**SZABADKAI MŰSZAKI SZAKFŐISKOLA**

**PROJEKTUM**

**Haladó web programozás és Integrált web rendszerek tantárgyakból**

**HALLGATÓ: MENTOR**

**Molnár Alesz dr Zlatko Čović**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SZABADKA, 2023.**

## TARTALOM

Catalog

[TARTALOM 2](#_Toc24654)

[A FELADATKÖR MEGHATÁROZÁSA 3](#_Toc18172)

[PIACKUTATÁS 3](#_Toc10767)

[KIDOLGOZÁS 5](#_Toc1776)

[AZ ADATBÁZIS STRUKTURÁJA 14](#_Toc15275)

[A PROJEKT MŰKÖDÉSÉNEK LEÍRÁSA 15](#_Toc25172)

[FELHASZNÁLT IRODALOM 21](#_Toc27388)

## A FELADATKÖR MEGHATÁROZÁSA

A projekt első fázisa egy weboldal létrehozása volt, mely lehetővé teszi számunkra, hogy bármely város aktuális és hét napos időjárás-előrejelzését megtekintsük. Emellett a weboldalon kaphatunk öltözködési tanácsokat az adott napi hőmérsékletre. Regisztrálhatunk és bejelentkezhetünk, lehetőséget kapunk profilunk és jelszavunk módosítására. Kedvenc városainkat összeállíthatjuk és törölhetjük, valamint kérhetünk csapadékértesítéseket választott településeinkhez. Profilunkban visszanézhetjük korábbi látogatott városainkat. Emellett a weboldalon elérhető egy hírek rovat is, ahol az időjárással kapcsolatos friss híreket olvashatunk. Ezen kívül világórával is rendelkezünk, amely segítségével bármelyik városban megtekinthetjük az aktuális időt.

Az adminisztrátori felület lehetővé teszi a felhasználók kezelését és a legtöbbet illetve legutóbbi látogatott városokról statisztikákat tekinthetünk meg.

A projekt második részében egy mobilalkalmazás készült, amely magában foglalja az összes fent említett funkcionalitást, ám más designnal és felülettel rendelkezik.

A webes alkalmazás React, JavaScript, PHP, CSS és SQL segítségével készült, míg a mobilalkalmazás Expo Go App platformon fut.

## PIACKUTATÁS

A piackutatást folytatván több időjárás előrejelzés oldalt megtekintettem és átnéztem. Mindegyiken megtalálható az alapvető funkciók mint az aznapi előrejelzés és a több napos előrejelzés. Számomra a legszebb és legszimpatikusabb weboldal a Magyarországi koponyeg.hu, azaz köpönyeg weboldal. Gyönyörű design és rengeteg kiegészítő opció. Könnyen áttekinthető. Rengeteg kiegészítéssel rendelkezik mint például több radar térkép, hírek, észlelések, tudásfelhő, orvosmeteorológia.

Innen jött az ötlet hogy egy hírportállal egészítsem ki a projektem.

Ez mellett bővítettem egy világórával. Ahol rákereshetünk városokra és megtekinthetjük hány óra van ott ebben a pillanatban.   
A harmadik funkció pedig egy ruházat leírás az adott napra. Ha rákeresünk egy városra a meteorológia alapján kapunk egy leírást az ajánlott öltözködésről.

## KIDOLGOZÁS

### Profile.js:

import React, { useEffect, useState } from "react";

import "./profile.css";

const Profile = () => {

  const [userData, setUserData] = useState(null);

  const [loading, setLoading] = useState(true);

  const [detailsSwitch, setDetailsSwitch] = useState(null);

  const [lastVisited, setLastVisited] = useState([]);

  const [notifications, setNotifications] = useState([]);

  useEffect(() => {

    const fetchUserProfile = async () => {

      try {

        const token = localStorage.getItem("token");

        if (!token) {

          window.location.href = "http://localhost:3000/login";

          return;

        }

        const response = await fetch(

          "https://localhost/mdb5react/profile.php",

          {

            method: "GET",

            headers: {

              Authorization: token,

            },

          }

        );

        const result = await response.json();

        if (result.success) {

          setUserData(result.user\_data);

          const lastVisitedResponse = await fetch(

            "https://localhost/mdb5react/profile.php",

            {

              method: "GET",

              headers: {

                Authorization: token,

              },

            }

          );

          const lastVisitedResult = await lastVisitedResponse.json();

          if (lastVisitedResult.success) {

            setLastVisited(lastVisitedResult.last\_visited);

          } else {

            console.error(lastVisitedResult.message);

          }

          const optionsResponse = await fetch(

            "https://localhost/mdb5react/getOptions.php",

            {

              method: "GET",

              headers: {

                Authorization: token,

              },

            }

          );

          const optionsResult = await optionsResponse.json();

          if (optionsResult.success) {

            setDetailsSwitch(optionsResult.detailsSwitch);

          } else {

            console.error(optionsResult.message);

          }

          const notificationsResponse = await fetch(

            "https://localhost/mdb5react/shownotifications.php",

            {

              method: "GET",

              headers: {

                Authorization: token,

              },

            }

          );

          const notificationsResult = await notificationsResponse.json();

          if (notificationsResult.success) {

            setNotifications(notificationsResult.notifications);

          } else {

            console.error(notificationsResult.message);

          }

        } else {

          console.error(result.message);

          if (result.message === "Invalid or expired token.") {

            localStorage.removeItem("token");

            window.location.href = "http://localhost:3000/login";

          }

        }

      } catch (error) {

        console.error("An unexpected error occurred:", error);

      } finally {

        setLoading(false);

      }

    };

    fetchUserProfile();

  }, []);

  const handleToggleNotifications = async (city) => {

    try {

      const token = localStorage.getItem("token");

      if (!token) {

        window.location.href = "http://localhost:3000/login";

        return;

      }

      const response = await fetch(

        "https://localhost/mdb5react/turnoffnotifications.php",

        {

          method: "POST",

          headers: {

            "Content-Type": "application/json",

            Authorization: token,

          },

          body: JSON.stringify({ city: city }),

        }

      );

      const result = await response.json();

      if (result.success) {

        const notificationsResponse = await fetch(

          "https://localhost/mdb5react/shownotifications.php",

          {

            method: "GET",

            headers: {

              Authorization: token,

            },

          }

        );

        const notificationsResult = await notificationsResponse.json();

        if (notificationsResult.success) {

          setNotifications(notificationsResult.notifications);

        } else {

          console.error(notificationsResult.message);

        }

      } else {

        console.error(result.message);

      }

    } catch (error) {

      console.error("An unexpected error occurred:", error);

    }

  };

  const handleRemoveFavorite = async (city) => {

    try {

      const token = localStorage.getItem("token");

      if (!token) {

        window.location.href = "http://localhost:3000/login";

        return;

      }

      const response = await fetch(

        `https://localhost/mdb5react/removeFavorite.php?city=${city}`,

        {

          method: "DELETE",

          headers: {

            Authorization: token,

          },

        }

      );

      const result = await response.json();

      if (result.success) {

        setUserData((prevUserData) => ({

          ...prevUserData,

          favorites: prevUserData.favorites.filter((fav) => fav !== city),

        }));

      } else {

        console.error(result.message);

      }

    } catch (error) {

      console.error("An unexpected error occurred:", error);

    }

  };

  const handleSwitchToggle = async () => {

    try {

      const token = localStorage.getItem("token");

      if (!token) {

        window.location.href = "http://localhost:3000/login";

        return;

      }

      const newSwitchValue = detailsSwitch ? 0 : 1;

      const switchResponse = await fetch(

        "https://localhost/mdb5react/updateSwitch.php",

        {

          method: "POST",

          headers: {

            "Content-Type": "application/json",

            Authorization: token,

          },

          body: JSON.stringify({ switchValue: newSwitchValue }),

        }

      );

      const switchResult = await switchResponse.json();

      if (switchResult.success) {

        setDetailsSwitch(newSwitchValue);

      } else {

        console.error(switchResult.message);

      }

    } catch (error) {

      console.error("An unexpected error occurred:", error);

    }

  };

  const handleSnowNotifications = async (city) => {

console.log("City:", city);

    try {

      const response = await fetch(

        "https://localhost/mdb5react/snownotifications.php",

        {

          method: "POST",

          headers: {

            "Content-Type": "application/json",

            Authorization: localStorage.getItem("token"),

          },

          body: JSON.stringify({ city: city }),

        }

      );

      const result = await response.json();

      if (result.success) {

        console.log("Snow notification added successfully!");

             } else {

        console.error(result.message);

        if (result.message === "Invalid or expired token.") {

          localStorage.removeItem("token");

          window.location.href = "http://localhost:3000/login";

        } else if (result.message === "Authorization header missing.") {

                  } else {

        }

      }

    } catch (error) {

      console.error("An unexpected error occurred:", error);

