Hódmezővásárhelyi SZC Makói
Návay Lajos Technikum és Kollégium

Vizsgaremek dokumentáció

Pajti Paradicsom

weboldalhoz

Készítette: Bézi Katalin

Szabó-Juhász Eszter

Szakképzés megnevezése: Szoftverfejlesztő és -tesztelő

2025.

Tartalomjegyzék

Bevezetés

A modern életvitel egyre gyakrabban hozza magával azt a kihívást, hogy a háziállat-tartók – különösen a kutyatulajdonosok – nem tudnak elegendő időt tölteni kedvenceikkel munka, utazás vagy más elfoglaltság miatt. Ilyen esetekben jelenthet ideális megoldást egy jól szervezett kutyapanzió vagy kutyanapközi, ahol a kutyák szakszerű ellátásban részesülnek. Ezek az intézmények nem csupán megőrzést biztosítanak, hanem élménydús, társasági környezetet kínálnak a négylábúak számára. A gondoskodó, felkészült személyzet, valamint a kiegészítő szolgáltatások – mint például a kutyakozmetika, egészségügyi felügyelet vagy akár fotózás – mind hozzájárulnak ahhoz, hogy a gazdik biztonságban tudják kedvencüket. Egy egyszerű, online foglalási rendszer jelentősen megkönnyítheti a szolgáltatások igénybevételét, így növelve a panziók elérhetőségét és hatékonyságát.

A projekt célja, motivációja

A házi kedvencek – különösen a kutyák – szerepe napjainkban jelentősen felértékelődött, sokan családtagként tekintenek rájuk. Ezzel párhuzamosan nőtt az igény a professzionális állatgondozási szolgáltatások iránt. A kutyapanziók és napközik alapítása így nemcsak társadalmilag indokolt, hanem üzletileg is vonzó lehetőség. A városi életforma és a hosszú munkaidők miatt sok gazdi keres megbízható, kényelmes megoldást kutyája napközbeni elhelyezésére. Az online foglalási rendszer célja, hogy ezt az igényt kielégítse: gyors és egyszerű lehetőséget biztosít a szolgáltatások lefoglalására, a kutya adatainak kezelésére, és speciális igények megadására. A felhasználóbarát platform révén a panziók könnyebben elérhetik célközönségüket, miközben a gazdik is nyugodtabban végezhetik mindennapi teendőiket, tudván, hogy kedvencük jó kezekben van.

A csapat bemutatása

A Pajti Paradicsom projekt egy lelkes, kétfős csapat közös kezdeményezésére jött létre. Bézi Katalin, aki a projektvezetői szerepet töltötte be, és Szabó-Juhász Eszter közösen dolgoztak azon, hogy egy felhasználóbarát és hatékony foglalási rendszert hozzanak létre a kisállattartók számára. A munkafolyamatokat nem választották szét élesen, ehelyett dinamikus együttműködésben, folyamatos kommunikáció mellett haladtak előre.

Mindketten aktívan kivették a részüket az ötletelésből, a rendszertervezésből, a programkód megírásából, valamint a rendszer teszteléséből és finomhangolásából is. A közös cél az volt, hogy olyan alkalmazást hozzanak létre, amely valós igényekre nyújt modern megoldást. Az egyéni feladatok elvégzése mellett nagy hangsúlyt fektettek arra, hogy támogassák egymást a felmerülő problémák megoldásában is.

A projekt alapjául szolgáló technológiák – köztük a HTML, CSS, JavaScript, PHP és MySQL – az iskolai tanulmányok során váltak ismerőssé számukra, de a fejlesztés során lehetőség nyílt ezek mélyebb gyakorlati elsajátítására. A fejlesztési folyamat során a csapat megtapasztalhatta, milyen fontos szerepe van az önálló tanulásnak, a hatékony csapatmunkának és a kreatív gondolkodásnak egy sikeres informatikai projekt megvalósításában.

Pajti Paradicsom struktúra

A **Pajti Paradicsom** egy AngularJS alapú webalkalmazás, amely kisállatok számára nyújt különféle szolgáltatásokat, például panzió, kozmetika és fotózás. Az alkalmazás célja, hogy a felhasználók könnyedén foglalhassanak időpontot kedvenceik számára, valamint információkat kapjanak a szolgáltatásokról és árakról.

A weboldal fő funkciói

1. **Időpontfoglalás**:
    - A felhasználók regisztrált kisállataik számára foglalhatnak időpontot különböző szolgáltatásokra.
    - A szolgáltatások típusai: panzió, kozmetika, fotózás.
    - Az időpontfoglalás során a felhasználók megadhatják a dátumot, időpontot, és opcionálisan megjegyzéseket is fűzhetnek a foglaláshoz.
2. **Szolgáltatások és árak megtekintése**:
    - A felhasználók böngészhetik a különböző szolgáltatásokat és azok árait.
    - Az árak szűrhetők a szolgáltatás típusa alapján.
3. **Kisállatok kezelése**:
    - A felhasználók regisztrálhatják kisállataikat, és megadhatják azok adatait (név, típus, kor, leírás).
    - A regisztrált kisállatok adatai alapján foglalhatnak időpontot.
4. **Felhasználói hitelesítés**:
    - A felhasználók regisztrálhatnak és bejelentkezhetnek az alkalmazásba.
    - A hitelesített felhasználók hozzáférhetnek a személyre szabott funkciókhoz, például a kisállatok kezeléséhez és az időpontfoglaláshoz.
5. **Kapcsolatfelvétel**:
    - A felhasználók üzenetet küldhetnek az adminisztrátoroknak a kapcsolatfelvételi űrlapon keresztül.
6. **Galéria**:
    - A galéria oldalon a felhasználók megtekinthetik a szolgáltatások során készült képeket.

Technológiai Stack

Frontend

- **AngularJS**: Az alkalmazás fő keretrendszere.
- **UI-Router**: Az oldalak közötti navigáció kezelésére.
- **HTML/CSS**: A felhasználói felület kialakításához.
- **Bootstrap:** CSS keretrendszer, amely reszponzív és modern megjelenést biztosít.

Backend

- **PHP**: Az API-k és az adatbázis-kezelés megvalósításához.
- **MariaDB**: Az adatok tárolására szolgáló relációs adatbázis.
- **XAMPP**: A helyi fejlesztéshez szükséges Apache webszerver, PHP, és MariaDB (MySQL) adatbázis

Fejlesztési Útmutató

1. **Követelmények**:
    - PHP 8.2 vagy újabb verzió.
    - MariaDB adatbázis.
    - AngularJS 1.x.
2. **Telepítés**:
    - Importálja a pajti-paradicsom.sql fájlt az adatbázisba.
    - Állítsa be a PHP fájlokban az adatbázis kapcsolatot.
    - Nyissa meg az alkalmazást egy böngészőben.
3. **Fejlesztés**:
    - Az AngularJS vezérlők és szolgáltatások a app.js fájlban találhatók.
    - A PHP API-k a php mappában találhatók.
    - A HTML fájlok a html mappában találhatók.

index.html

Az index.html fájl a **Pajti Paradicsom** webalkalmazás belépési pontja. Ez az oldal tölti be az AngularJS alkalmazást, és biztosítja az alapvető struktúrát, amelyre az alkalmazás épül. Az index.html tartalmazza a szükséges könyvtárak, stíluslapok és JavaScript fájlok hivatkozásait, valamint az alkalmazás gyökér elemét.

Fő Funkciók

1. **Alapvető HTML Struktúra**:
    - Meghatározza az oldal alapvető HTML szerkezetét, beleértve a fejlécet, a metaadatokat és a gyökérelemet.
2. **AngularJS Alkalmazás Betöltése**:
    - Az AngularJS keretrendszer inicializálása az ng-app attribútum segítségével.
3. **Külső Könyvtárak és Stíluslapok Betöltése**:
    - Tartalmazza az alkalmazás működéséhez szükséges CSS és JavaScript fájlokat.
4. **Gyökérelem Meghatározása**:
    - Az AngularJS alkalmazás gyökéreleme, amelyen belül az összes dinamikus tartalom megjelenik.

HTML Felépítés

1. Fejléc (Head)

A fejléc tartalmazza az oldal metaadatait, a stíluslapok hivatkozásait és az oldal címét.

