Kauno technologijos universitetas

Informatikos fakultetas

T120B165 Saityno taikomųjų programų projektavimas
Projekto „TravelRec“ ataskaita

Aurimas Žadajevas
Studentas

Prof. Blažauskas Tomas
Doc. prakt. Mažutienė Rasa
Dėstytojas

Kaunas, 2025

Turinys

1. Sprendžiamo uždavinio aprašymas

1.1. Sistemos paskirtis

Projekto tikslas – sukurti internetinę platformą, kuri palengvina kelionių planavimą: vartotojas gali pasirinkti šalį ir gauti rekomenduojamus miestus bei veiklas kiekviename mieste. Sistema skirta tiek paprastam lankytojui, ieškančiam kelionių idėjų, tiek registruotam vartotojui, kuris gali papildyti informaciją savo rekomendacijomis.

Veikimo principas – platforma susideda iš trijų pagrindinių objektų: **Šalis, Miestas, Veikla**.

- Pasirinkus šalį, pateikiamas miestų sąrašas.
- Pasirinkus miestą, pateikiamas siūlomų veiklų sąrašas.
- Veiklos turi pavadinimą, aprašymą ir įvertinimą.

Sistemoje yra trys rolės:

- Svečias – gali tik peržiūrėti informaciją.
- Registruotas vartotojas – gali pridėti naujas šalis, miestus ir veiklas.
- Administratorius – gali šalinti netinkamą turinį ir prižiūrėti sistemą.

1.2. Funkciniai reikalavimai

Neregistruotas sistemos naudotojas gales:

1. Peržiūrėti pagrindinį puslapį su šalių sąrašu.
2. Pasirinkti šalį ir matyti jos miestus.
3. Pasirinkti miestą ir matyti jo veiklas su vidutiniais įvertinimais.
4. Užsiregistruoti arba prisijungti prie sistemos.

Registruotas vartotojas (keliautojas) gales:

5. Atsijungti nuo sistemos.
6. Pridėti naują šalį.
7. Pridėti naują miestą pasirinktoje šalyje.
8. Pridėti naują veiklą pasirinktame mieste (su pavadinimu, aprašymu).
9. Redaguoti arba ištrinti savo šalį, miestą ar veiklą.
10. Įvertinti veiklą (pridėti balą 1–5).
11. Redaguoti arba ištrinti savo paliktą įvertinimą.

Administratorius gales:

12. Peržiūrėti visus vartotojus.

13. Pašalinti vartotoją iš sistemos.

14. Pašalinti netinkamą šalį, miestą ar veiklą.

15. Redaguoti bet kurį sistemos objektą (šalį, miestą, veiklą ar įvertinimą).

2. Sistemos architektūra

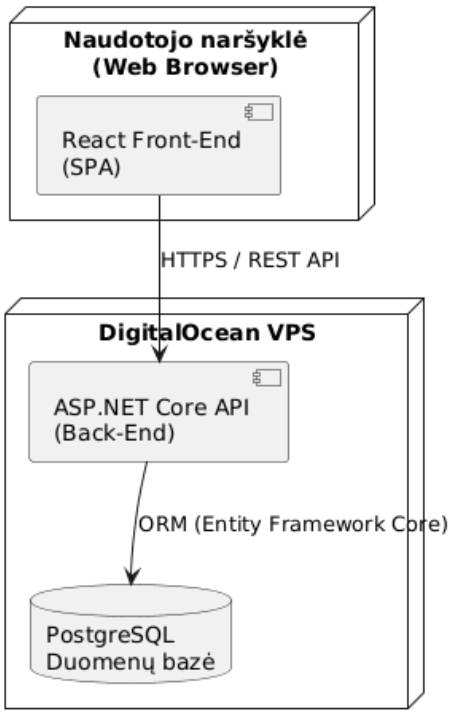
Sistemos sudedamosios dalys:

- Kliento pusė (Front-End) – sukurta naudojant React.js. Kliento pusė atsakinga už naudotojo sąsajos atvaizdavimą, vartotojo veiksmų apdorojimą ir duomenų gavimą iš serverio per REST API.
- Serverio pusė (Back-End) – realizuota naudojant ASP.NET Core (.NET Core). Serverio pusė teikia REST tipo API, vykdo verslo logiką, autentifikaciją bei autorizaciją ir apdoroja užklausas iš kliento pusės.
- Duomenų bazė – naudojama PostgreSQL reliacinė duomenų bazė. Joje saugomi sistemos duomenys: vartotojai, šalys, miestai, veiklos ir jų įvertinimai.

