melyet kontrollerenkénti bontásban hoztunk létre. A TestHelper osztály segítségével felépítünk egy memóriabeli adatbázist, és a kontrollerek által végzett műveleteket ezen hajtjuk végre. Az in-memory adatbázis felépítéséhez az AllatmenhelyDBContext osztályt használtuk, így a valós adatbázisbeli táblákat reprezentáló model-osztályok használhatóak a unit tesztekben és teljes mértékben leképezik a valós viselkedést.

A tesztek a hagyományos AAA (Arrange, Act and Assert) szabály szerint lettek kialakítva, ahol az Arrange részben létrehozuk a tesztelés során használatos objektumokat és beállítjuk azok kezdőértékét. Az Act-szekció a tesztelendő kontroller-metódus meghívását végzi, a kiértékelés pedig az Assert részben történik.

A tesztek lefedik az összes lehetséges esetet, ezért egy-egy kontroller metódushoz több unit teszt is tartozik, vizsgálva a sikeres végrehajtást, valamint az esetleges hibára futásokat is.

Mivel minden teszt a memóriabeli adatbázis felépítésével és bekonfigurálásával kezdődik, az összes teszt osztályt a TestBase Ősosztályból származtatjuk, ami azért felelős, hogy összefogja a közös viselkedést / funkciókat.

CONTROLLERSUNITTESTS:

AdminControllerTests

Az AdminController osztályának metódusait teszteli.

LoginTest_Successful()

Ez a metódus teszteli a sikeres bejelentkezést olyan adatokkal, amelyek megfelelnek az adatbázisban tárolt admin adatoknak. Az elvárt eredmény egy LoginResponseModel típusú objektum. Ahol nincs hiba (IsError érték false), az admin azonosítója és e-mail címe megegyezik az elvárt értékekkel, valamint meg kell kapnunk a JWTHelper osztály GenerateToken metódusa által generált JWT tokenet.

LoginTest_Unsuccessful_WrongEmail()

Ez a metódus teszteli a bejelentkezés sikertelenségét, ha az e-mail cím nem felel meg az adatbázisban tárolt admin e-mail címének. Az elvárt eredmény ahol van hiba (IsError érték true), és az üzenet tartalma "Admin nem található".

LoginTest_Unsuccessful_WrongPassword()

Ez a metódus teszteli a bejelentkezés sikertelenségét, ha a megadott jelszó nem egyezik meg az adatbázisban tárolt admin jelszóval. Az elvárt eredmény ahol van hiba (IsError érték true), és az üzenet tartalma "Admin email vagy jelszó nem megfelelő".

AnimalControllerTests

Az AnimalController osztályának különböző metódusait teszteli.

GetAllAnimalsTest()

Ez a tesztet ellenőrzi, hogy a GetAllAnimals metódus helyesen visszaadja-e az összes állatot, és hogy a visszatérő válasz nem tartalmaz-e hibát.

GetAnimalByIdTest_Successful()

Ellenőrzi, hogy a GetAnimalById metódus helyesen adja-e vissza az adott azonosítójú állatot, és hogy a válasz nem tartalmaz-e hibát.

GetAnimalByIdTest_Unsuccessful()

Ez a teszt ellenőrzi, hogy a GetAnimalById metódus helyesen kezeli-e a nem létező azonosítójú állat lekérését. A teszt sikerességét a visszaadott hiba jelzi.

GetAnimalsByKindIdTest_NoCatsOrDogs()

Mivel három állatfajta kategória van, itt azt ellenőrizzük, hogy csak az „Egyéb” kategóriájú állatokat adja vissza.

GetAnimalsByKindIdTest_CatsOnly()

A teszt ellenőrzi, hogy csak a macska fajta kategóriájú állatokat kapjuk vissza.

CreateAnimalTest()

Ez a teszt vizsgálja, hogy a CreateAnimal metódus létrehoz-e egy új állatot, és hogy a válasz nem tartalmaz hibát.