Fontos Elemei:

- **Metaadatok**:
    - charset="UTF-8": Az oldal karakterkódolása.
    - viewport: A reszponzív megjelenés biztosítása mobil eszközökön.
- **Cím**:
    - Az oldal címe, amely a böngésző fülén jelenik meg (pl. *"Pajti Paradicsom"*).
- **CSS Fájlok**:
    - Hivatkozások az alkalmazás stíluslapjaira (pl. Bootstrap, egyedi CSS).

2. Törzs (Body)

A törzs tartalmazza az AngularJS alkalmazás gyökérelemét és a szükséges JavaScript fájlok hivatkozásait.

Fontos Elemei:

- **AngularJS Inicializálás**:
  - Az ng-app="app" attribútum inicializálja az AngularJS alkalmazást.
- **Gyökérelem**:
  - Az AngularJS alkalmazás tartalma a <div ui-view> elemben jelenik meg, amely az ui-router által kezelt dinamikus tartalom helye.
- **JavaScript Fájlok**:
  - Hivatkozások az AngularJS, az alkalmazás saját JavaScript fájljaira és az egyéb szükséges könyvtárakra.

Betöltött Könyvtárak és Fájlok

CSS Fájlok

1. **bootstrap.min.css**:
   - A Bootstrap keretrendszer reszponzív stílusainak betöltése.
2. **style.css**:
   - Az alkalmazás egyedi stílusainak meghatározása.

JavaScript Fájlok

1. **angular.min.js**:
   - Az AngularJS keretrendszer.
2. **angular-ui-router.min.js**:
   - Az ui-router könyvtár az oldalak közötti navigáció kezeléséhez.
3. **app.js**:
   - Az AngularJS alkalmazás fő fájlja, amely tartalmazza a vezérlőket, szolgáltatásokat és konfigurációkat.

Főbb Funkcionális Elemei

1. **Reszponzív Megjelenés**:
    - A Bootstrap és az egyedi CSS biztosítja, hogy az oldal minden eszközön megfelelően jelenjen meg.
2. **AngularJS Alkalmazás Indítása**:
    - Az ng-app="app" attribútum inicializálja az AngularJS alkalmazást.
3. **Dinamikus Tartalom**:
    - A <div ui-view> elem az ui-router által kezelt dinamikus tartalom helye.
    -

```
<!DOCTYPE html>
<html lang="hu">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="icon" type="image/png" sizes="16x16" href="./media/image/favicon.png">
    <title>Pajti-paradicsom</title>

    <!-- Application components -->
  <link rel="stylesheet" href="../components/bootstrap/5.3.3/css/bootstrap.min.css">
    <link rel="stylesheet" href="../components/font-awesome/6.5.2/css/all.min.css">
  <link rel="stylesheet" href="../common/css/app-common.css">
  <link rel="stylesheet" href="./css/app.css">
</head>
<body ng-app="app">

    <!-- Application container -->
    <ui-view class="app-container"></ui-view>

    <!-- Application components -->
    <script src="../components/jquery/3.7.1/js/jquery.min.js"></script>
    <script src="../components/angular-js/1.8.2/js/angular.min.js"></script>
    <script src="../components/angular-js/angular-ui-router/1.0.30/js/angular-ui-router.min.js"></script>
    <script src="../components/moment/2.30.1/js/moment-with-locales.min.js"></script>
    <script src="../components/bootstrap/5.3.3/js/bootstrap.bundle.min.js"></script>
    <script src="../common/js/app-common.js"></script>
    <script src="./js/app.js"></script>
</body>
</html>      You, yesterday • KOMMENT …
```

Adatbázis Struktúra

Az adatbázis a következő fő táblákat tartalmazza:

users: A regisztrált felhasználók adatait tárolja.

Oszlopok: id, username, email, password, phone.

pets: A felhasználók által regisztrált kisállatok adatait tárolja.

Oszlopok: id, user_id, name, type, age, description.

services: A különböző szolgáltatások leírását tartalmazza.

Oszlopok: id, name, description, img.

prices: A szolgáltatások árait tartalmazza.

Oszlopok: id, service_id, type, price.

schedule: Az időpontfoglalások adatait tárolja.

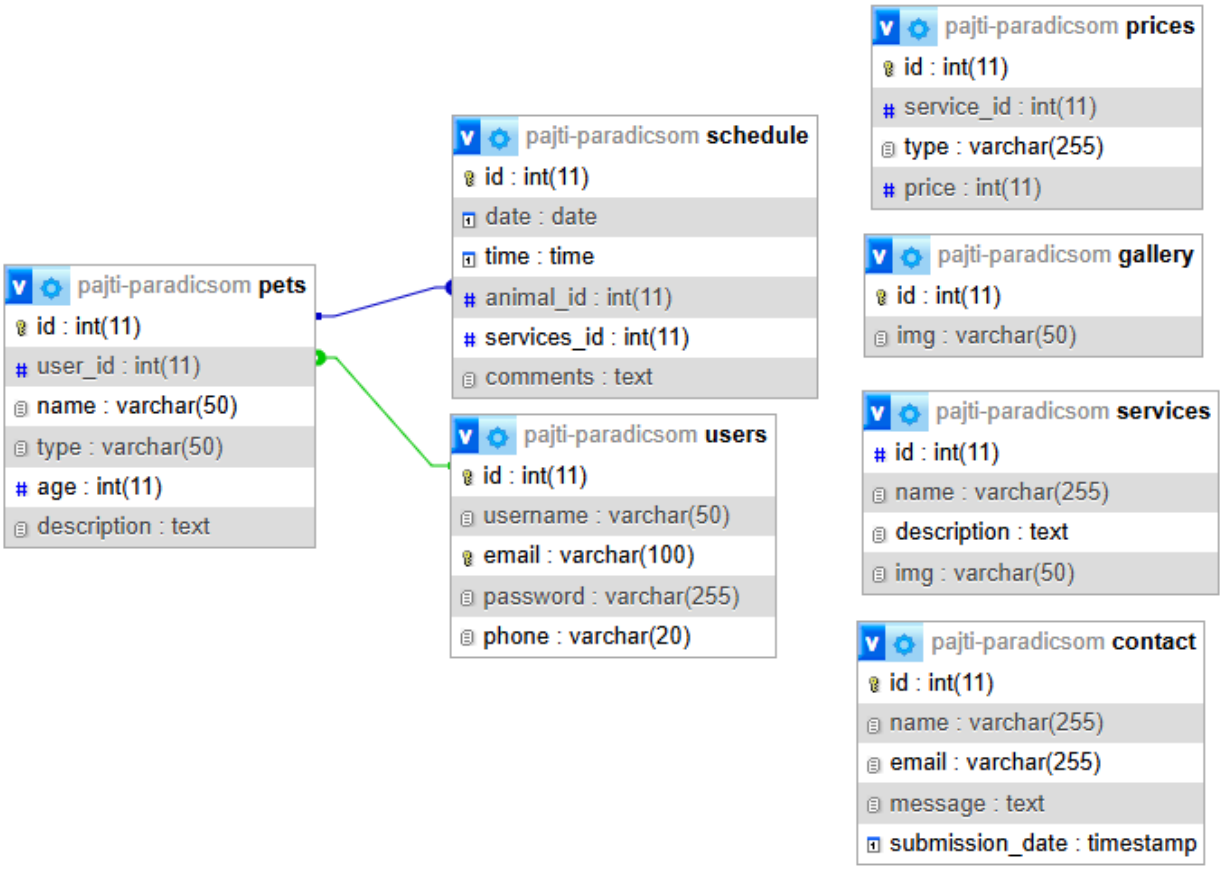
Oszlopok: id, date, time, animal_id, services_id, comments.

contact: A kapcsolatfelvételi űrlapon keresztül küldött üzeneteket tárolja.

Oszlopok: id, name, email, message, submission_date.

gallery: A galériában megjelenített képek adatait tartalmazza.

Oszlopok: id, img.



HTML Oldalak

A projekt HTML fájljai funkciójuk alapján csoportosíthatók, és mindegyik egy adott célt szolgál az alkalmazás felépítésében és működésében.

1. Főoldalak

Ezek az oldalak a weboldal fő tartalmát és funkcióit biztosítják a felhasználók számára:

home.html: A kezdőlap, amely bemutatja a szolgáltatásokat és a panziót. Ez az első oldal, amit a látogatók meglátnak.

services.html: A szolgáltatások részletes leírását tartalmazza, hogy a felhasználók megismerhessék az elérhető lehetőségeket.