2.1 pav. pavaizduota kuriamos sistemos diegimo (deployment) diagrama.
Sistemos talpinimui naudojamas DigitalOcean debesijos serveris (Virtual Private Server). Kiekviena sistemos dalis – kliento pusė, serverio pusė ir duomenų bazė – yra diegiama tame pačiame serveryje.

Internetinė aplikacija naudotojams pasiekiama per HTTPS protokolą. Kliento pusės aplikacija (React.js) komunikuoja su serverio puse per REST tipo aplikacijų programavimo sąsają (API).

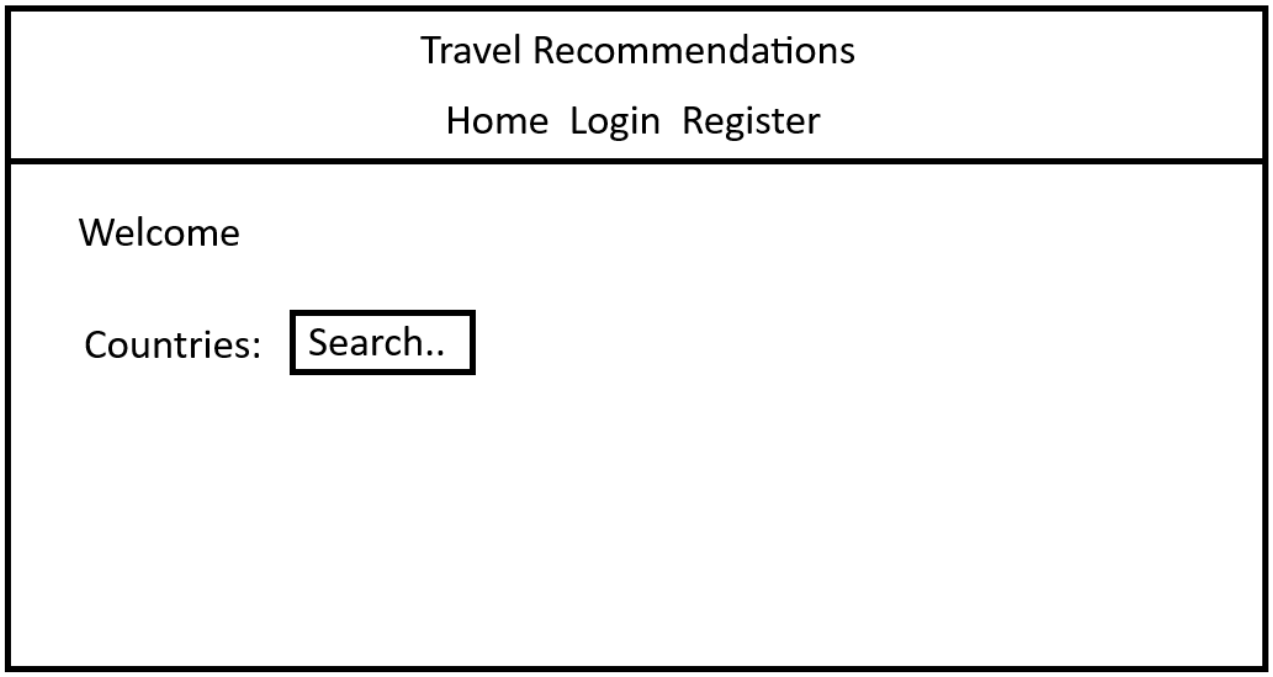
Serverio pusė (ASP.NET Core API) apdoroja gaunamas užklausas, vykdo sistemos verslo logiką ir atlieka duomenų mainus su PostgreSQL duomenų baze. Duomenų prieiga realizuota naudojant ORM (Object–Relational Mapping) sąsają – Entity Framework Core, kuri leidžia efektyviai ir saugiai dirbti su duomenų baze objektiniu principu.