UpdateAnimalTest_NotFound()

A teszt vizsgálja, hogy le van-e kezelve az az eset, amikor egy nem létező állat id-val próbálunk módosítani.

UpdateAnimalTest_Successful()

A teszt vizsgálja, hogy sikeresen tudunk-e módosítani a már meglévő állat tulajdonságain.

DeleteAnimalTest_Successful()

A teszt vizsgálja, hogy sikeresen tudunk-e törölni/inaktiválni egy már meglévő állatot.

DeleteAnimalTest_NotFound()

Ellenőrzi, hogy a DeleteAnimal metódus helyesen kezeli-e azt az esetet, ha nem létező id-jú állatot próbálunk törölni.

EnquiryControllerTests

Az EnquiryController osztály műveleteit teszteli.

GetAllEnquiriesTest()

Ellenőrzi, hogy a GetAllEnquiries metódus helyesen visszaadja-e az összes érdeklődést, és hogy a visszatérő válasz nem tartalmaz hibát.

GetEnquiriesByAnimalIdTest()

Ellenőrzi, hogy a GetEnquiriesByAnimalId metódus helyesen visszaadja-e az adott állathoz tartozó lekérdezéseket, és hogy a válasz nem tartalmaz hibát.

CreateEnquiryTest() Ellenőrzi, hogy a CreateEnquiry metódus helyesen létrehozza-e az új érdeklődést, és hogy a visszatérő válasz nem tartalmaz hibát. Ellenőrzi továbbá, hogy az új érdeklődésben az érdeklődő adatai helyesen szerepelnek-e.

KindControllerTests

A KindController osztály műveleteit teszteli.

GetAllKindsTest()

Ellenőrzi, hogy a GetAllKinds metódus helyesen visszaadja-e az összes fajtát, és hogy a válasz nem tartalmaz-e hibát.

GetKindByIdTest_Successful()

A teszt vizsgálja, hogy a GetKindById metódus helyesen adja-e vissza az adott azonosítójú fajtát, és hogy a válasz nem tartalmaz-e hibát.

GetKindByIdTest_Unsuccessful()

Ez a teszt ellenőrzi, hogy a metódus helyesen kezeli-e a nem létező azonosítójú fajta lekérését. A teszt sikerességét a visszaadott hiba jelzi.

CreateKindTest_Successful()

Ez a teszt ellenőrzi, sikeresen létre tudunk-e hozni egy új állatfajtát.

CreateKindTest_Unsuccessful()

Vizsgálja, hogy a fajta létrehozása esetén kezeljük-e azt az esetet, amikor olyan fajtát akar létrehozni, ami már létezik. Ez ugyanis nem engedélyezett.

UpdateKindTest_Successful()

A teszt vizsgálja, hogy tudjuk-e módosítani a kiválasztott állatfajta nevét.

UpdateKindTest_Unsuccessful_NotFound()

Ellenőrzi, hogy fajta módosításánál kezeljük-e azt a hibalehetőséget, amikor az admin egy nem létező id-jú fajtát próbál módosítani.

UpdateKindTest_Unsuccessful_AlreadyExists()

A teszt vizsgálja, hogy a módosítandó fajtánál megadott új állatfajta-név nem létezik-e már egy másik állatfajtánál, ezt ugyanis nem engedjük, mert duplikációhoz vezetne.

DeleteKindTest_Successful()

A teszt ellenőrzi, hogy sikeresen tudunk-e törölni/inaktiválni egy már meglévő fajtát.

DeleteKindTest_NotFound()

Ellenőrzi, hogy kezeljük-e azt az esetet, amikor az admin egy olyan id-jú fajtát próbál törölni/inaktiválni, ami nem létezik.