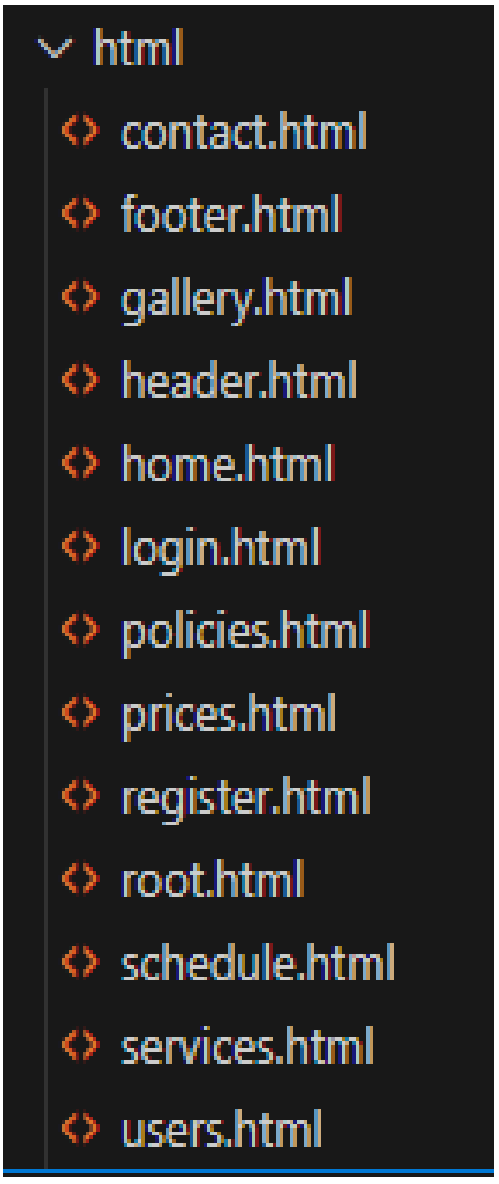
schedule.html: Az időpontfoglalási oldal, ahol a felhasználók kiválaszthatják a kívánt szolgáltatást és időpontot.

prices.html: Az árak megtekintésére szolgáló oldal, amely segít a felhasználóknak tájékozódni a költségekről.

gallery.html: A galéria oldal, amely vizuális tartalmat nyújt a látogatóknak, például képeket a panzióról vagy a szolgáltatásokról.

contact.html: Kapcsolatfelvételi oldal, amely lehetőséget biztosít az üzenetek küldésére és a kapcsolatfelvételre.

policies.html: Szabályzatok és feltételek oldala, amely tartalmazza a GY.I.K.-et, adatvédelmi irányelveket és egyéb fontos információkat.



2. Felhasználói műveletekhez kapcsolódó oldalak

Ezek az oldalak a felhasználók regisztrációját, bejelentkezését és profiljuk kezelését szolgálják:

login.html: A bejelentkezési oldal, ahol a meglévő felhasználók hozzáférhetnek fiókjukhoz.
register.html: A regisztrációs oldal, amely lehetővé teszi új felhasználók számára a fiók létrehozását.
users.html: A felhasználói profil oldal, ahol a felhasználók kezelhetik adataikat, regisztrált kisállataikat és időpontfoglalásaikat.

3. Közös komponensek

Ezek a fájlok az oldalakon ismétlődő elemeket tartalmazzák, amelyeket az alkalmazás dinamikusan betölt:

header.html: Az oldal tetején található navigációs sáv és logó, amely az összes oldal közös része.
footer.html: Az oldal alján található lábléc, amely tartalmazza a kapcsolatfelvételi információkat, közösségi média linkeket és egyéb fontos információkat.

A közös komponensek biztosítják az egységes megjelenést és a könnyebb karbantarthatóságot az oldalak között.

5. Alkalmazás gyökérfájl

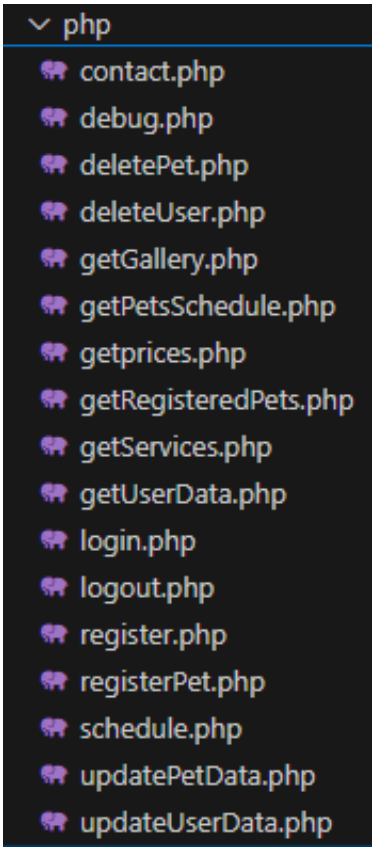
root.html: Az AngularJS alapú alkalmazás gyökérfájlja, amely az oldalak betöltését és az alkalmazás struktúráját kezeli.
A gyökérfájl az AngularJS keretrendszer működését biztosítja, és az oldalak közötti navigációt kezeli.

Ezek a fájlok együtt biztosítják a weboldal teljes funkcionalitását, a felhasználói interakciókat és az egységes megjelenést.

Backend API-k (PHP)

A PHP fájlok különböző API végpontokat valósítanak meg:

1. getPetsSchedule.php: A felhasználó regisztrált kisállatainak lekérdezése.
2. getprices.php: Az elérhető szolgáltatások árainak lekérdezése.
3. schedule.php: Új időpontfoglalás rögzítése.
4. getServices.php: A szolgáltatások listájának lekérdezése.
5. getUserData.php: Felhasználói adatok lekérdezése.
6. register.php: Új felhasználó regisztrálása.
7. login.php: Felhasználói bejelentkezés.
8. logout.php: Felhasználói kijelentkezés.
9. contact.php: Kapcsolatfelvételi üzenet rögzítése.
10. debug.php: Tesztelési célokra használt fájl.
11. deletePet.php: Egy regisztrált kisállat törlése.
12. deleteUser.php: Felhasználói fiók törlése.
13. getGallery.php: A galéria képeinek lekérdezése.
14. getRegisteredPets.php: A felhasználó regisztrált kisállatainak és időpontjaiknak lekérdezése.
15. registerPet.php: Új kisállat regisztrálása.
16. updatePetData.php: Egy regisztrált kisállat adatainak frissítése.
17. updateUserData.php: A felhasználói adatok frissítése.



Backend API-k Részletes Dokumentációja

Az alábbiakban részletesen bemutatom a **Pajti Paradicsom** projekt PHP API végpontjait, azok funkcióit, bemeneti paramétereit és működését.

1. contact.php
   - Funkció: Kapcsolatfelvételi üzenet rögzítése.
   - Bemeneti paraméterek:
     - name: A felhasználó neve.
     - email: A felhasználó e-mail címe.
     - message: Az üzenet szövege.
   - Működés:
     - Az adatokat beszúrja a contact adatbázis-táblába.
     - Siker esetén visszaad egy megerősítő üzenetet.
     - Hiba esetén hibát jelez.

2. debug.php
   - Funkció: Tesztelési célokra használt fájl.
   - Működés:
     - Tesztadatokat állít be a POST kéréshez.
     - Meghívja a schedule.php fájlt a tesztadatokkal.

3. deletePet.php
   - Funkció: Egy regisztrált kisállat törlése.
   - Bemeneti paraméterek:
     - pet_id: A törlendő kisállat azonosítója.
   - Működés:
     - Törli a megadott azonosítójú kisállatot a pets táblából.
     - Siker esetén visszaad egy megerősítő üzenetet.
     - Hiba esetén hibát jelez.

4. deleteUser.php
   - Funkció: Felhasználói fiók törlése.
   - Bemeneti paraméterek:
     - user_id: A törlendő felhasználó azonosítója.
   - Működés:
     - Törli a megadott azonosítójú felhasználót a users táblából.
     - Siker esetén visszaad egy megerősítő üzenetet.
     - Hiba esetén hibát jelez.