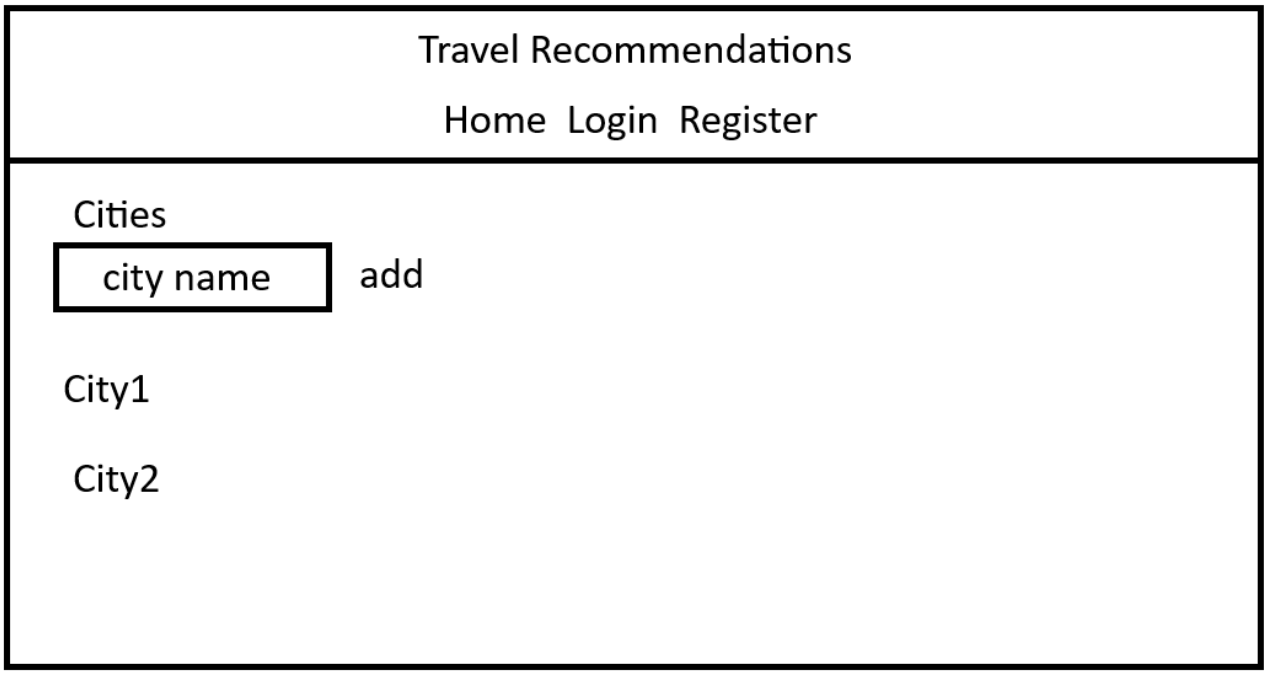
2.1 pav. Sistemos TravelRec diegimo diagrama