Front end teszt

A front end tesztelését manuálisan végeztük. Az ellenőrzések szempontjai voltak:

- Felületi elemek elhelyezkedése
- Funkció elérhetősége felhasználótól függően
 - A nem bejelentkezett felhasználó esetében csak a „Bejelentkezés” gomb jelenik meg a felső sorban, illetve az állatok képei alatt a „Gazdája szeretnék lenni” gomb található.
 - A bejelentkezett felhasználóval megjelennek az „Állat felvétele”, „Jelentkezések” és „Kilépés” gombok. Továbbá az állatok kártyái alatt a „Módosítás” és „Törlés” gombok szerepelnek.
- Beviteli mezők esetében
 - Az állat felvételénél nincs külön megkötés, a bevitt adatok bekerülnek az adatbázisba és változatlan formában jelennek meg a felületen;
 - Módosítás esetében szintén nincs kikötés - módosított adatok bekerülnek az adatbázisba és a módosított formában jelennek meg a felületen;
 - Az érdeklődések rögzítésénél a két beviteli mezőre ellenőrzés van:
 - A telefonszám nem kötelező, de ha meg van adva, akkor annak kötött formájúnak kell lennie: nn-nnn-nnnn (például: 20-123-4567). Ha ennek a kikötésnek nem felel meg a bevitt adat, akkor a felület nem fogadja el;
 - Az email cím megadása kötelező, és az email cím formáját Angular függvény ellenőrzi – a '@' jelnek szerepelnie kell benne.

Felhasználói dokumentáció

Felhasználói oldalról nézve a webes felület kezeléséhez nincs szükség különösebb hardver és szoftver igényre, a honlap egy átlagos teljesítményű asztali számítógépen, laptopon, tableten és mobiltelefonon is jól használható, azonban minden esetben internet szükséges hozzá.

Szoftver igényt tekintve bármilyen böngésző használható hozzá, legyen az Google Chrome, Mozilla Firefox, stb.

A webes felületen elérhető lehetőségek:

- Böngészés, olvasás
- Menüpontokon navigálás

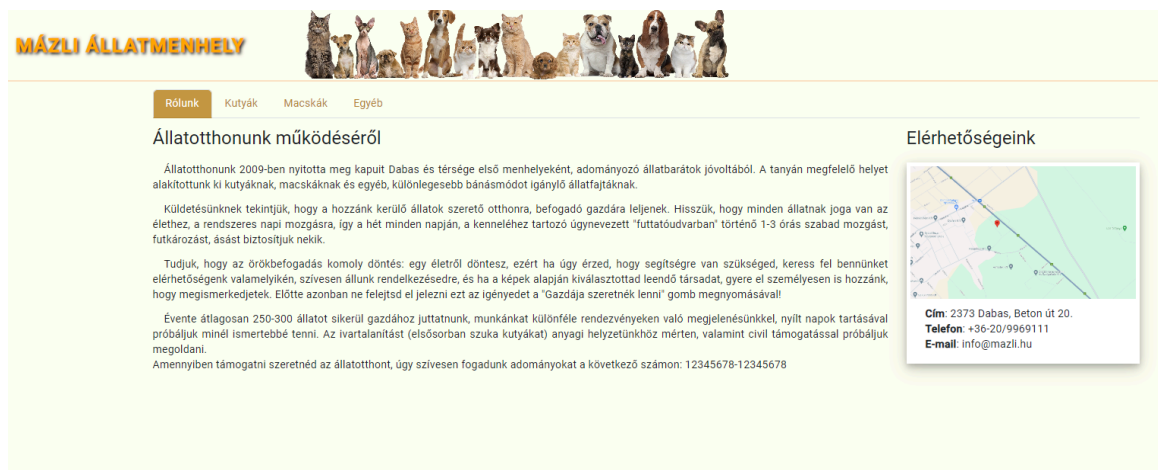
- Állatok foglalása
- Állatok keresése

Admin:

- Bejelentkezés, kilépés
- Állat felvétele, módosítása, törlése
- Érdeklődések listázása

FÖLDAL

A legelső oldal, amivel találkozik a felhasználó, amikor megnyitja a honlapot. A menüsorban megtalálhatja a különböző állatfajtaikat és a menhely elérhetőségét. Jobb oldalon az admin bejelentkezési gomb van.