          }

  };

  return (

    <div className="profile-container">

      {loading ? (

        <p>Loading...</p>

      ) : userData ? (

        <div className="profile-info">

          <h2>Welcome, {userData.user\_name}!</h2>

          <p>Email: {userData.user\_email}</p>

          <h3>Favorites:</h3>

          {userData.favorites && userData.favorites.length > 0 ? (

            <ul className="favorites-list">

              {userData.favorites.map((favorite, index) => (

                <li key={index}>

                  {favorite}{" "}

                  <button onClick={() => handleRemoveFavorite(favorite)}>

                    Remove

                  </button>

                  <button onClick={() => handleSnowNotifications(favorite)}>

                    Turn on notifications

                  </button>

                </li>

              ))}

            </ul>

          ) : (

            <p>No favorites yet.</p>

          )}

          {}

          {detailsSwitch !== null && (

            <div className="details-switch">

              <p>Details are turned {detailsSwitch ? "off" : "on"}.</p>

              <button onClick={handleSwitchToggle}>Toggle Switch</button>

            </div>

          )}

          <h3>Last Visited:</h3>

          {lastVisited && lastVisited.length > 0 ? (

            <ul className="last-visited-list">

              {lastVisited.map((visit, index) => (

                <li key={index}>

                  {visit.city} - {visit.date\_time}

                </li>

              ))}

            </ul>

          ) : (

            <p>No last visited sites yet.</p>

          )}

          <h3>Notifications:</h3>

          {notifications && notifications.length > 0 ? (

            <>

              <table className="notifications-table">

                <thead>

                  <tr>

                    <th>City</th>

                    <th>Expires</th>

                    <th>Status</th>

                    <th>Action</th>

                  </tr>

                </thead>

                <tbody>

                  {notifications.map((notification, index) => (

                    <tr key={index}>

                      <td>{notification.city}</td>

                      <td>{notification.expires}</td>

                      <td>{notification.active ? "Active" : "Inactive"}</td>

                      <td>

                        <button

                          onClick={() =>

                            handleToggleNotifications(notification.city)

                          }

                        >

                          {notification.active ? "Turn Off" : "Turn On"}

                        </button>

                      </td>

                    </tr>

                  ))}

                </tbody>

              </table>

            </>

          ) : (

            <p>No notifications yet.</p>

          )}

          <button

            className="update-profile-btn"

            onClick={() =>

              (window.location.href = "http://localhost:3000/updateProfile")

            }

          >

            Update your profile

          </button>

        </div>

      ) : (

        <p>Error loading profile.</p>

      )}

    </div>

  );

};

export default Profile;

### Profile.php:

<?php

header('Content-Type: application/json');

header("Access-Control-Allow-Origin: http://localhost:3000");

header("Access-Control-Allow-Methods: GET");

header("Access-Control-Allow-Headers: Content-Type, Authorization");

ini\_set('display\_errors', 'On');

error\_reporting(E\_ALL);

$response = array();

if ($\_SERVER['REQUEST\_METHOD'] === 'GET') {

       $headers = apache\_request\_headers();

    if (isset($headers['Authorization'])) {

        $token = $headers['Authorization'];

        include("db\_config1.php");

        $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username, $password);

        $pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

        $stmt = $pdo->prepare("SELECT \* FROM tokens WHERE token = :token AND expiry\_time > NOW()");

        $stmt->bindParam(':token', $token);

        $stmt->execute();

        $row = $stmt->fetch(PDO::FETCH\_ASSOC);

        if ($row) {

            $user\_id = $row['user\_id'];

            $stmt = $pdo->prepare("SELECT \* FROM user WHERE user\_id = :user\_id");

            $stmt->bindParam(':user\_id', $user\_id);

            $stmt->execute();

            $user\_data = $stmt->fetch(PDO::FETCH\_ASSOC);

            if ($user\_data) {

                $response['success'] = true;

                $response['user\_data'] = $user\_data;

                $stmtFavorites = $pdo->prepare("SELECT city FROM favorites WHERE user\_id = :user\_id");

                $stmtFavorites->bindParam(':user\_id', $user\_id);

                $stmtFavorites->execute();

                $favorites = $stmtFavorites->fetchAll(PDO::FETCH\_COLUMN);

                $response['user\_data']['favorites'] = $favorites

                $stmtLastVisited = $pdo->prepare("

                    SELECT city, date\_time

                    FROM views

                    WHERE user\_id = :user\_id

                    ORDER BY date\_time DESC

                    LIMIT 5

                ");

                $stmtLastVisited->bindParam(':user\_id', $user\_id);

                $stmtLastVisited->execute();

                $lastVisited = $stmtLastVisited->fetchAll(PDO::FETCH\_ASSOC);

                $response['last\_visited'] = $lastVisited;

            } else {

                $response['success'] = false;

                $response['message'] = "User not found.";

            }

        } else {

            $response['success'] = false;

            $response['message'] = "Invalid or expired token.";

        }

    } else {

        $response['success'] = false;

        $response['message'] = "Authorization header missing.";

    }

} else {

    $response['success'] = false;

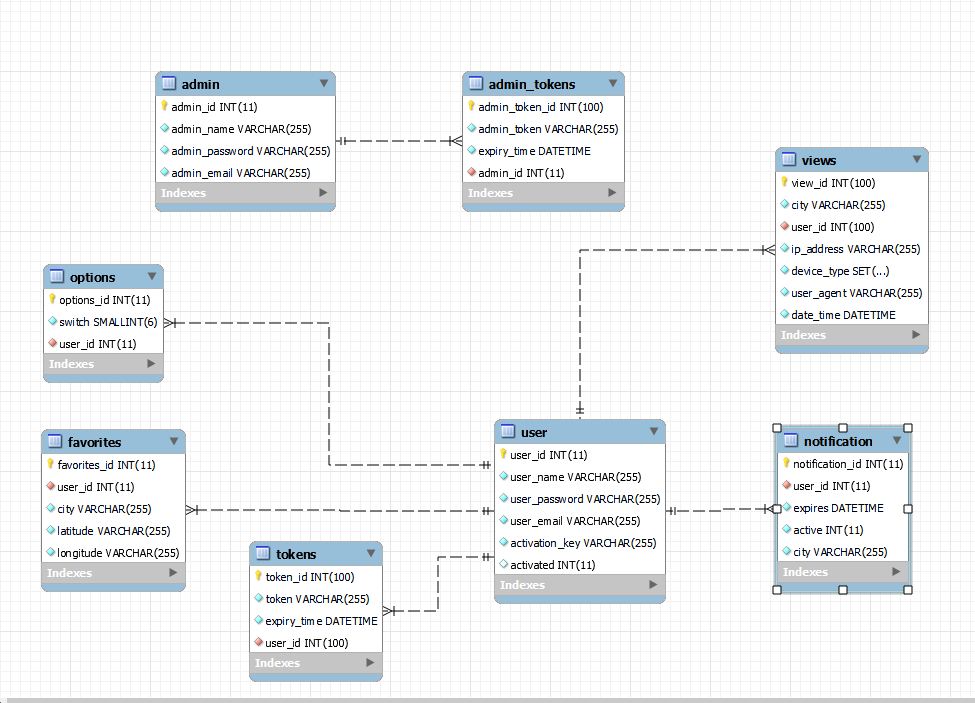
    $response['message'] = "Invalid request method.";

}

echo json\_encode($response);

## AZ ADATBÁZIS STRUKTURÁJA

Az időjárás adatbázis struktúrája a MySQL workbench használatával:



## A PROJEKT MŰKÖDÉSÉNEK LEÍRÁSA

A weboldal front-end-je React-ben készült, a back-end-je pedig PHP.

Mindegyik React oldalon JavasSripttel fetch-eljük a PHP kódot így összekapcsoljuk a két kódot és adatokat küldünk oda-vissza. Az összes ilyen fetch-elésnél több catch-et használtam, hogy a hibákat elkapjam és console.log-oljam ha valami sikerült vagy sem.

A weather oldal 3 komponensből áll: Current-weather, Forecast és Search.

A Search-el keresünk rá az adott helyre.

A Current-weather a mai időt mutatja.

A Forecast pedig a hét napos előrejelzés.

A Time oldalon a világórát tudjuk megtekinteni, a Föld minden pontján.

A News oldalon hírlapokat tudunk olvasni az időjárással kapcsolatban.

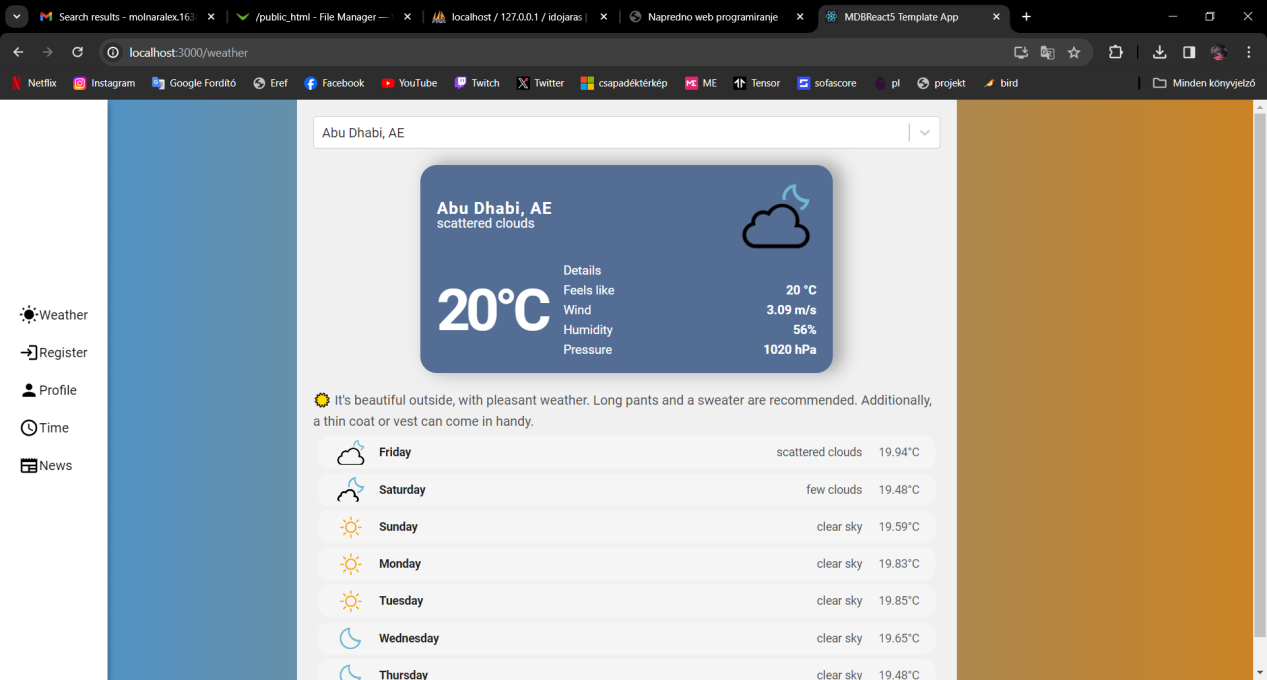
A Navbar változik, ha nem vagyunk bejelentkezve akkor látjuk a Register gombot, ha bejelentkeztünk a Register helyett logout gomb van.  
  