3. Naudotojo sąsajos projektas

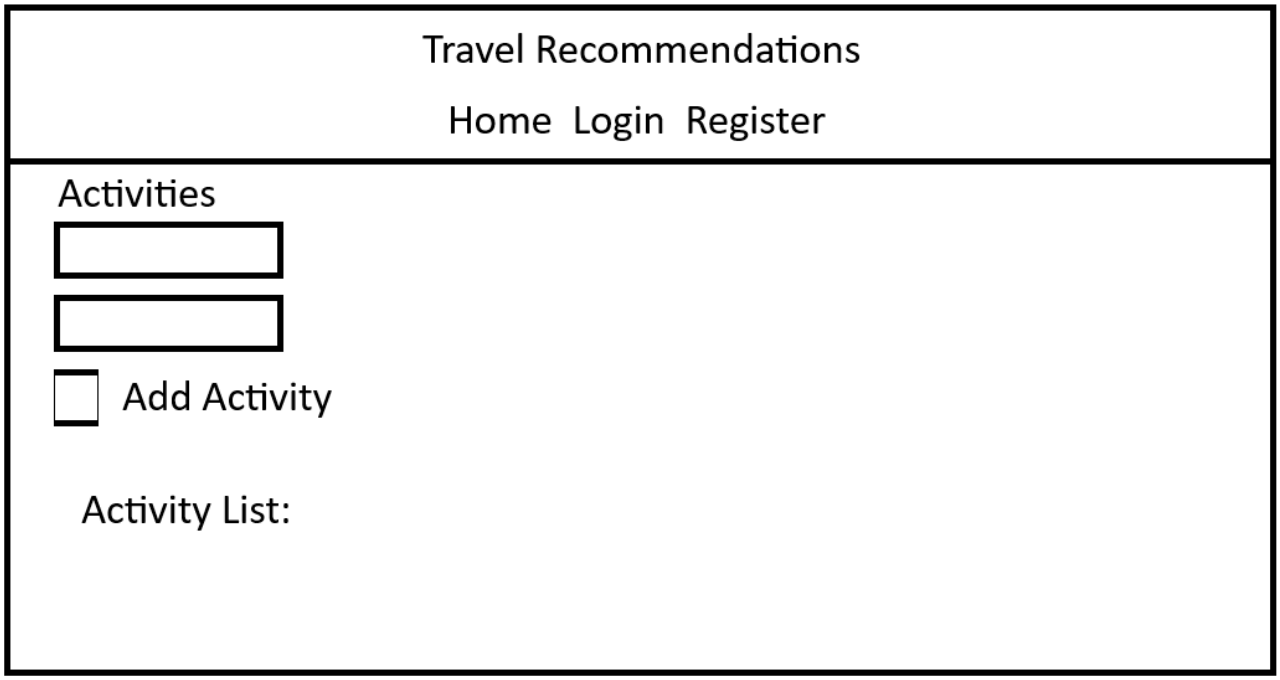
3.1. Projektuojamos sąsajos langų wireframe`ai