MENÜPONTOK


Kutyák menüpont

A Kutyák menüpontra klikkelve láthatjuk az elvihető kutyák főbb tulajdonságait kártyákba rendezve, úgy, mint név, kor, ivar és tulajdonság. A Gazdája szeretné lenni gombra nyomva egy form ablak ugrik fel, ahol be kell írunk az e-mail címünket és a telefonszámunkat. Az összefoglaló nézetre kattintva felugrik egy ablak, melyben az állat összes tulajdonsága és a teljes leírása látható. Ugyanezek a lehetőségek a Macskáknál, és az Egyéb állatoknál is.

MÁZLI ÁLLATMENHELY

Rólunk Kutyák Macskák Egyéb


Csahos
4 hónapos
♂



A japán Akiita egy nyugodt és méltóságteljes kutya...

Gazdája szeretné lenni


Bátor
24 hónapos
♀



Bátor rendelkezik a keverék kutyák minden bájával...

Gazdája szeretné lenni


Kócos
36 hónapos
♀ ivartalanított



Kócos különleges, szeretetre éhes, 3 év körüli, puli k...

Gazdája szeretné lenni


Blóki
56 hónapos
♂



Blóki nagyon szerethető, vidám, kedves kutya.

Gazdája szeretné lenni


Milo
86 hónapos
♂ ivartalanított



Milo már nyugodt, csak békességre, szeretetre vágy...

Gazdája szeretné lenni

Reni
12 hónapos
♀



Roni Amstafff jellegű, gyerekszerető, vidám, könnye...


Gazdája szeretné lenni

A képre klikkelve láthatjuk az állat tulajdonságait, a teljes jellemzését:

Részletek

Kócos
36 hónapos
♀ ivartalanított

Kócos különleges, szeretetre éhes, 3 év körüli, puli keverék kutya. Nagyon kedves, bájos, okos, imádja a simogatást, bálványként tekint az emberre. Új gazdája igazi kis négy lábú társra talál majd benne! Nagyon szeret sétálni, más kutyákkal is jó a viszonya.



Érdeklődés rögzítése

A „Gazdája szeretné lenni” gombra nyomva felugrik egy ablak, ahol az adatok beírása után elküldhető az érdeklődés, vagy a mégsem gombbal bezárjuk az ablakot.

Érdeklődés rögzítése

Telefonszám
nn-nnn-nnnn

Email cím*



Elküld Mégsem

Macskák menüpont

MÁZLI ÁLLATMENHELY

RólunkKutyákMacskákEgyéb

Cirmi
4 hónapos
♂ ivartalanított

Nagyon kedves, barátságos, első cicának is ideális. ...
[Gazdája szeretné lenni](#)

Mirci
36 hónapos
♀ ivartalanított

Bár elsőre félénk lehet idegenekkel, de ha egyszer ...
[Gazdája szeretné lenni](#)

Kormi
23 hónapos
♀ ivartalanított

Vagány, játékos, de ugyanakkor bújós kis kedvenc.
[Gazdája szeretné lenni](#)

Mr. Macska
44 hónapos
♂

Macsek
[Gazdája szeretné lenni](#)

Egyéb menüpont

MÁZLI ÁLLATMENHELY

RólunkKutyákMacskákEgyéb

Füles
12 hónapos
♀ ivartalanított

Füles kicsit félénk nyuszi, de hamar megbarátkozik, ...
[Gazdája szeretné lenni](#)

Pocok
12 hónapos
♂ ivartalanított

Pocok kézhez szokott, kedves, gondoskodásra vágy...
[Gazdája szeretné lenni](#)

Bubi
38 hónapos
♂

Bubi ragadozó, ezért otthoni tartásnál figyelni kell ...
[Gazdája szeretné lenni](#)

Rózi
28 hónapos
♀

Rózi nagyon kíváncsi, élénk, sok törődést igénylő ki...
[Gazdája szeretné lenni](#)