Register oldalon tudunk regisztrálni ami a sikeres regisztráció után emailt küld a profilunk aktiválásához.

Login oldalon tudunk bejelentkezni, ha sikeresen bejelentkeztünk átvisz bennünket a Profile oldalra. Itt tudunk átugrani a jelszó csere vagy az admin login oldalra.

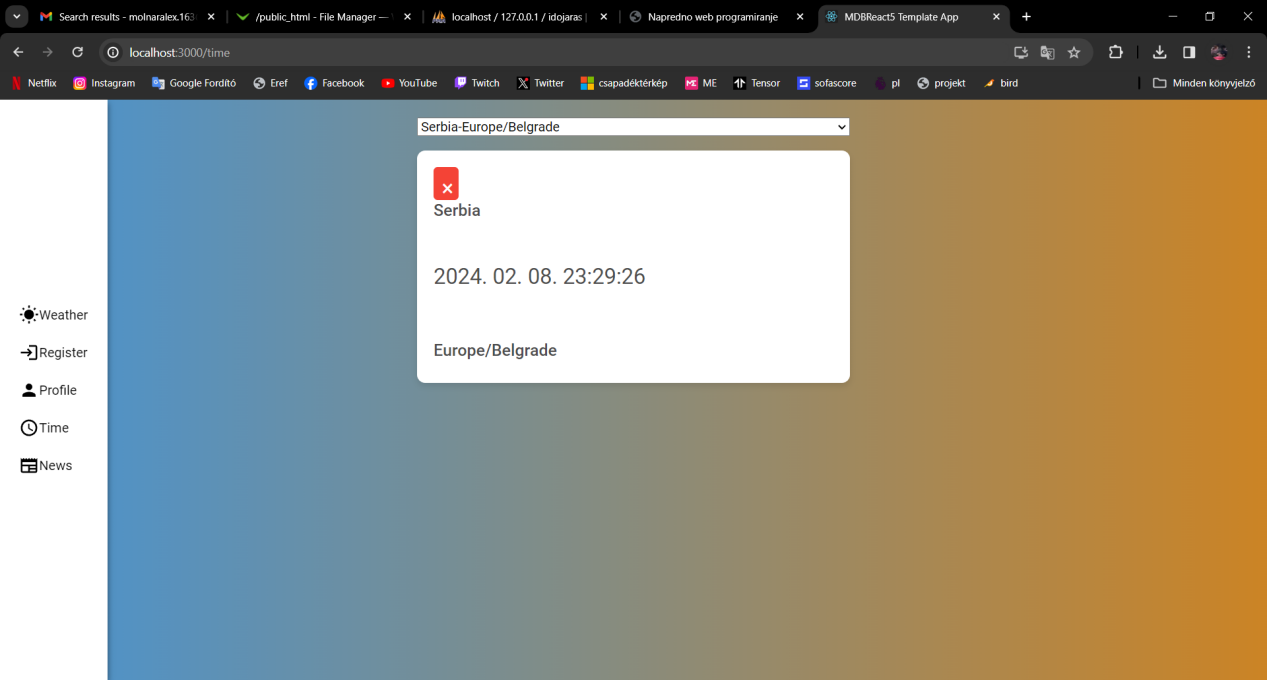
A Profile oldalon láthatjuk a saját adatainkat. A kedvenc városainkat ahol törölhetjük őket és értesítést tudunk kérni. A részleteket tudjuk ki/be kapcsolni. Láthatjuk az utoljára megtekintett városokat. Láthatjuk az eddigi értesítéseinket.

Az updateProfile oldalon módosíthatjuk adatainkat.

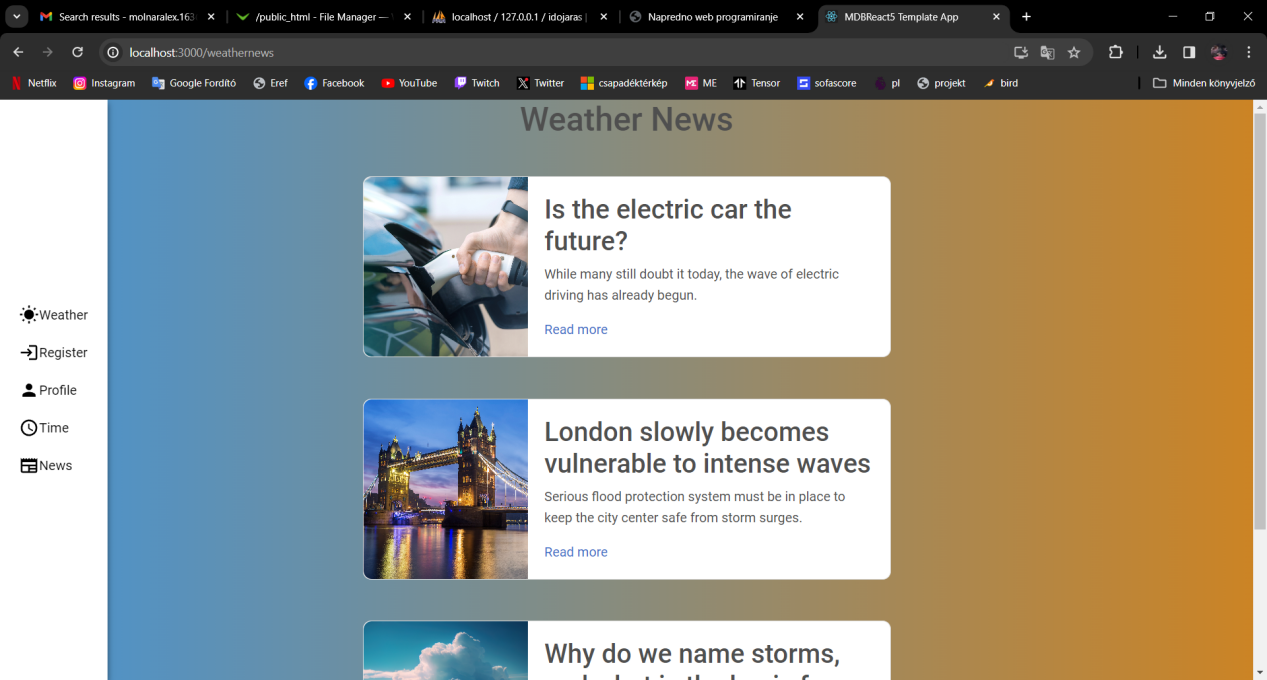
Az admin oldalon kezelhetjük a felhasználókat, törölhetjük és deaktiválhatjuk őket.  
A statisztikákat is csak az adminisztrátor érheti el, itt megtekinthetjük a látogatási statisztikákat.