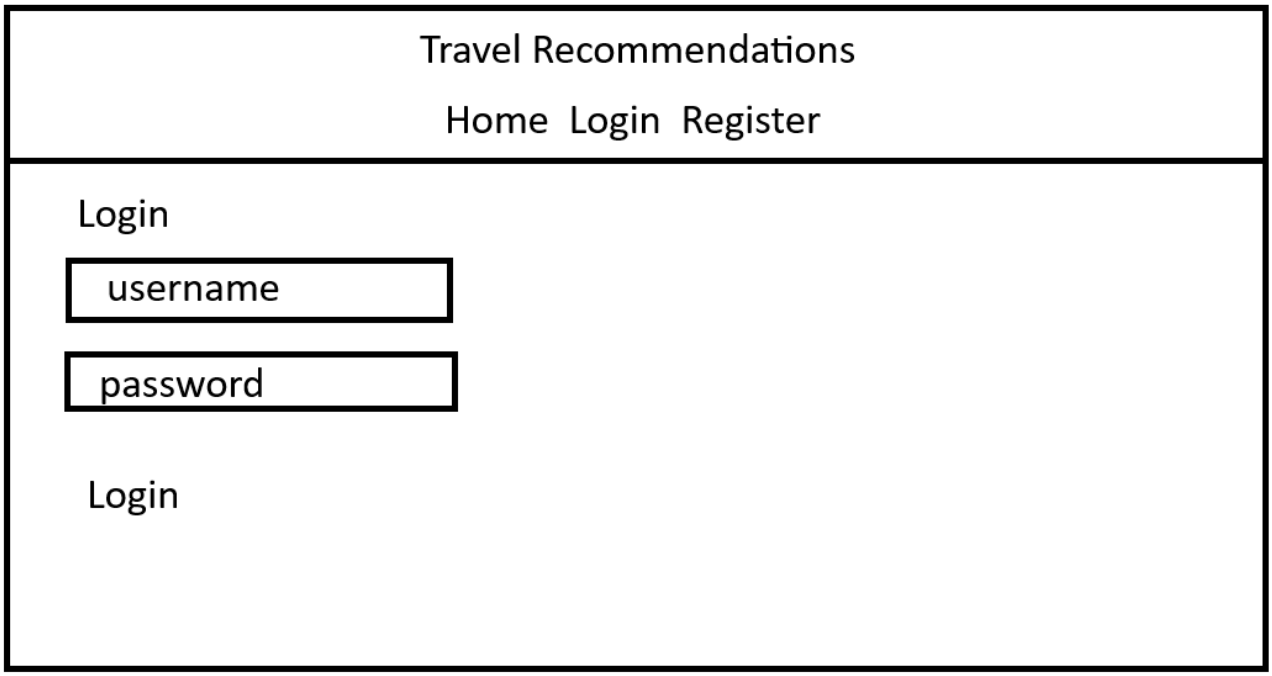
pav. 1 home wireframe



pav. 2 cities wireframe

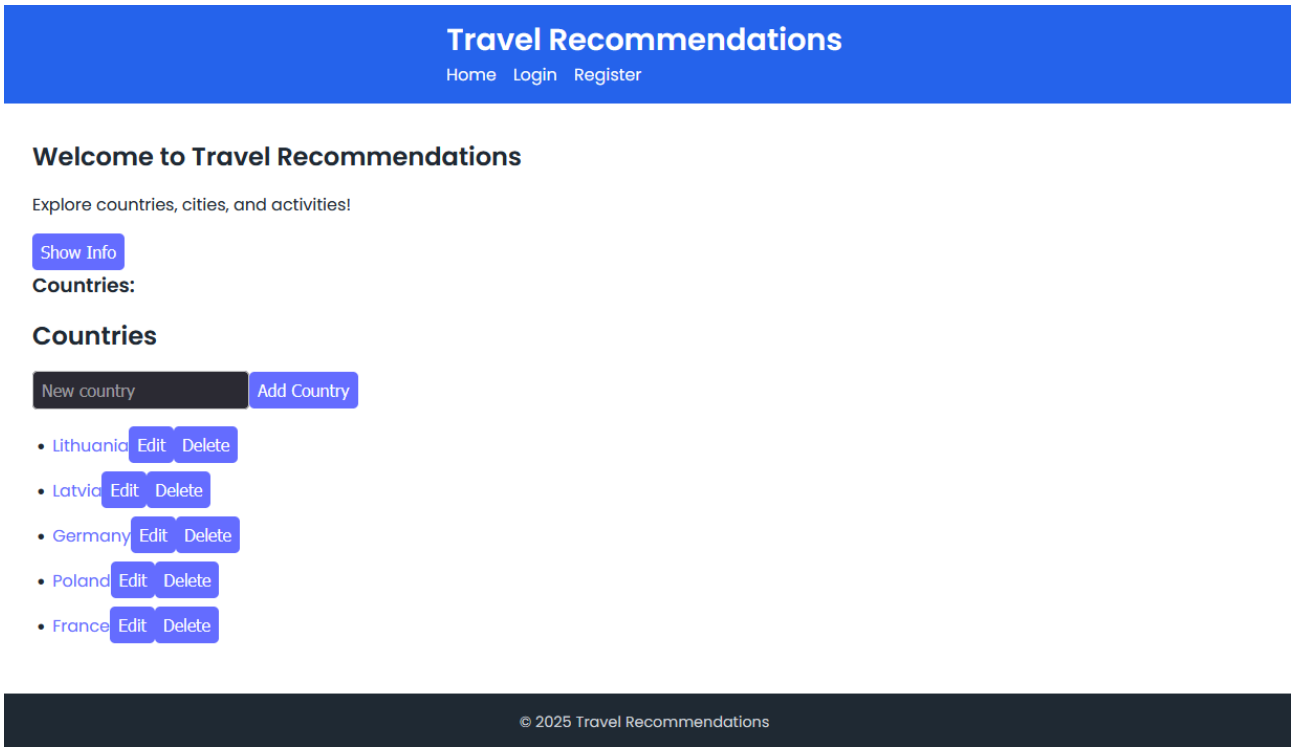


pav. 3 activities wireframe