5. getGallery.php
   - Funkció: A galéria képeinek lekérdezése.
   - Bemeneti paraméterek: Nincs.
   - Működés:
     - Lekérdezi a gallery táblából a képek adatait.
     - Véletlenszerű sorrendbe állítja az eredményeket.
     - Az adatokat JSON formátumban adja vissza.

6. getPetsSchedule.php
   - Funkció: A felhasználó regisztrált kisállatainak lekérdezése.
   - Bemeneti paraméterek:
     - user_id: A felhasználó azonosítója.
   - Működés:
     - Lekérdezi a pets táblából a megadott felhasználóhoz tartozó kisállatokat.
     - Az adatokat JSON formátumban adja vissza.

7. getprices.php
   - Funkció: Az elérhető szolgáltatások árainak lekérdezése.
   - Bemeneti paraméterek: Nincs.
   - Működés:
     - Lekérdezi a prices és services táblákból az árakat és a szolgáltatások nevét.
     - Az adatokat JSON formátumban adja vissza.

8. getRegisteredPets.php
   - Funkció: A felhasználó regisztrált kisállatainak és időpontjaiknak lekérdezése.
   - Bemeneti paraméterek:
     - user_id: A felhasználó azonosítója.
   - Működés:
     - Lekérdezi a pets táblából a kisállatokat és a hozzájuk tartozó jövőbeli időpontokat.
     - Az adatokat JSON formátumban adja vissza.

9. getServices.php
   - Funkció: Az elérhető szolgáltatások listájának lekérdezése.
   - Bemeneti paraméterek: Nincs.
   - Működés:
     - Lekérdezi a services táblából a szolgáltatások nevét, leírását és képét.
     - Az adatokat JSON formátumban adja vissza.

10. getUserData.php
    - Funkció: A bejelentkezett felhasználó adatainak lekérdezése.
    - Bemeneti paraméterek:
      - user_id: A felhasználó azonosítója.
    - Működés:
      - Lekérdezi a users táblából a felhasználó adatait (név, e-mail, telefonszám).
      - Az adatokat JSON formátumban adja vissza.

11. login.php
    - Funkció: Felhasználói bejelentkezés.
    - Bemeneti paraméterek:
      - username: A felhasználó neve.
      - password: A felhasználó jelszava.
    - Működés:
      - Ellenőrzi a megadott felhasználónévhez tartozó jelszót.
      - Siker esetén visszaadja a felhasználó azonosítóját.
      - Hiba esetén hibát jelez.

12. logout.php
    - Funkció: Felhasználói kijelentkezés.
    - Bemeneti paraméterek: Nincs.
    - Működés:
        - Megszünteti a felhasználó munkamenetét (session).
        - Siker esetén visszaad egy megerősítő üzenetet.

13. register.php
    - Funkció: Új felhasználó regisztrálása.
    - Bemeneti paraméterek:
        - username: A felhasználó neve.
        - email: A felhasználó e-mail címe.
        - password: A felhasználó jelszava.
        - phone: A felhasználó telefonszáma.
    - Működés:
        - Ellenőrzi az e-mail cím egyediségét.
        - Beszúrja az új felhasználót a users táblába.
        - Siker esetén visszaadja az új felhasználó azonosítóját.

14. registerPet.php
    - Funkció: Új kisállat regisztrálása.
    - Bemeneti paraméterek:
        - user_id: A felhasználó azonosítója.
        - pet_name: A kisállat neve.
        - pet_type: A kisállat típusa.
        - pet_age: A kisállat életkora.
        - description: Opcionális leírás.
    - Működés:
        - Beszúrja az új kisállatot a pets táblába.
        - Siker esetén visszaadja az új kisállat azonosítóját.

15. schedule.php
    - Funkció: Új időpontfoglalás rögzítése.
    - Bemeneti paraméterek:
        - services_id: A kiválasztott szolgáltatás azonosítója.
        - animal_id: A kisállat azonosítója.
        - date: A foglalás dátuma.
        - time: A foglalás időpontja.
        - comments: Opcionális megjegyzések.
    - Működés:
        - Ellenőrzi, hogy az időpont szabad-e.
        - Beszúrja az új foglalást a schedule táblába.
        - Siker esetén visszaadja a foglalás adatait.

16. updatePetData.php
    - Funkció: Egy regisztrált kisállat adatainak frissítése.
    - Bemeneti paraméterek:
        - id: A kisállat azonosítója.
        - name: A kisállat neve.
        - age: A kisállat életkora.
        - description: Opcionális leírás.
    - Működés:
        - Frissíti a megadott kisállat adatait a pets táblában.
        - Siker esetén visszaad egy megerősítő üzenetet.

17. updateUserData.php
    - Funkció: A felhasználói adatok frissítése.
    - Bemeneti paraméterek:
        - id: A felhasználó azonosítója.
        - email: Az új e-mail cím.
        - phone: Az új telefonszám.
    - Működés:
        - Frissíti a megadott felhasználó adatait a users táblában.
        - Siker esetén visszaad egy megerősítő üzenetet.

Az app.js Fájl Felépítése

Modul Definíció

Az alkalmazás AngularJS modulja az app néven van definiálva, és a következő függőségeket használja:

- **ui.router**: Az oldalak közötti navigáció kezelésére.
- **app.common**: Egyedi, közös funkciókat tartalmazó modul.

angular.module("app", ["ui.router", "app.common"]);

Konfiguráció (config)

Az alkalmazás konfigurációját a $stateProvider és $urlRouterProvider segítségével valósítja meg. Az állapotok (oldalak) definiálása a következőképpen történik:

```
// Application config
.config([
  "$stateProvider",
  "$urlRouterProvider",
  function ($stateProvider, $urlRouterProvider) {
    $stateProvider
      .state("root", {
        views: {
          "": {
            templateUrl: "./html/root.html",
          },
          "header@root": {
            templateUrl: "./html/header.html",
            controller: "headerController",
          },
          "footer@root": {
            templateUrl: "./html/footer.html",
          },
        },
      })
```

Állapotok

1. root: Az alapállapot, amely tartalmazza a fejlécet, láblécet és az alapvető elrendezést.
2. home: A kezdőlap, amely bemutatja az alkalmazás fő funkcióit.
3. services: A szolgáltatások oldala, ahol a felhasználók megtekinthetik a kínált szolgáltatásokat.
4. schedule: Az időpontfoglalási oldal, ahol a felhasználók időpontot foglalhatnak.
5. prices: Az árak oldala, ahol a szolgáltatások árai jelennek meg.
6. gallery: A galéria oldal, ahol a szolgáltatások során készült képek láthatók.
7. contact: A kapcsolatfelvételi oldal, ahol a felhasználók üzenetet küldhetnek.
8. policies: Az alkalmazás szabályzatait és feltételeit tartalmazó oldal.
9. login: A bejelentkezési oldal, ahol a felhasználók hozzáférhetnek a fiókjukhoz.
10. register: A regisztrációs oldal, ahol új felhasználók hozhatnak létre fiókot.
11. users: A felhasználói profil oldal, ahol a felhasználók kezelhetik adataikat és regisztrált kisállataikat.

Alapértelmezett Útvonal

Ha az URL nem egyezik egyetlen állapottal sem, az alkalmazás a kezdőlapra (/) irányítja a felhasználót:
$urlRouterProvider.otherwise("/");

Futásidejű Konfiguráció (run)

A run blokk inicializálja az alkalmazást, és eseményeket kezel.

```
91  .run([
92    "trans",
93    function (trans) {
94      trans.events(["users"]);
95    },
96  ])
```

Szolgáltatások

Az authService szolgáltatás kezeli a felhasználói hitelesítést:

- isLoggedIn: Ellenőrzi, hogy a felhasználó be van-e jelentkezve.
- login: Bejelentkezéskor beállítja a felhasználói azonosítót.
- logout: Kijelentkezteti a felhasználót, és visszairányítja a kezdőlapra.
- getUserId: Visszaadja a bejelentkezett felhasználó azonosítóját.

Fontos AngularJS Vezérlők

scheduleController

A **scheduleController** az időpontfoglalási funkciók kezeléséért felelős. Ez a vezérlő biztosítja, hogy a felhasználók kiválaszthassák a szolgáltatásokat, szűrhessék az al-szolgáltatásokat, és időpontot foglalhassanak.

Fő Funkciók

1. Betölti a regisztrált kisállatokat a felhasználó számára.
2. Betölti az elérhető szolgáltatások árait.
3. Szűri az árakat a kiválasztott szolgáltatás típus alapján.
4. Kezeli az időpontfoglalási űrlap elküldését.