Weather oldal kijelentkezve



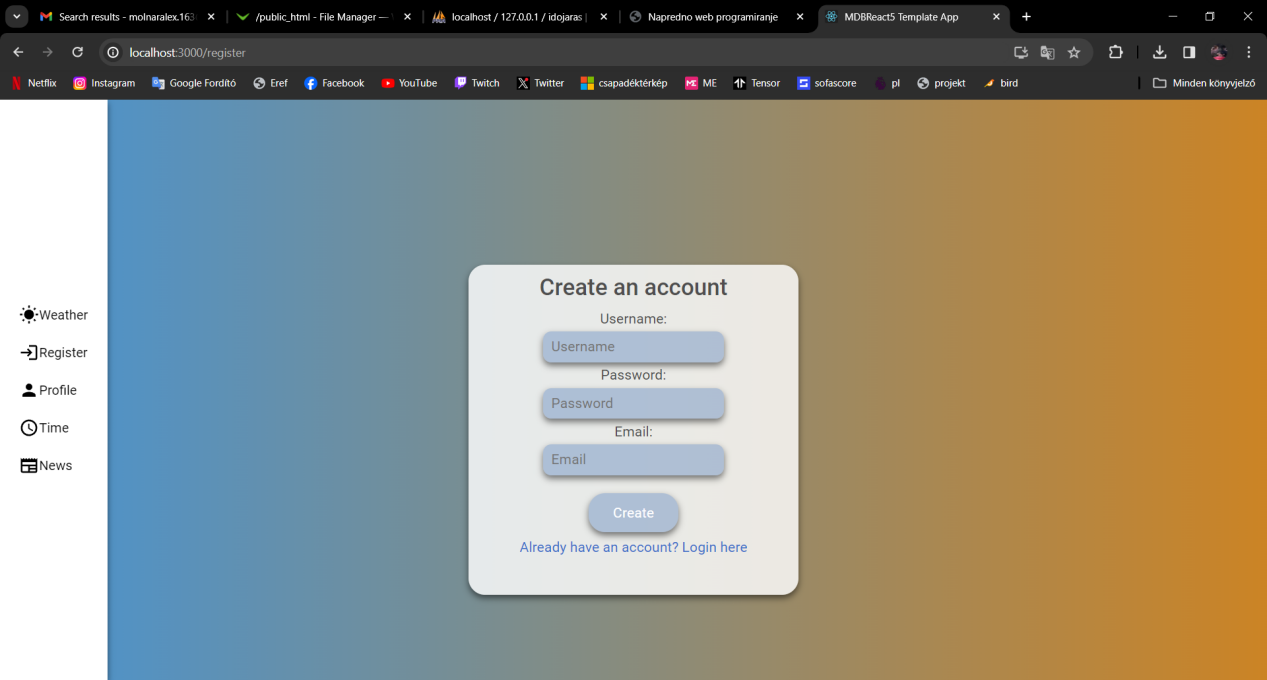
Time oldal



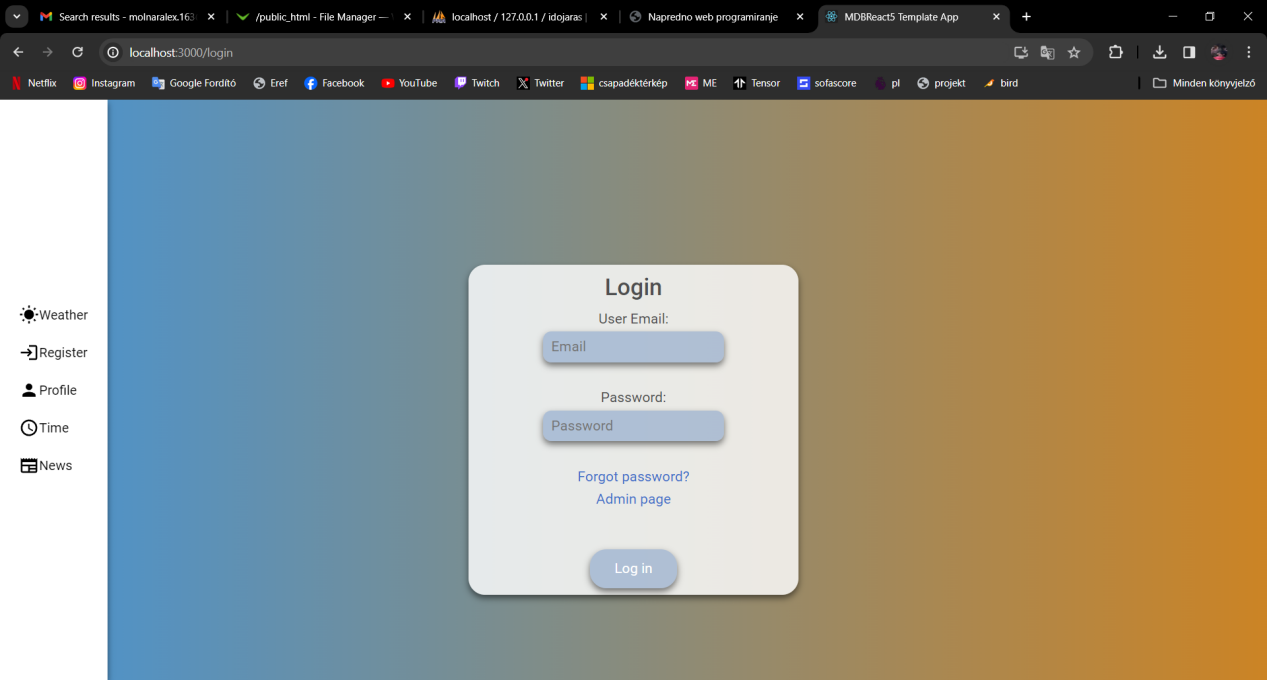
WeatherNews oldal



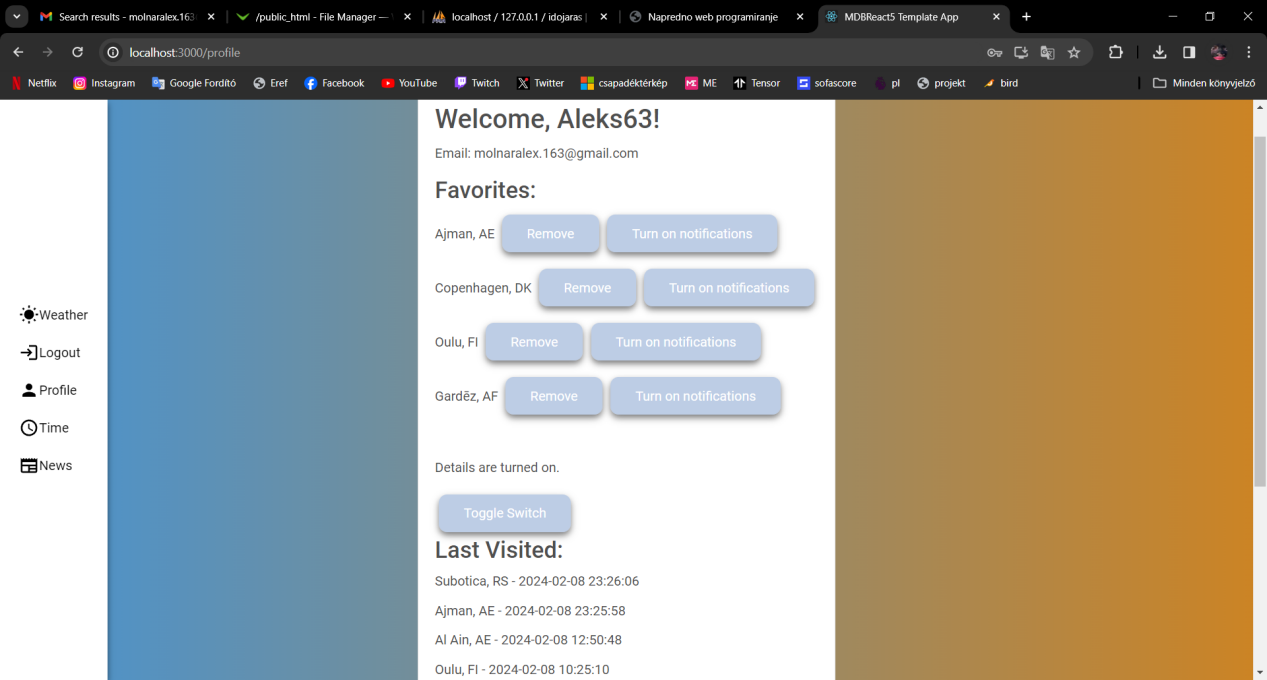
Első hírlap



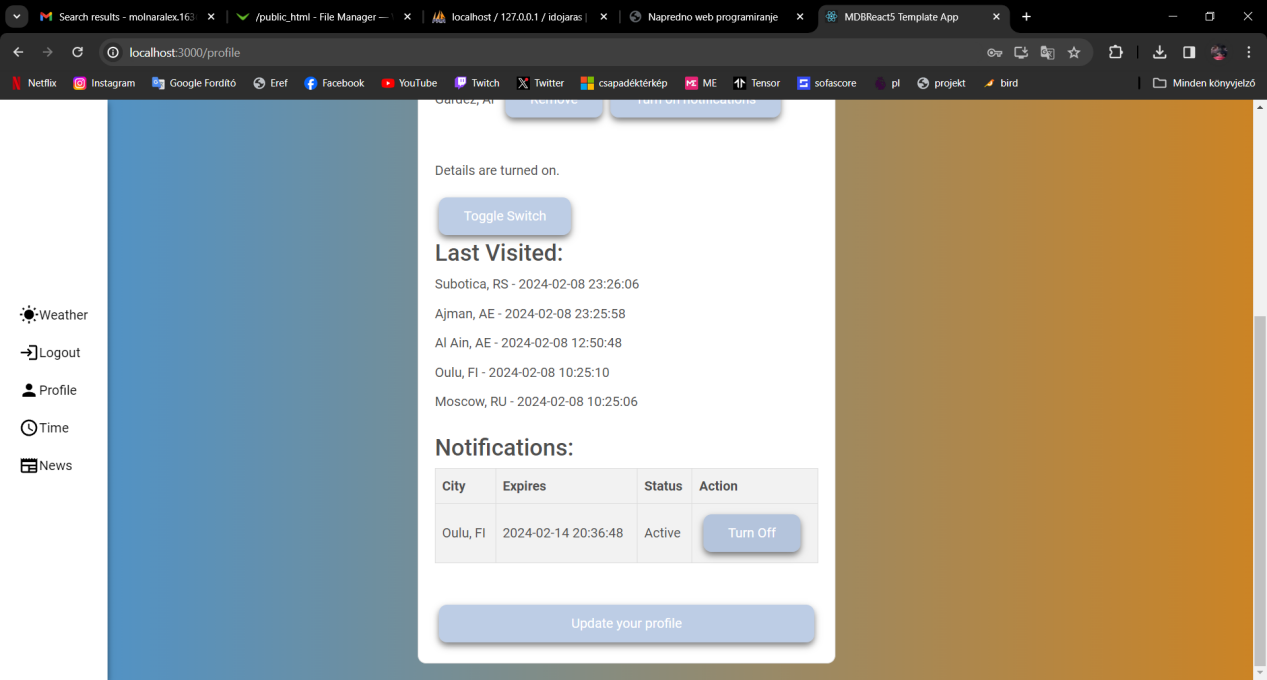