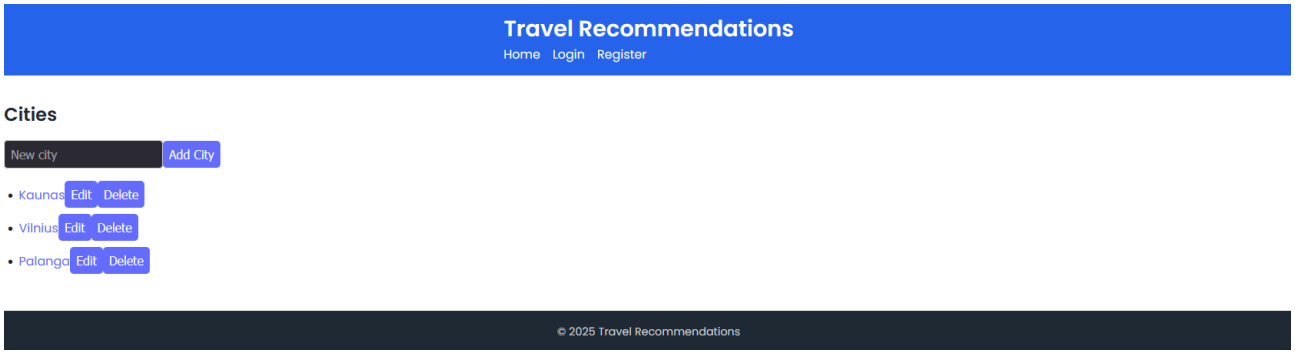


pav. 4 login wireframe

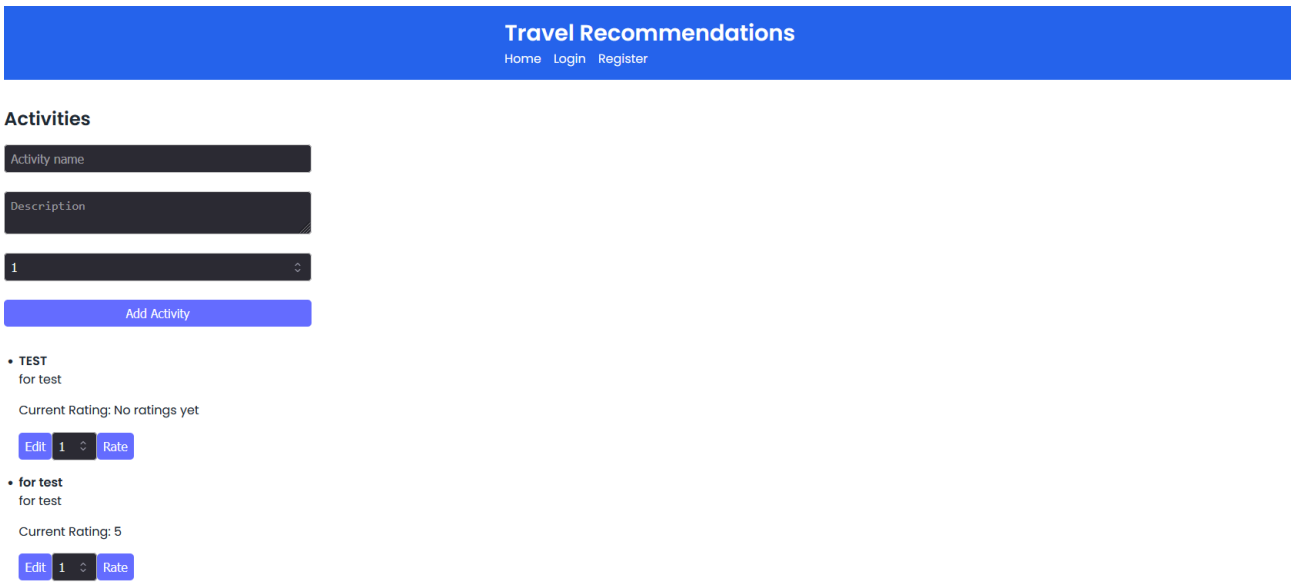
3.2. Atitinkančios realizacijos langų iškarpos