Fontos Változók

- $scope.scheduleData: Az időpontfoglalási űrlap adatait tartalmazza (szolgáltatás típusa, al-szolgáltatás, kisállat, dátum, időpont, megjegyzések).
- $scope.pets: A felhasználó által regisztrált kisállatok listája.
- $scope.allPrices: Az összes elérhető szolgáltatás ára.
- $scope.filteredPrices: A kiválasztott szolgáltatás típus alapján szűrt árak.

Fontos Metódusok

1. $scope.loadRegisteredPets():
    - Betölti a felhasználó regisztrált kisállatait a getPetsSchedule.php API-n keresztül.
    - Hiba esetén figyelmeztetést jelenít meg.
2. $scope.loadAllPrices():
    - Betölti az összes elérhető szolgáltatás árait a getprices.php API-n keresztül.

    - Hiba esetén figyelmeztetést jelenít meg.
3. $scope.loadSubServices():
    - Szűri az árakat a kiválasztott szolgáltatás típus alapján.
    - Ha nincs kiválasztott szolgáltatás, törli a szűrt árakat.
4. $scope.submitAppointment():
    - Ellenőrzi, hogy a felhasználó megadott-e érvényes dátumot és időpontot.
    - Elküldi az időpontfoglalási adatokat a schedule.php API-nak.
    - Sikeres foglalás esetén alaphelyzetbe állítja az űrlapot, és visszajelzést ad.

Hibakezelés

- Hiba esetén a vezérlő konzolra írja a hibát, és figyelmeztetést jelenít meg a felhasználónak.

Példa Munkafolyamat a scheduleController-ben

1. **Adatok Betöltése**:
    - A vezérlő inicializálásakor betöltődnek a regisztrált kisállatok és a szolgáltatások árai.
2. **Szolgáltatás Kiválasztása**:
    - A felhasználó kiválaszt egy szolgáltatást, amely szűri az elérhető al-szolgáltatásokat.
3. **Űrlap Kitöltése**:
    - A felhasználó kiválaszt egy kisállatot, dátumot, időpontot, és opcionálisan megjegyzést ad hozzá.
4. **Időpontfoglalás Elküldése**:
    - Az űrlap elküldése után a rendszer feldolgozza a foglalást.
5. **Siker/Hiba Kezelése**:
    - Siker esetén az űrlap alaphelyzetbe áll, és üzenet jelenik meg.
    - Hiba esetén figyelmeztetés jelenik meg.

registerController

A **registerController** a regisztrációs folyamat kezeléséért felelős. Ez a vezérlő biztosítja, hogy az új felhasználók regisztrálhassanak az alkalmazásba, és az adataik megfelelően elmentésre kerüljenek az adatbázisba.

Fő Funkciók

1. Kezeli a regisztrációs űrlap adatait.
2. Ellenőrzi, hogy a megadott adatok érvényesek-e.
3. Elküldi a regisztrációs adatokat a backendnek (register.php API).
4. Sikeres regisztráció esetén átirányítja a felhasználót a bejelentkezési oldalra.

Fontos Változók

- **$scope.registerData**: A regisztrációs űrlap adatait tartalmazza (pl. felhasználónév, e-mail cím, jelszó, telefonszám).
- **$scope.errorMessage**: Hibaüzenetek megjelenítésére szolgál, ha a regisztráció sikertelen.

Fontos Metódusok

$scope.submitRegistration()

- **Funkció**: Elküldi a regisztrációs adatokat a register.php API-nak.
- **Működés**:
    1. Ellenőrzi, hogy a kötelező mezők (felhasználónév, e-mail, jelszó) ki vannak-e töltve.
    2. Elküldi az adatokat a backendnek.
    3. Sikeres regisztráció esetén átirányítja a felhasználót a bejelentkezési oldalra.
    4. Hiba esetén figyelmeztetést jelenít meg.

Hibakezelés

- **Hiányzó vagy érvénytelen adatok**:
    - Ha a kötelező mezők nincsenek kitöltve, a vezérlő figyelmeztetést jelenít meg.
- **Backend hiba**:
    - Ha a register.php API hibát küld vissza (pl. e-mail cím már létezik), a vezérlő megjeleníti a hibaüzenetet.

Példa Munkafolyamat a registerController-ben

1. **Űrlap Kitöltése**:
    - A felhasználó kitölti a regisztrációs űrlapot (pl. név, e-mail, jelszó).
2. **Adatok Ellenőrzése**:
    - A vezérlő ellenőrzi, hogy minden kötelező mező ki van-e töltve.
3. **Adatok Elküldése**:
    - Az űrlap adatait elküldi a register.php API-nak.
4. **Siker/Hiba Kezelése**:
    - Siker esetén a felhasználó átirányításra kerül a bejelentkezési oldalra.
    - Hiba esetén figyelmeztetés jelenik meg.

loginController

A **loginController** a bejelentkezési folyamat kezeléséért felelős. Ez a vezérlő biztosítja, hogy a regisztrált felhasználók bejelentkezhessenek az alkalmazásba, és hozzáférjenek a személyre szabott funkciókhoz.

Fő Funkciók

1. Kezeli a bejelentkezési űrlap adatait.
2. Ellenőrzi, hogy a megadott adatok érvényesek-e.
3. Elküldi a bejelentkezési adatokat a backendnek (login.php API).
4. Sikeres bejelentkezés esetén átirányítja a felhasználót a főoldalra vagy a felhasználói profil oldalra.

Fontos Változók

- **$scope.loginData**: A bejelentkezési űrlap adatait tartalmazza (pl. e-mail cím, jelszó).
- **$scope.errorMessage**: Hibaüzenetek megjelenítésére szolgál, ha a bejelentkezés sikertelen.

Fontos Metódusok

$scope.submitLogin()

- **Funkció**: Elküldi a bejelentkezési adatokat a login.php API-nak.
- **Működés**:
    1. Ellenőrzi, hogy a kötelező mezők (e-mail, jelszó) ki vannak-e töltve.
    2. Elküldi az adatokat a backendnek.
    3. Sikeres bejelentkezés esetén elmenti a felhasználói munkamenetet (session) és átirányítja a felhasználót.
    4. Hiba esetén figyelmeztetést jelenít meg.

Hibakezelés

- **Hiányzó vagy érvénytelen adatok**:
    - Ha a kötelező mezők nincsenek kitöltve, a vezérlő figyelmeztetést jelenít meg.
- **Helytelen e-mail vagy jelszó**:
    - Ha a login.php API hibát küld vissza (pl. helytelen e-mail vagy jelszó), a vezérlő megjeleníti a hibaüzenetet.
- **Backend hiba**:
    - Ha a szerver nem érhető el, a vezérlő általános hibaüzenetet jelenít meg.

Példa Munkafolyamat a loginController-ben

1. **Űrlap Kitöltése**:
    - A felhasználó kitölti a bejelentkezési űrlapot (pl. e-mail, jelszó).
2. **Adatok Ellenőrzése**:
    - A vezérlő ellenőrzi, hogy minden kötelező mező ki van-e töltve.
3. **Adatok Elküldése**:
    - Az űrlap adatait elküldi a login.php API-nak.
4. **Siker/Hiba Kezelése**:
    - Siker esetén a felhasználó átirányításra kerül a főoldalra vagy a profil oldalra.
    - Hiba esetén figyelmeztetés jelenik meg.

Tesztelési Stratégia

Cél

A tesztelés célja annak biztosítása, hogy az alkalmazás minden funkciója megfelelően működjön, és a felhasználói élmény zökkenőmentes legyen.

Tesztelési Típusok

1. **Funkcionális Tesztelés**:
    - Az alkalmazás funkcióinak ellenőrzése (pl. regisztráció, bejelentkezés, időpontfoglalás).
2. **Integrációs Tesztelés**:
    - Az API-k és a frontend közötti kommunikáció tesztelése.