Regisztráció



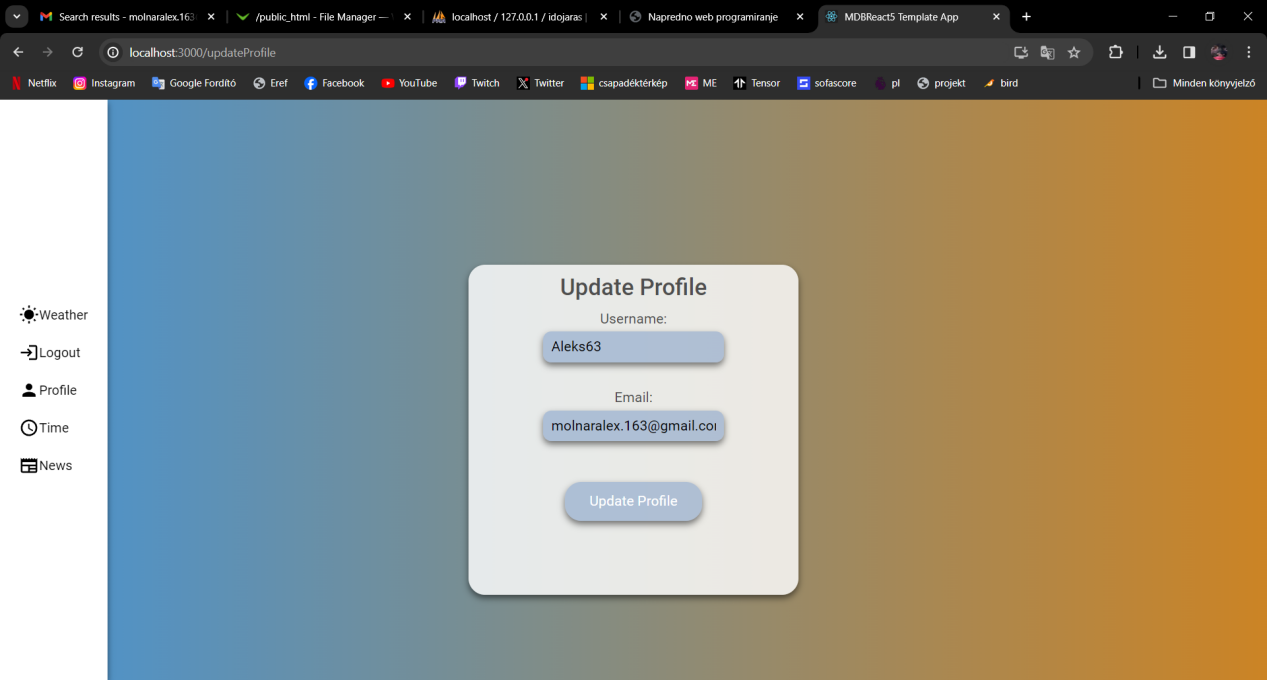
Bejelentkezés



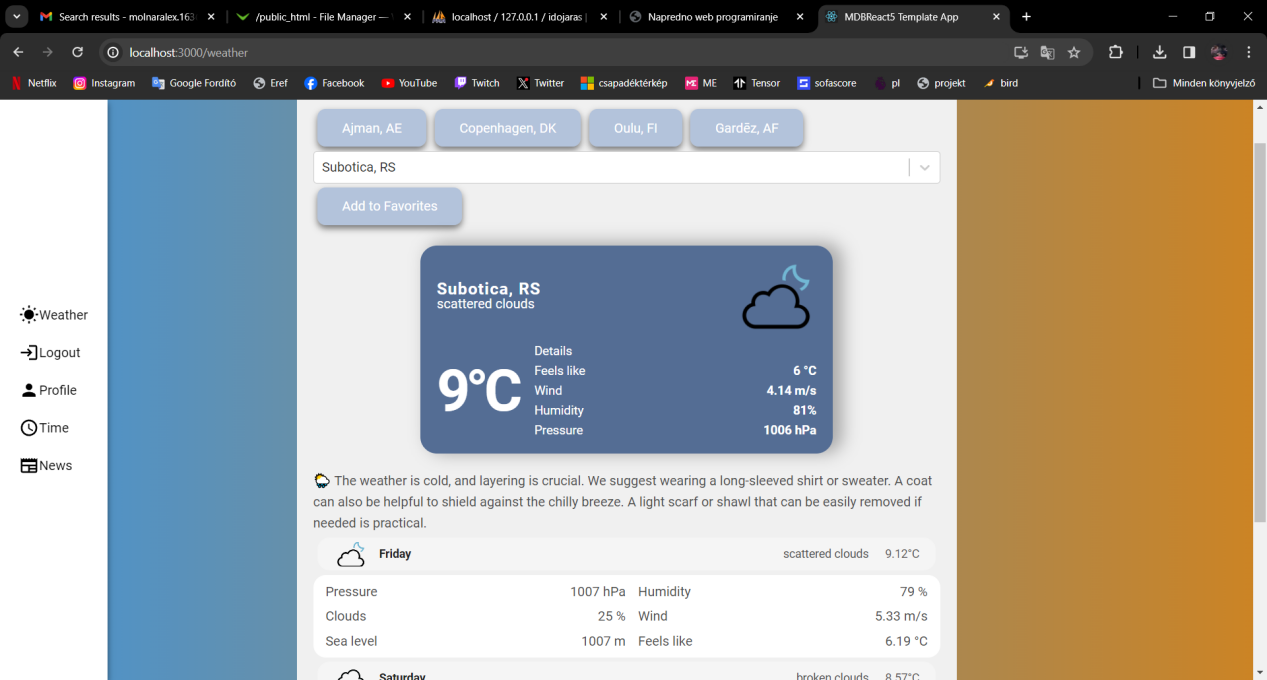
Felhasználó profil 1.



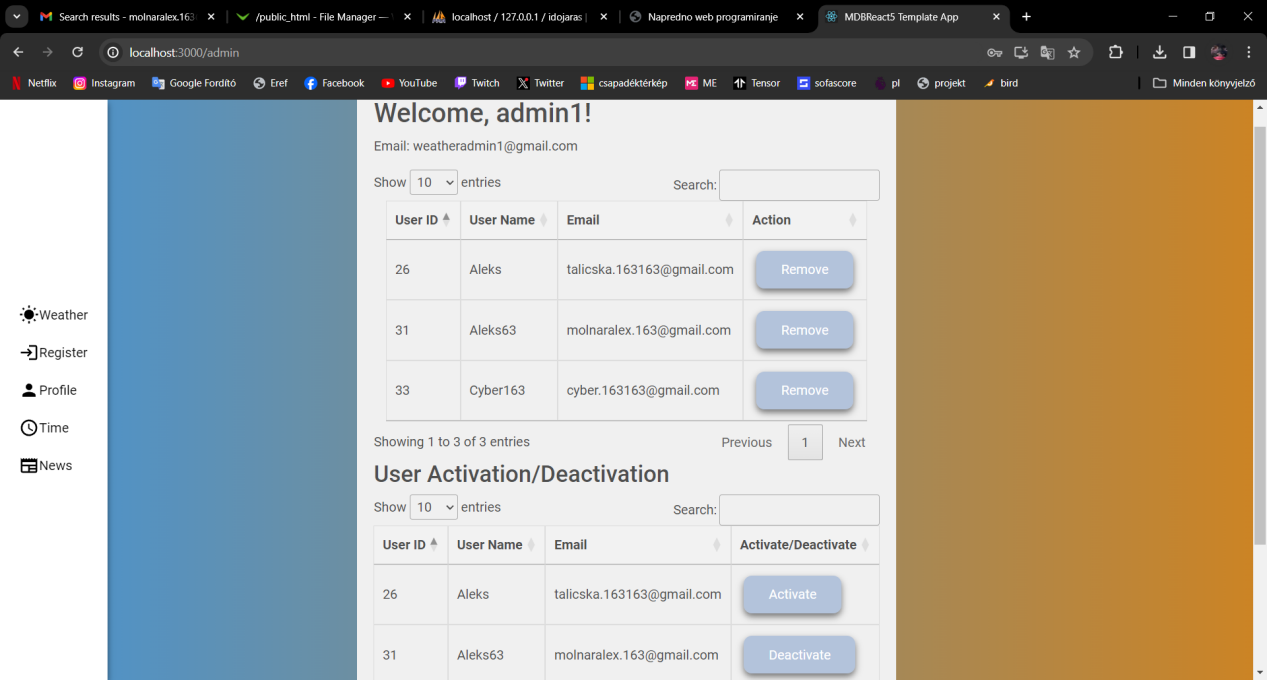
Felhasználó profil 2.