Arline
12 hónapos
♀

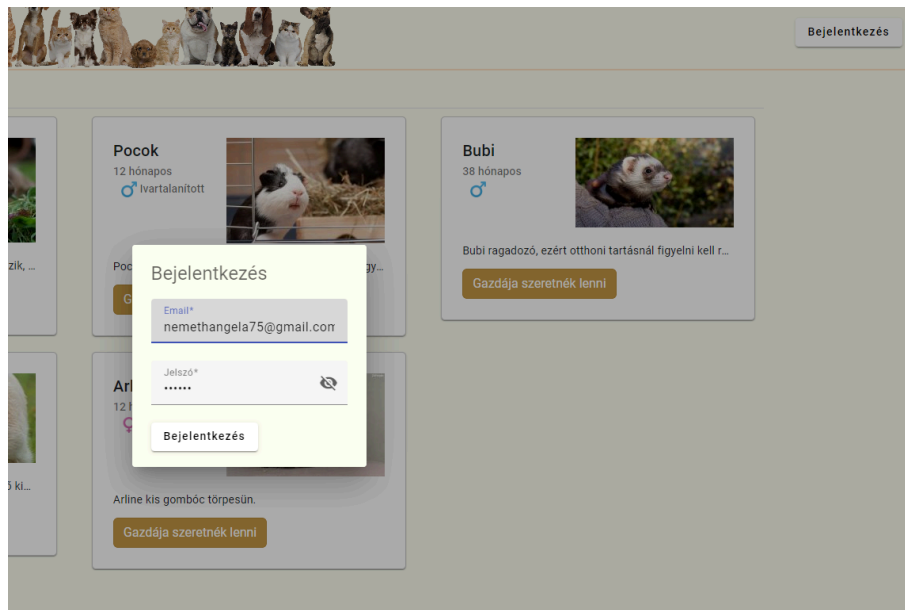
Arline kis gombóc törpesүн.
[Gazdája szeretné lenni](#)

ADMIN FUNKCIÓK

Az adminisztrátor bejelentkezés után tud állatot felvenni, módosítani, törölni. Törlésnél az adatbázisból nem törlődik, csak inaktív lesz. Az érdeklődések lekérdezésénél csak az aktív állapotúak jelennek meg. Az admin funkciókhoz autentikálni kell, minden más funkció elérhető bejelentkezés nélkül.

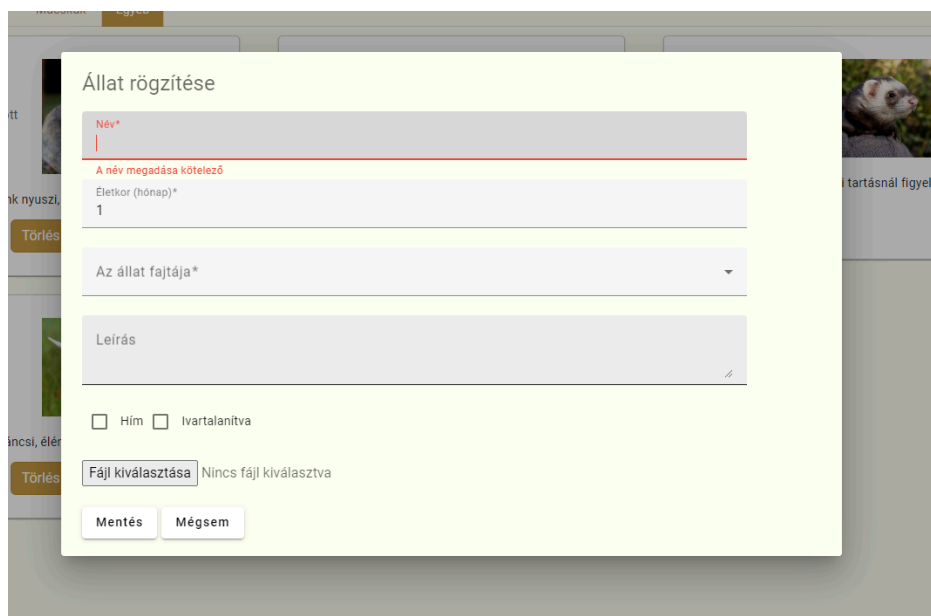
Bejelentkezés

Bejelentkezéshez meg kell adnia az e-mail címét és a jelszavát.