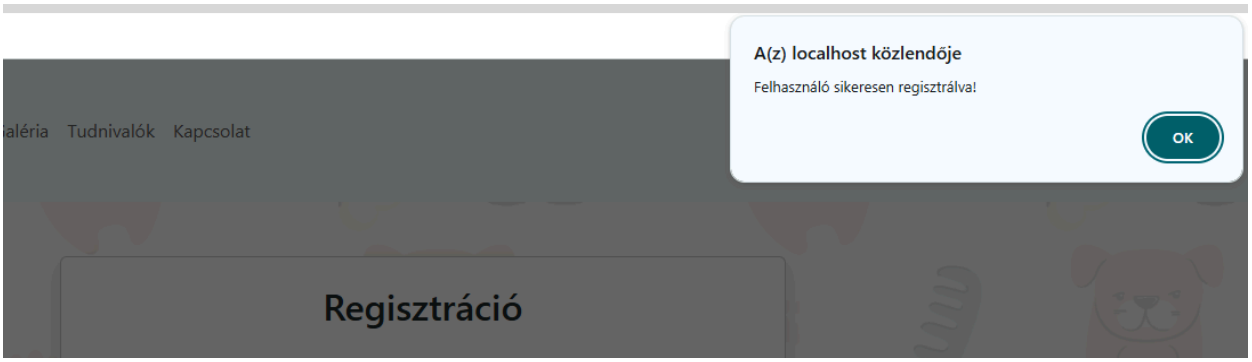
3. **Felhasználói Felület (UI) Tesztelés**:
    - Az oldalak megjelenésének és interakcióinak ellenőrzése.

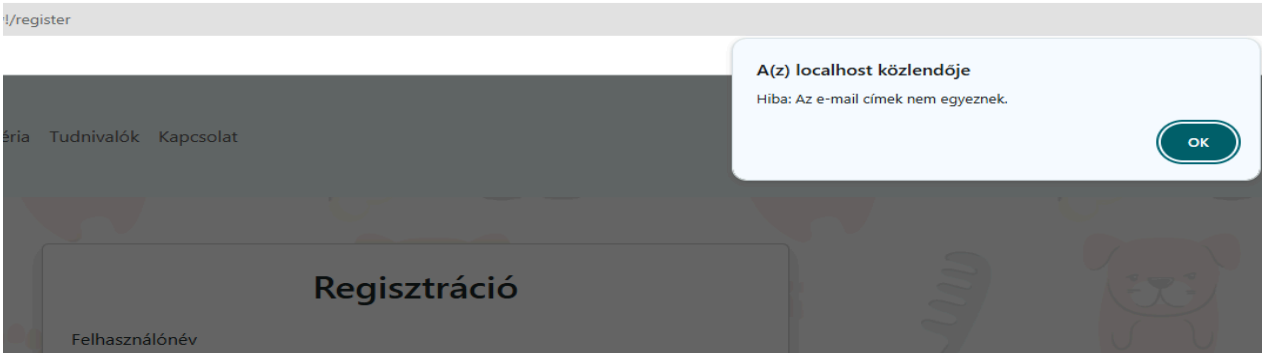
Tesztelési Forgatókönyvek

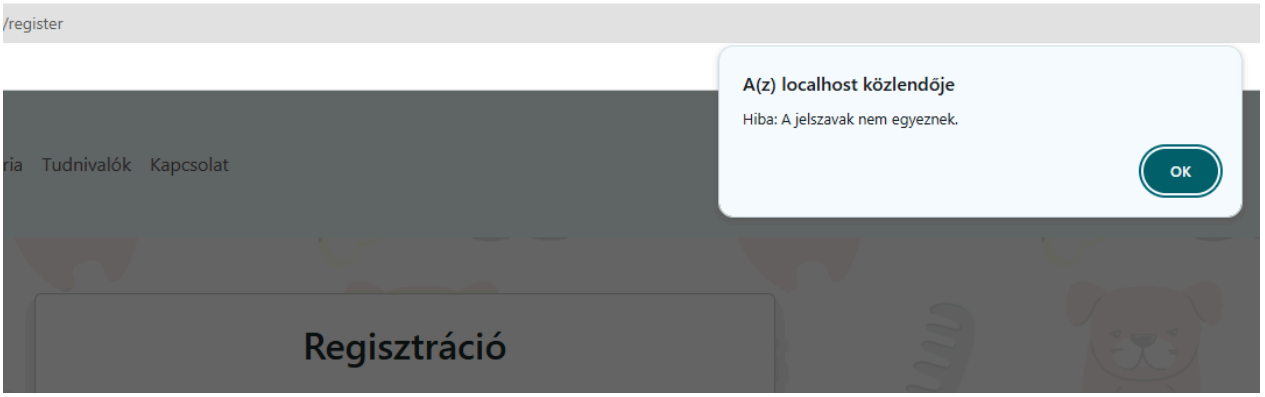
1. **Regisztráció Tesztelése**

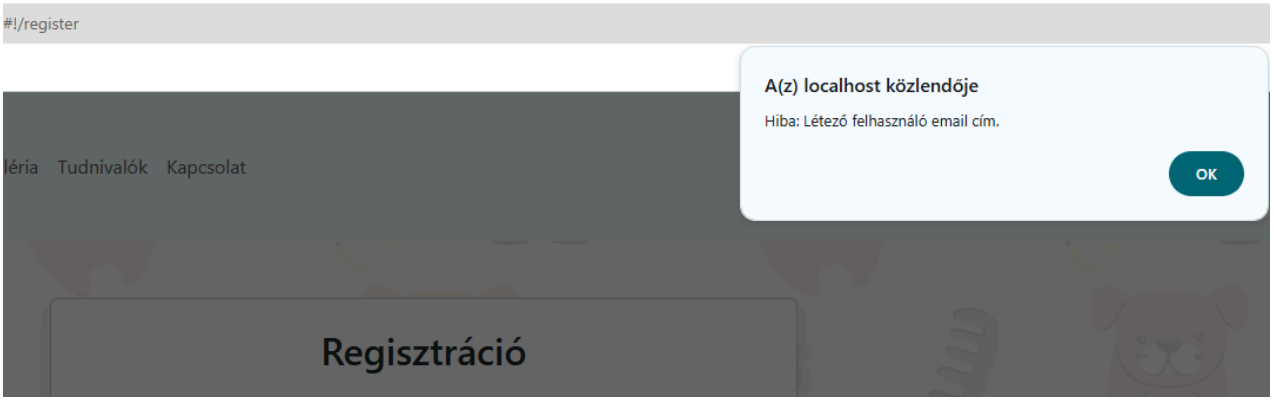
Forgatókönyvek:

1. **Sikeres regisztráció**:
    - Adatok: Érvényes felhasználónév, e-mail cím, jelszó.
    - Várható eredmény: A felhasználó sikeresen regisztrál, és átirányításra kerül a bejelentkezési oldalra.
2. **Nem egyező e-mail címek**/jelszavak
    - Adatok: Hibás ellenőrző e-mail/jelszó bevitel
    - Várható eredmény: Hibaüzenet jelenik meg: *"Az e-mail címek nem egyeznek" "A jelszavak nem egyeznek"*
3. **Már létező e-mail cím**:
    - Adatok: Olyan e-mail cím, amely már regisztrálva van.
    - Várható eredmény: Hibaüzenet jelenik meg: *"Létező felhasználó email cím."*


aléria Tudnivalók Kapcsolat
A(z) localhost közlendője
Felhasználó sikeresen regisztrálva!
OK
Regisztráció


!/register
éria Tudnivalók Kapcsolat
A(z) localhost közlendője
Hiba: Az e-mail címek nem egyeznek.
OK
Regisztráció
Felhasználónév

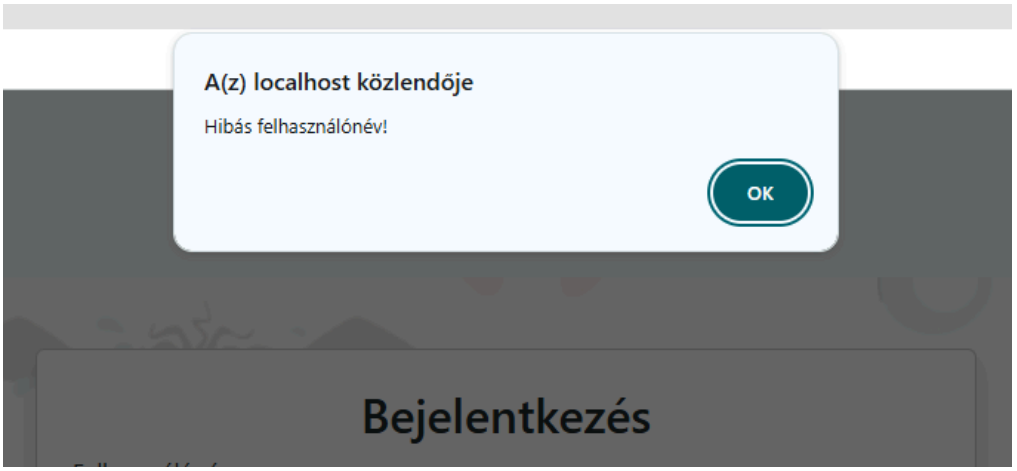

/register
ria Tudnivalók Kapcsolat
A(z) localhost közlendője
Hiba: A jelszavak nem egyeznek.
OK
Regisztráció