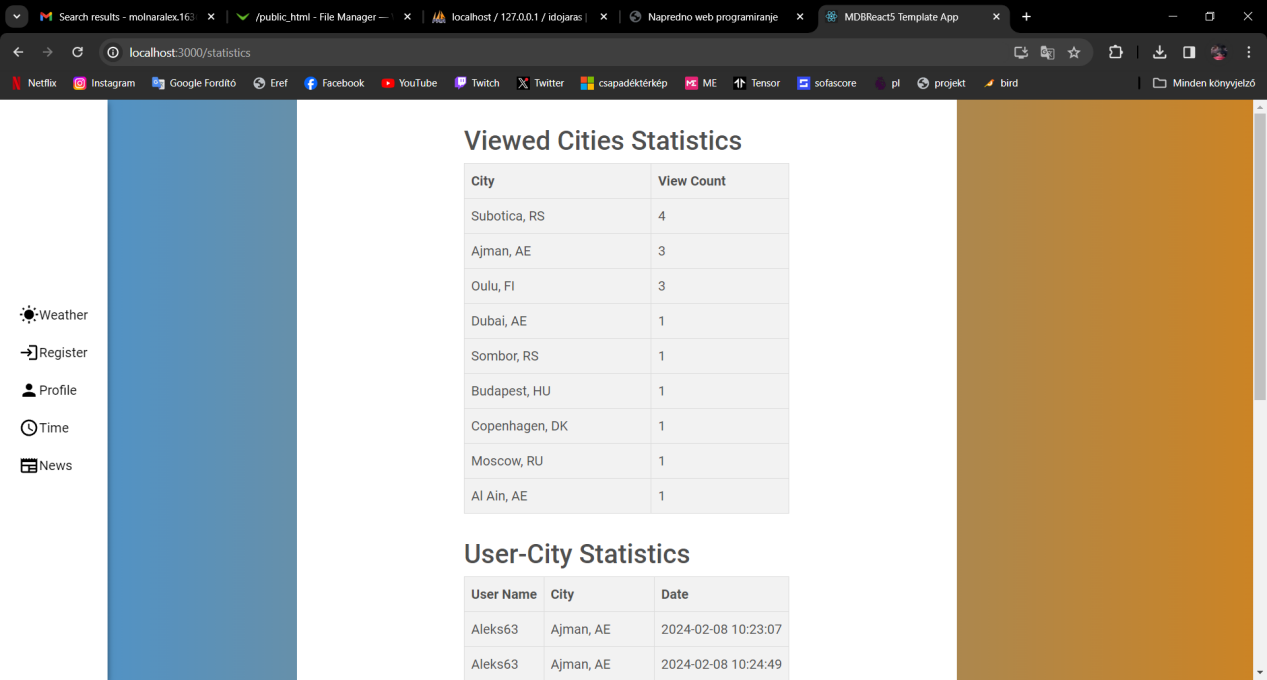
Felhasználó profil szerkesztése



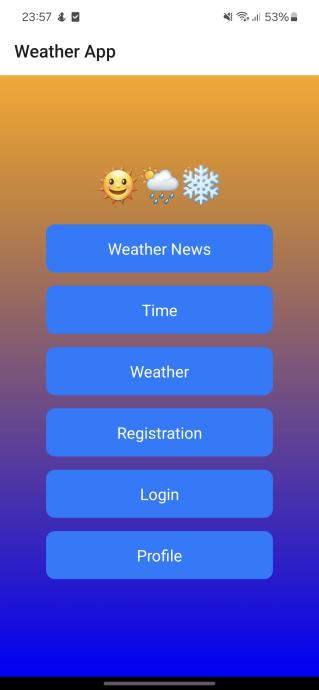
Weather oldal bejelentkezve



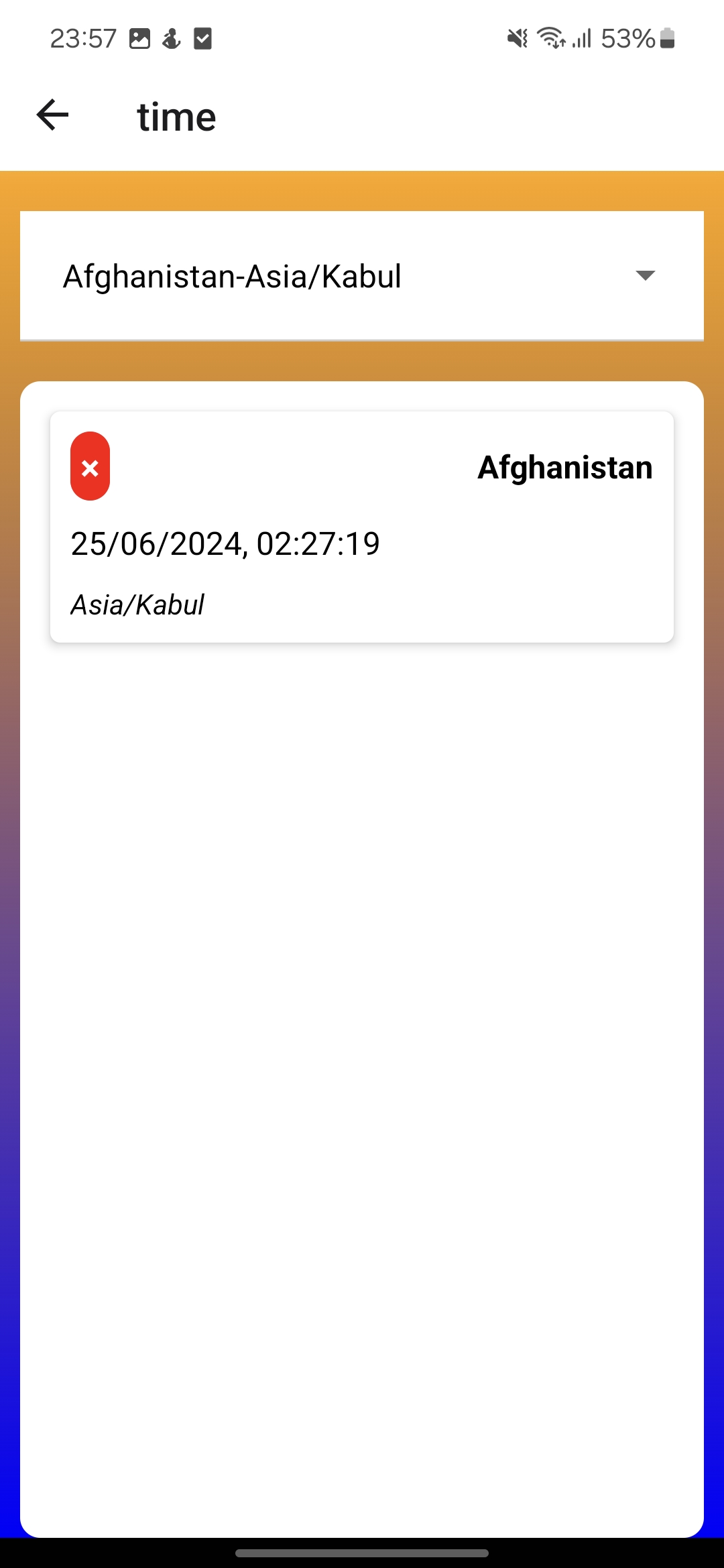
Admin oldal



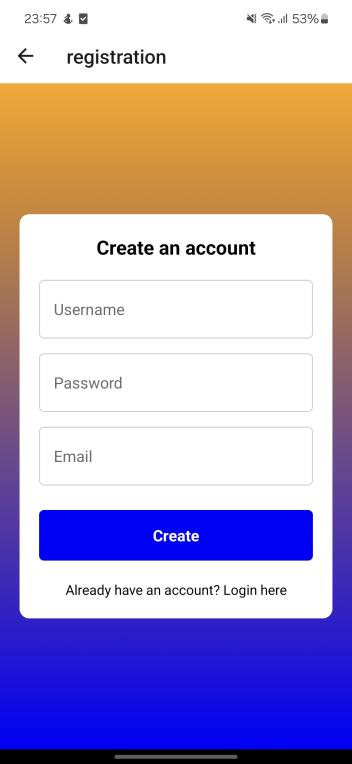
Statistics oldal



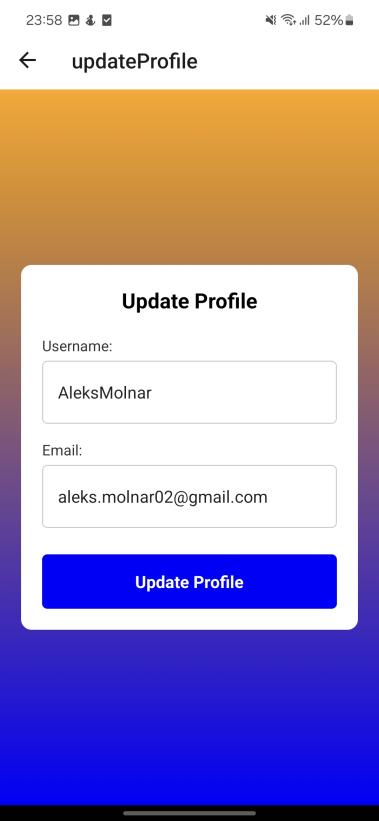
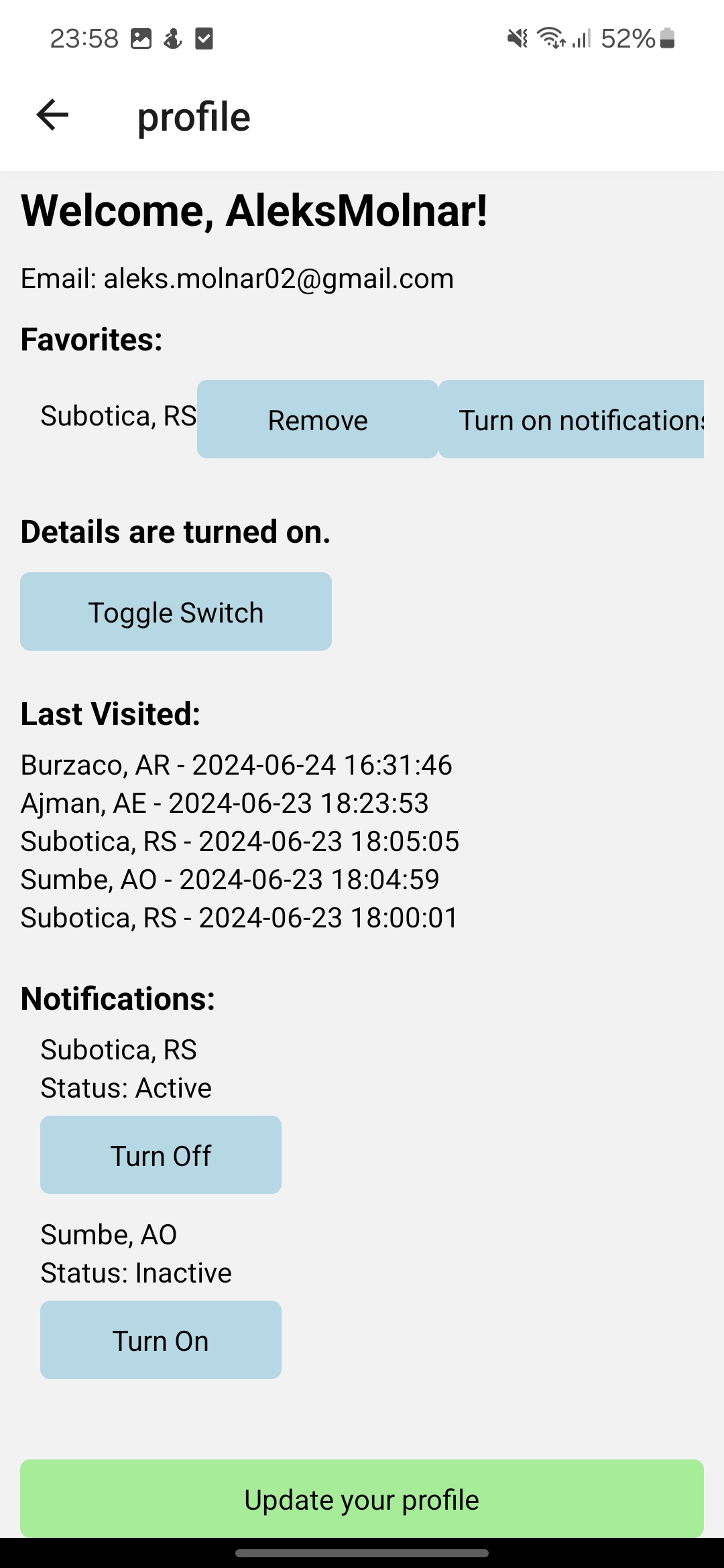
Kezdőlap és hírek