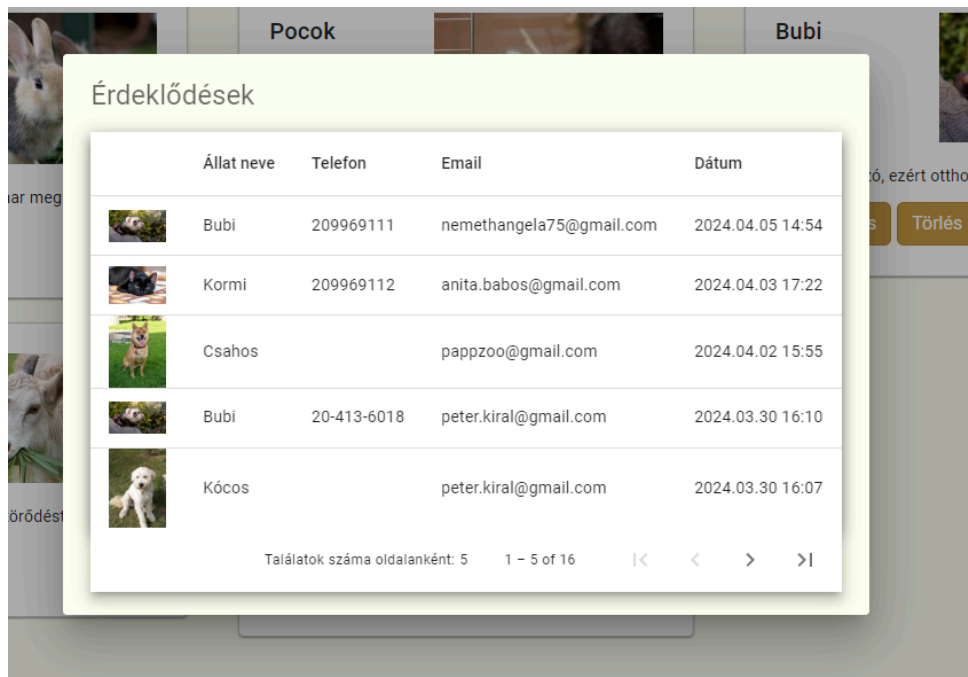
Új állat felvétele






Bejelentkezés után az admin tud új állatot rögzíteni, az állat nevének, korának, fajtájának, nemének megadásával, és írhat jellemzést is, ill. feltölti az állat fotóját.



Érdeklődések listázása

Bejelentkezés után az admin listázhatja az érdeklődéseket.