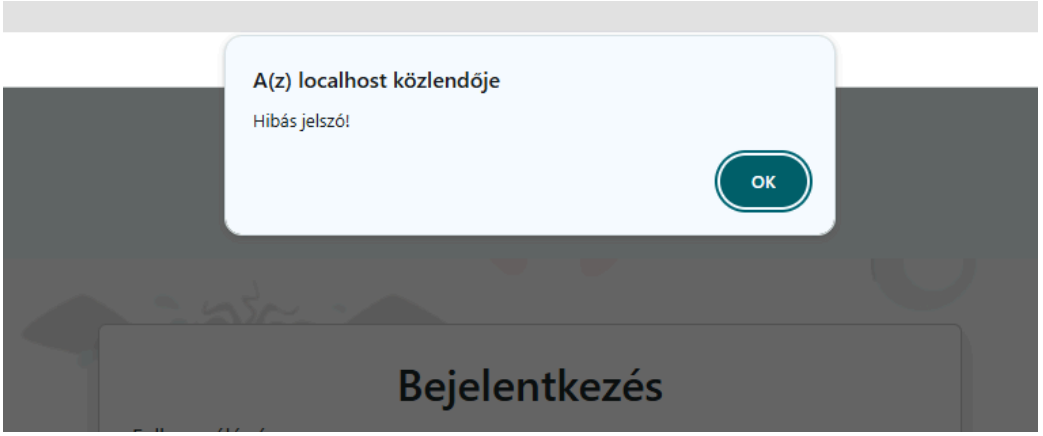

#!/register
léria Tudnivalók Kapcsolat
A(z) localhost közlendője
Hiba: Létező felhasználó email cím.
OK
Regisztráció

2. Bejelentkezés Tesztelése

Forgatókönyvek:

1. **Sikeres bejelentkezés**:
    - Adatok: Érvényes felhasználónév és jelszó.
    - Várható eredmény: A felhasználó sikeresen bejelentkezik, és átirányításra kerül a főoldalra.
2. **Helytelen felhasználónév**:
    - Adatok: Érvényes jelszó, de hibás felhasználónév
    - Várható eredmény: Hibaüzenet jelenik meg: *"Helytelen felhasználónév vagy jelszó."*
3. **Helytelen jelszó**
    - Adatok: Érvényes felhasználónév, de hibás jelszó
    - Várható eredmény: Hibaüzenet jelenik meg: "Helytelen felhasználónév vagy jelszó."

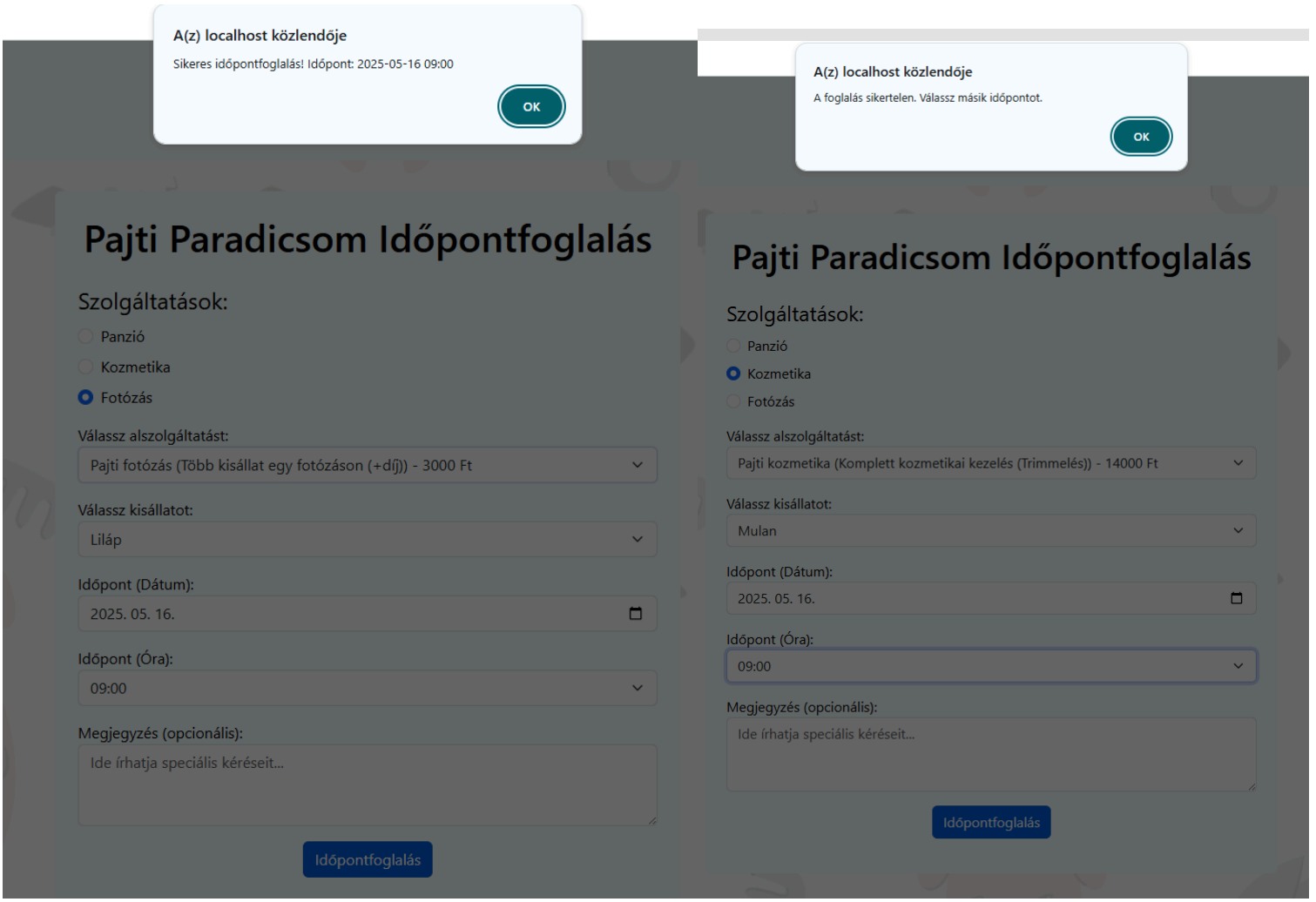




3. Időpontfoglalás Tesztelése

Forgatókönyvek:

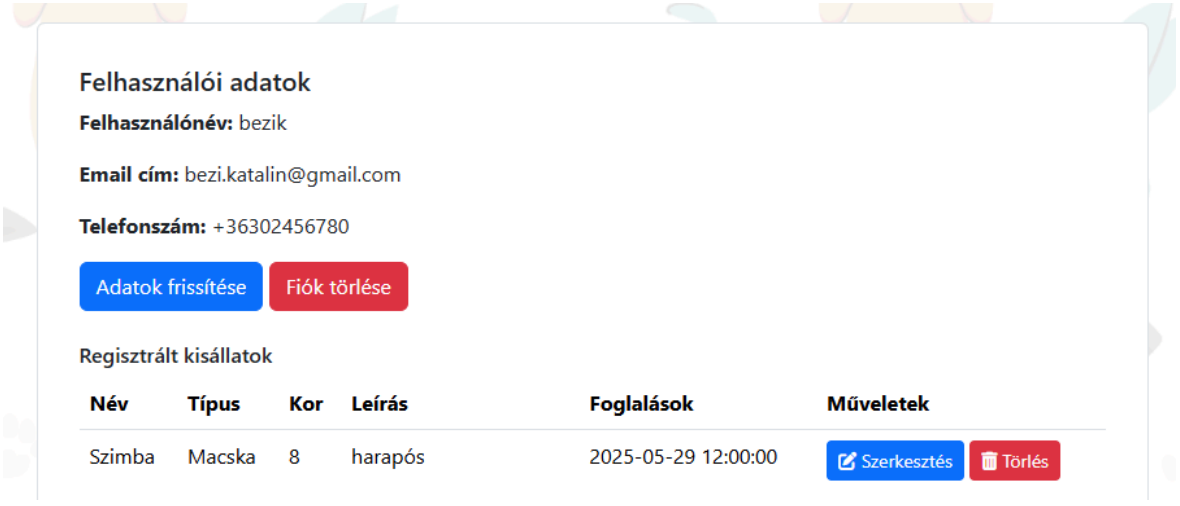
1. **Sikeres időpontfoglalás**:
    - Adatok: Érvényes kisállat, szolgáltatás, dátum és időpont.
    - Várható eredmény: Az időpont sikeresen rögzítésre kerül, és visszajelzés jelenik meg: *"Sikeres időpontfoglalás. Időpont:2025-02-04 9:00 "*
2. **Foglalás ütközés**:
    - Adatok: Már foglalt időpont.
    - Várható eredmény: Hibaüzenet jelenik meg: *"A foglalás sikertelen. Válasszon másik időpontot."*



4. Felhasználói Profil Tesztelése

Forgatókönyvek:

1. **Felhasználói adatok betöltése**:
    - Adatok: Bejelentkezett felhasználó.
    - Várható eredmény: A felhasználói adatok sikeresen betöltődnek.
2. **Felhasználói adatok frissítése:**
    - Adatok: Létező felhasználó azonosító.
    - Várható eredmény: A felhasználói adatok törlődnek. Átirányítás a kezdőlapra.
3. **Felhasználói fiók törlése:**
    - Adatok: Bejelentkezett felhasználó adatai (telefonszám, e-mail).
    - Várható eredmény: A felhasználói adatok sikeresen módosulnak.
4. **Kisállat hozzáadása**:
    - Adatok: Érvényes kisállat adatok (név, típus, kor,leírás).
    - Várható eredmény: A kisállat sikeresen hozzáadásra kerül, és megjelenik a listában.
5. **Kisállat törlése**:
    - Adatok: Létező kisállat azonosító.
    - Várható eredmény: A kisállat sikeresen törlésre kerül, és eltűnik a listából.
6. **Kisállat adatainak frissítése**:
    - Adatok: Érvényes kisállat adatok (név, kor, leírás).
    - Várható eredmény: A kisállat sikeresen frissítésre kerül, és megjelenik a listában.



5. Kapcsolatfelvétel Tesztelése

Forgatókönyvek:

1. **Sikeres üzenetküldés**:
    - Adatok: Érvényes név, e-mail cím és üzenet.
    - Várható eredmény: Az üzenet sikeresen elküldésre kerül, és visszajelzés jelenik meg: *"Üzenetét sikeresen elküldtük."*
2. **Hiányzó mezők**:
    - Adatok: Hiányzó név vagy üzenet.
    - Várható eredmény: Hibaüzenet jelenik meg
3. **Email formátum**
    1. Adatok: Nem megfelelő email formátum
    2. Várható eredmény: Hibaüzenet jelenik meg

Fejlesztési lehetőségek

1. Felhasználói élmény javítása

- Dátum- és időválasztó fejlesztése: A dátum- és időpontválasztó mezők helyett modern, felhasználóbarát naptár- és időválasztó komponensek (pl. Angular Material Datepicker) használata.
- Betöltési animációk: Adatok betöltésekor (pl. kisállatok, árak) betöltési animációk vagy "loading" indikátorok megjelenítése.

2. Backend kommunikáció optimalizálása

- Hibakezelés javítása: A backendről érkező hibák részletesebb kezelése és felhasználóbarát üzenetek megjelenítése.

3. Biztonság növelése

- Adatvédelem: A felhasználói adatok (pl. jelszavak, foglalási adatok) titkosítása HTTPS-en keresztül.
- Bejelentkezési védelem: Túl sok sikertelen bejelentkezési kísérlet esetén ideiglenes zárolás.

4. Új funkciók bevezetése

- Foglalások kezelése: A felhasználók számára lehetőség a meglévő foglalások megtekintésére, módosítására vagy törlésére.
- Értesítések: E-mail vagy SMS értesítések küldése a foglalásokról és emlékeztetők az időpontokról.
- Keresés és szűrés: Szolgáltatások, árak kereshetősége és szűrhetősége.

5. Adminisztrációs felület

- Admin panel: Egy adminisztrációs felület létrehozása, ahol a szolgáltatások, árak, foglalások és felhasználók kezelhetők.
- Statisztikák: Foglalási statisztikák megjelenítése az adminisztrátorok számára.

6. Többnyelvűség támogatása

- Nyelvváltás: A weboldal többnyelvűvé tétele (pl. magyar és angol), AngularJS fordítási modulok (pl. angular-translate) használatával.

7. Modernizáció

- AngularJS frissítése: A projekt migrálása egy modernebb Angular verzióra (pl. Angular 15+), mivel az AngularJS támogatása már megszűnt.

Összegzés

A projekt egy átgondolt és jól felépített webalkalmazás, amely a felhasználók és a szolgáltatások közötti interakciókat helyezi előtérbe. Az alkalmazás célja, hogy egyszerű és hatékony módot biztosítson a felhasználók számára a kisállataik kezelésére, időpontfoglalások rögzítésére, valamint a szolgáltatások és árak megismerésére. A különálló HTML, PHP és JavaScript fájlok logikus szétválasztása nemcsak a kód átláthatóságát segíti, hanem a karbantarthatóságot és a jövőbeli bővítések lehetőségét is megkönnyíti.

A backend API-k széles skálája gondoskodik arról, hogy a rendszer minden fontos funkciót lefedjen, legyen szó felhasználói regisztrációról, bejelentkezésről, kisállatok kezeléséről vagy időpontfoglalásokról. Az adatbázis-lekérdezések és a JSON formátumú válaszok biztosítják a gyors és hatékony adatkommunikációt a frontend és a backend között. A frontend oldalak pedig letisztultak és felhasználóbarátok, így a látogatók könnyen eligazodhatnak az oldalon, és gyorsan megtalálhatják a számukra fontos információkat.

A projekt erősségei közé tartozik a modularitás, amely lehetővé teszi az egyes komponensek újrafelhasználását, például a közös elemek, mint a fejléc és a lábléc, minden oldalon egységes megjelenést biztosítanak. Az AngularJS alapú struktúra dinamikus és reszponzív működést tesz lehetővé, ami különösen fontos a modern webalkalmazásoknál. Az időpontfoglalási rendszer például jól példázza, hogyan lehet a felhasználói élményt egyszerűsíteni, miközben a háttérben komplex adatkezelési folyamatok zajlanak.

Ugyanakkor a projekt további fejlesztési lehetőségeket is kínál. A biztonság terén például érdemes lenne bevezetni erősebb adatvédelmi mechanizmusokat, mint például a jelszavak titkosítása. A teljesítmény optimalizálása érdekében a képek tömörítése, a gyorsítótárazás, valamint a backend API-k válaszidejének csökkentése is fontos lehet. Emellett a reszponzív design továbbfejlesztése és a mobilbarát felület kialakítása még szélesebb felhasználói bázist vonzhatna.

Összességében a projekt egy erős alapokon nyugvó, jól működő rendszer, amely már most is számos hasznos funkciót kínál a felhasználók számára. A jövőbeli fejlesztések és finomhangolások révén pedig még inkább kiemelkedővé válhat, és hosszú távon is fenntartható, modern megoldást nyújthat.

Irodalomjegyzék

Űrlap-elemek megvalósítása HTML-ben
http://infojegyzet.hu/webszerkesztes/html/urlapok/

PHP alapok - szerveroldali programozás
http://infojegyzet.hu/webszerkesztes/php/alapok/

Záródolgozat témák, szempontok, ötletek
http://infojegyzet.hu/webszerkesztes/zarodolgozat/

W3School
https://www.w3schools.com/

Bootstrap 5
https://getbootstrap.com/

Nagy Gusztáv: Webprogramozás alapismeretek (2011)

Tutorials Point -SimplyEasyLearning : Learn AngularJS , web application framework (2014)