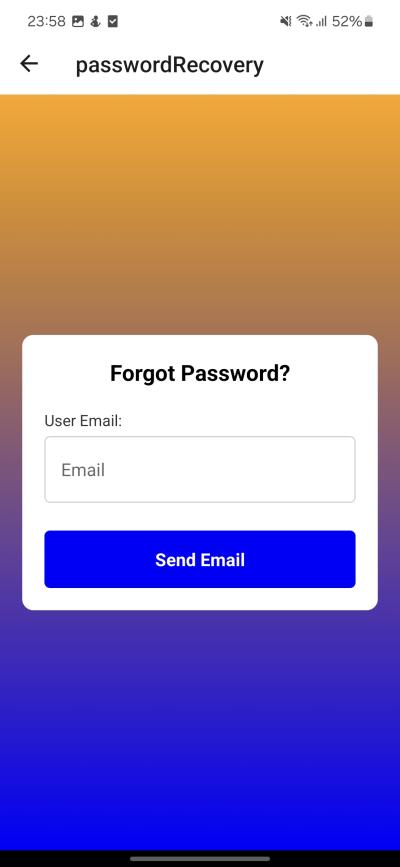
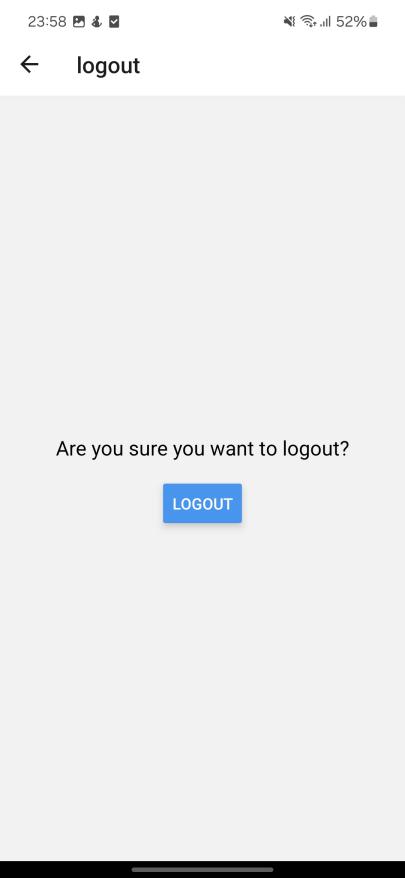
Hírlap és idő



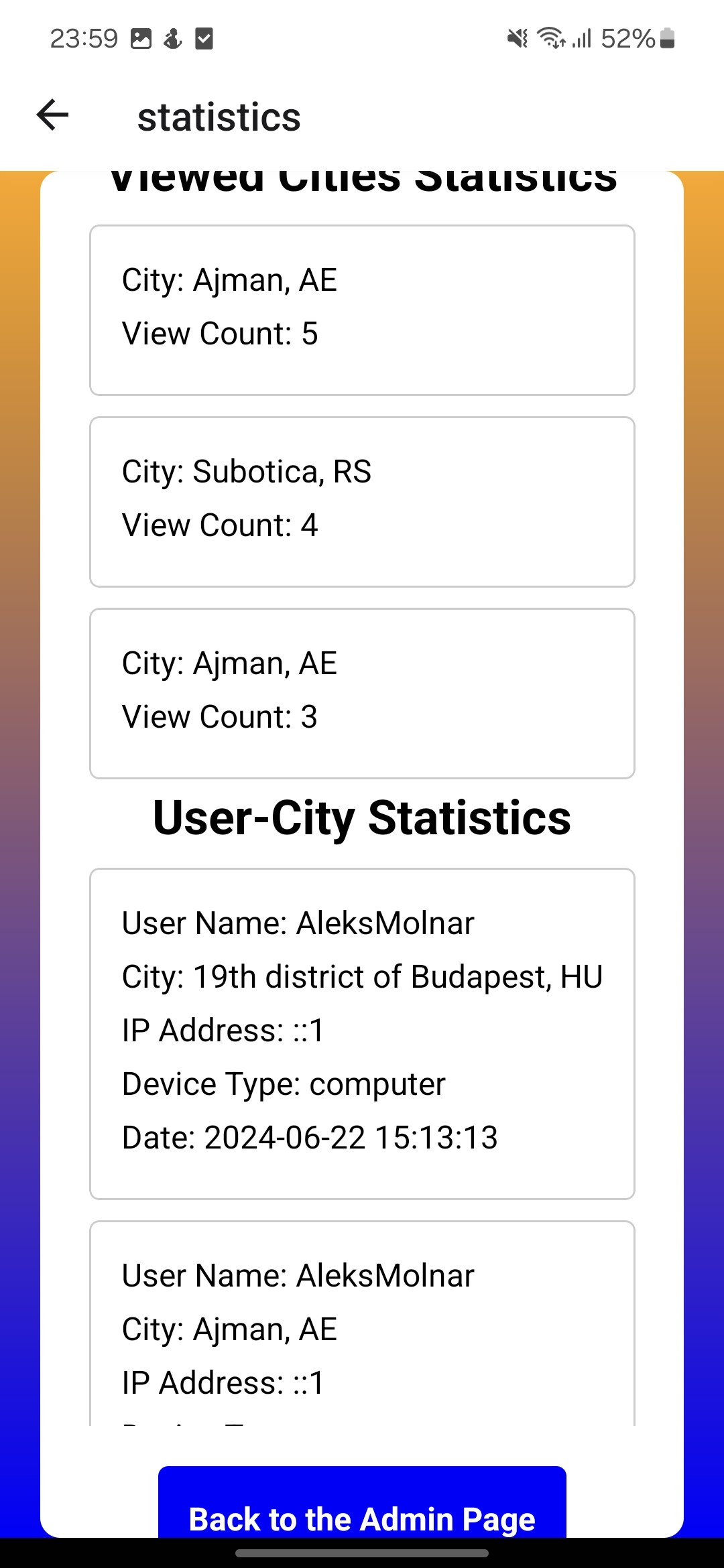
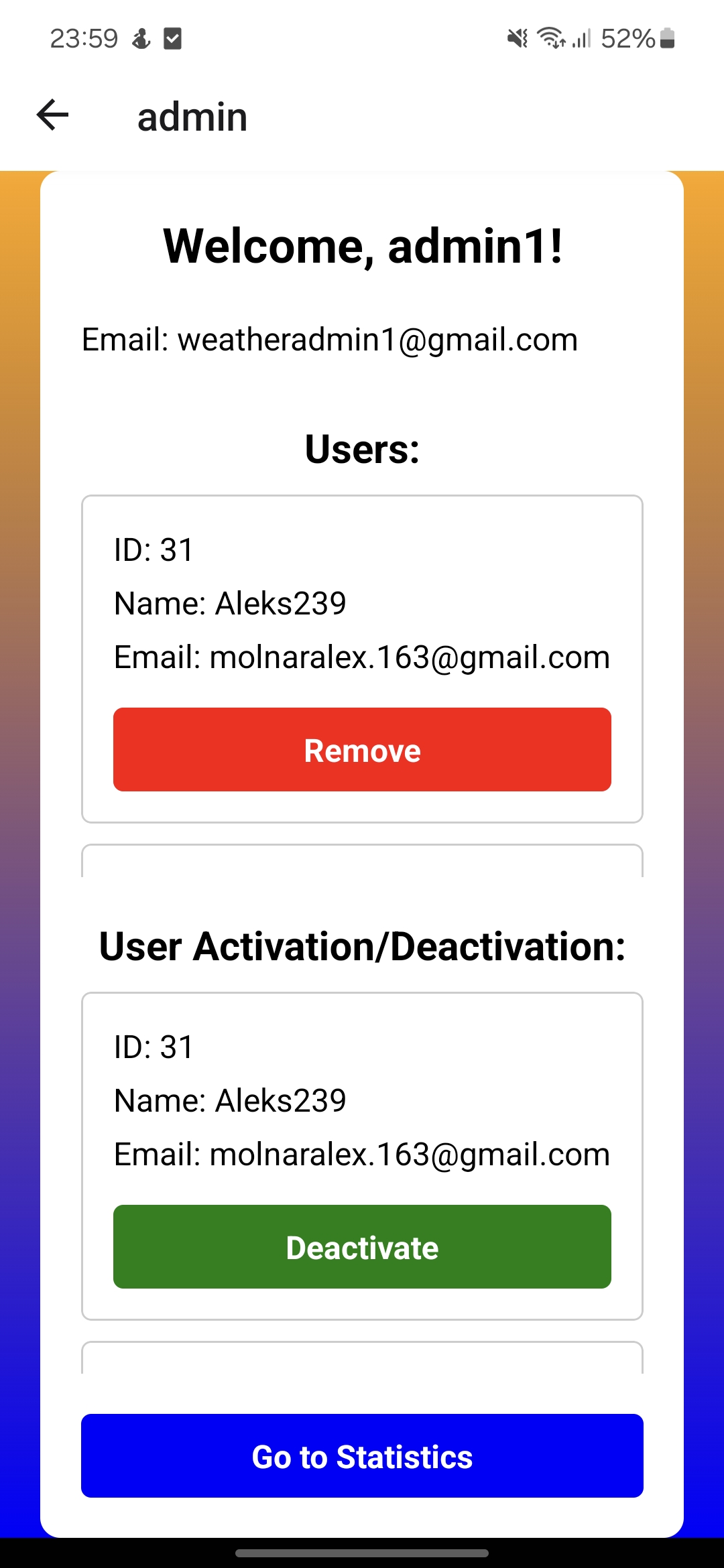
Időjárás és regisztráció