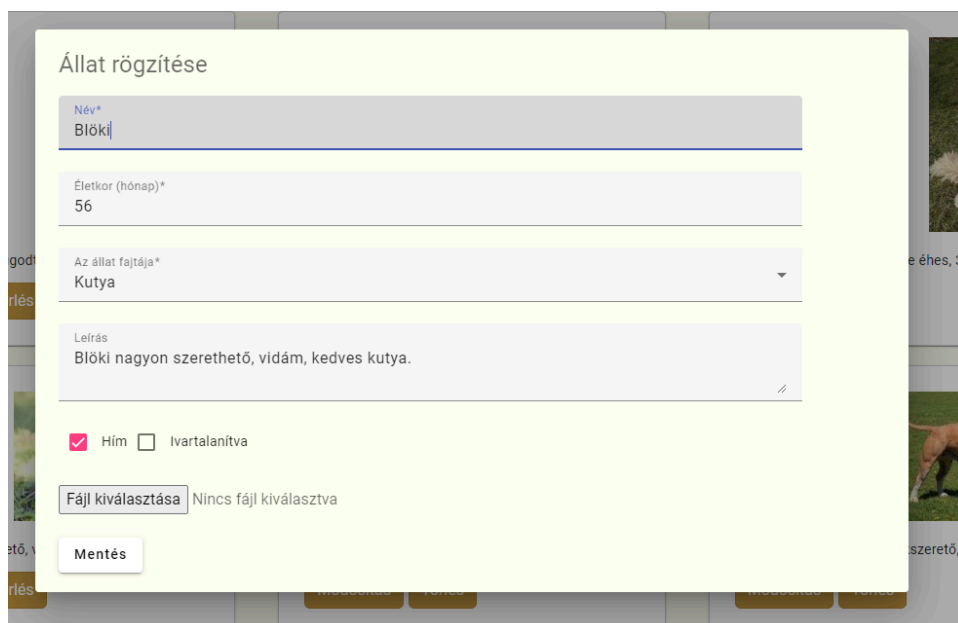


| | Állat neve | Telefon | Email | Dátum | |
|---|------------|-------------|--------------------------|------------------|---------|
|  | Bubi | 209969111 | nemethangela75@gmail.com | 2024.04.05 14:54 | Törölés |
|  | Kormi | 209969112 | anita.babos@gmail.com | 2024.04.03 17:22 | |
|  | Csahos | | pappzoo@gmail.com | 2024.04.02 15:55 | |
|  | Bubi | 20-413-6018 | peter.kiral@gmail.com | 2024.03.30 16:10 | |
|  | Kócos | | peter.kiral@gmail.com | 2024.03.30 16:07 | |

Találatok száma oldalanként: 5 1 - 5 of 16 |< < > >|

Állat módosítása

Adminként lehetőség van a kiválasztott állat tulajdonságainak módosítására. Ekkor megjelennek a felvitt adatok, így látható a változtatni kívánt paraméter.



Állat rögzítése

Név*
Blöki

Életkor (hónap)*
56

Az állat fajtája*
Kutya

Leírás
Blöki nagyon szerethető, vidám, kedves kutya.

☒ Hím ☐ Ivartalanítva

Fájl kiválasztása Nincs fájl kiválasztva

Mentés

Állat törlése

Törléskor az adatbázisból nem törlődik az állat, hanem inaktívvá válik. Mielőtt végleg elküldenénk a parancsot, a Mégsem gombra kattintva visszaléphetünk.



Összefoglalás

Olyan reszpónzív weboldal létrehozása volt a cél, ahol a felhasználók állatokat, kis társakat kereshetnek, és fogadhatnak örökbe, ezzel is segítve egy árva élőlény sorsát, hogy szerető, gondoskodó családban nevelkedhessen. A weboldalt böngészve kereshetünk bizonyos állatfajokra, pl. csak kutyára, kilistázhathatjuk őket, ahol is láthatjuk az adott állat tulajdonságait. A felhasználók sok esetben nem kedvelik az olyan oldalakat, ahol bizonyos funkciók regisztrációhoz kötöttek, így ezt elhagytuk, esetünkben az „Érdeklődés” gomb váltja ki ezt. Későbbiekben szeretnénk az oldal lehetőségeit bővíteni.

Az eredeti ötletünk egy robot kölcsönző létrehozása volt, viszont azt elvetettük, mert nem találtuk elég életszerűnek, nem tudtunk igazán azonosulni a témakörrel, és sokkal hasznosabbnak is találtuk az elhagyott, elárvult állatok segítését.

Úgy érezzük, hogy lehetőségeinkhez képest sikerült jól megvalósítani a projektet, talán a legnagyobb kihívást az jelentette, hogy eldöntsük, milyen nyelven készüljön a program.

Úgy döntöttünk, hogy a backend C#-ban íródik majd, a frontendet pedig Angular keretrendszerben valósítottuk meg. de mindezek előtt létre kellett hoznunk az átgondolt, anomáliáktól mentes adatbázist.

TOVÁBBI CÉLKITŰZÉSEK

- Vélemény írási lehetőség a felhasználók számára
- Állatok keresési lehetősége
- Admin funkciók bővítése (pl. több fotó egy állathoz, inaktív állat újraaktiválása)
- Keresőoptimalizálás