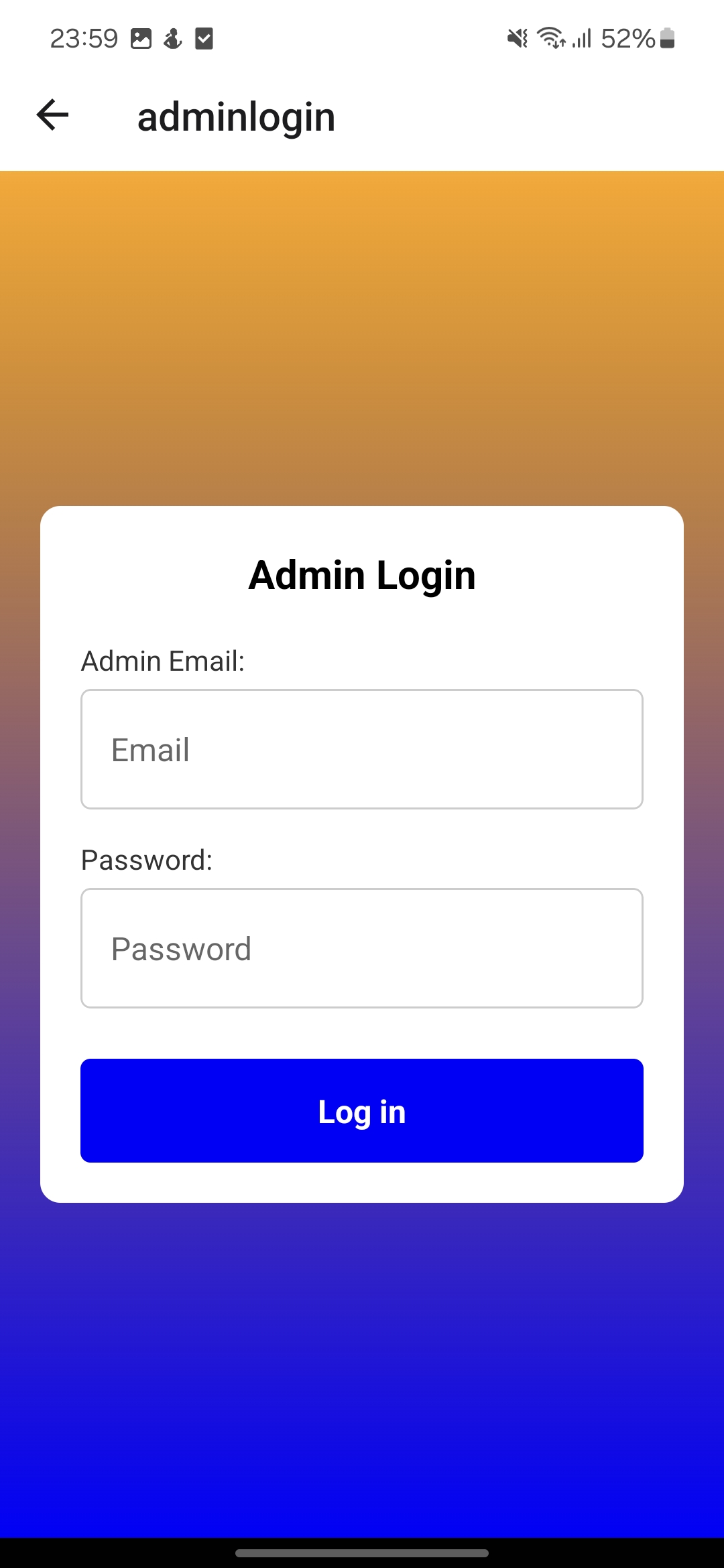
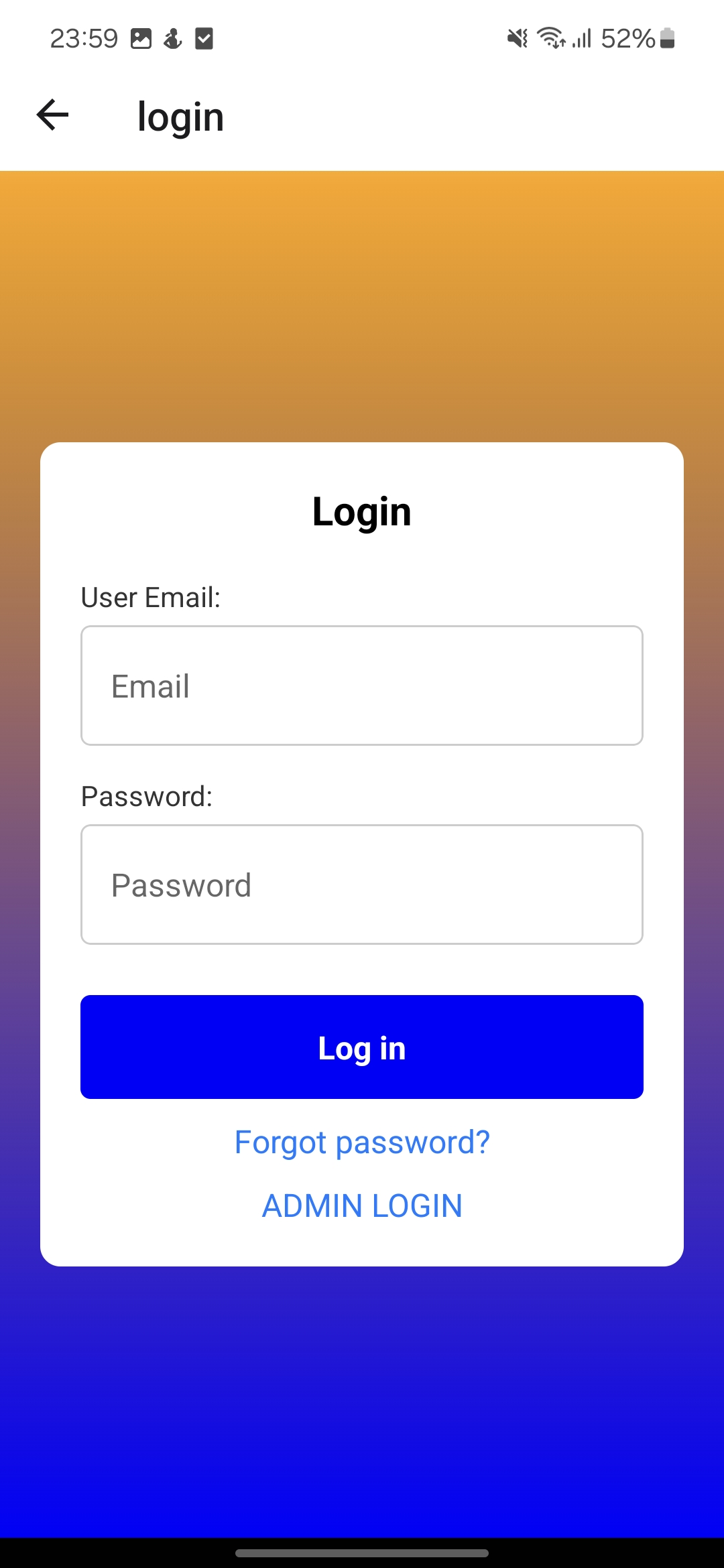
Profil és profil szerkesztése



Kijelentkezés és jelszó csere



Admin és statisztika



bejelentkezés

## FELHASZNÁLT IRODALOM

W1. [https://rapidapi.com/wirefreethought/api/geodb-cities/l](https://reactnative.dev/docs/tutorial)

W2. <https://openweathermap.org/>

W3. <https://www.mysql.com/products/workbench/>

W4. <https://mui.com/material-ui/icons/>

W5. <https://reactnative.dev/>

W6. <https://www.php.net/>

W7. <https://datatables.net/>