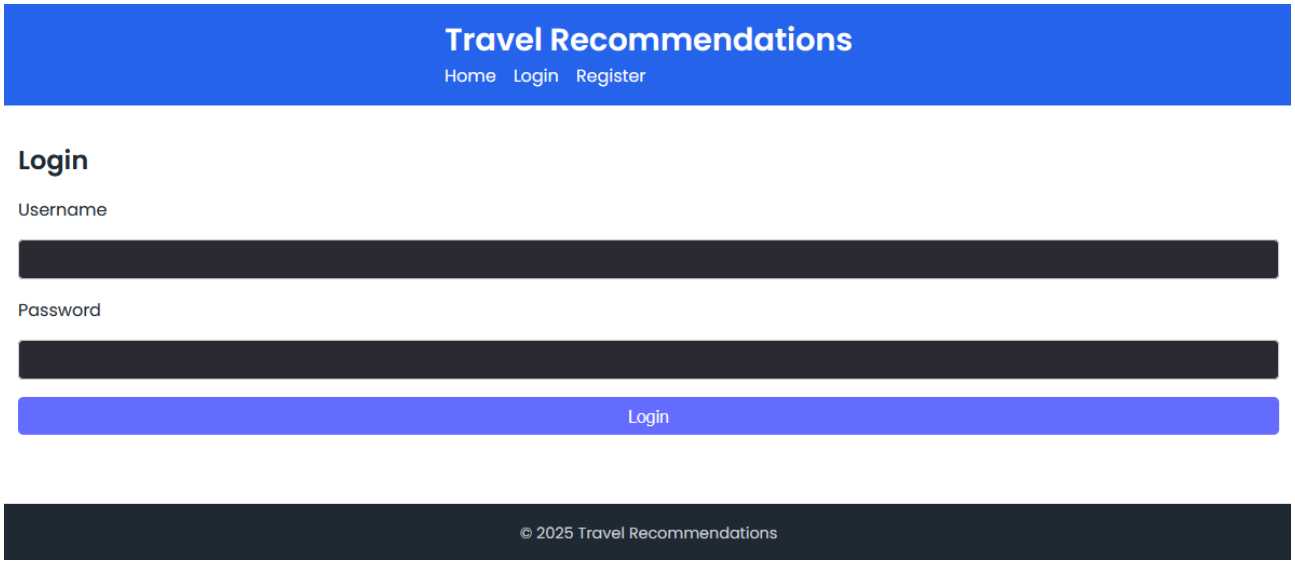
pav. 5 home lango iškarpa



pav. 6 cities lango iškarpa



pav. 7 activities lango iškarpa



pav. 8 login lango iškarpa

4. API specifikacija

TravelApp REST API yra sukurta naudojant OpenAPI 3.0.4 standartą. API leidžia front-end programai (React SPA) komunikuoti su serveriu (ASP.NET Core) ir atlikti veiksmus su šalimis, miestais bei veiklomis.
Specifikacijoje aprašyti:
Visi REST endpoint'ai:

- /api/countries – šalys
- /api/countries/{id} – konkreti šalis
- /api/countries/{countryId}/cities – miestai šalyje
- /api/countries/{countryId}/cities/{cityId} – konkretus miestas
- /api/countries/{countryId}/cities/{cityId}/activities – veiklos mieste
- /api/countries/{countryId}/cities/{cityId}/activities/{activityId} – konkreti veikla

Endpoint'ų parametrai (path, body)
Request ir response schemos: CountryDto, CityDto, ActivityDto ir jų Create/Update versijos
Galimi atsako kodai: 200 OK, 400 Bad Request, 404 Not Found
JSON pavyzdžiai, parodantys request ir response turinį
Pilna API specifikacija pateikta atskirame faile api-spec.yaml.

Link: api-spec.yaml

- **1. Countries**

GET /api/countries

- **Response codes:** 200 OK
- **Request:** nėra body
- **Response example:**

```
[
  { "id": 1, "name": "Lithuania" },
  { "id": 2, "name": "France" }
]
```

POST /api/countries

- **Response codes:** 200 OK, 400 Bad Request
- **Request example:**

```
{ "name": "Germany" }
```

- **Response example:**

```
{ "id": 3, "name": "Germany" }
```

GET /api/countries/{id}

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

```
{ "id": 1, "name": "Lithuania" }
```

PUT /api/countries/{id}

- **Response codes:** 200 OK, 400 Bad Request, 404 Not Found
- **Request example:**

```
{ "name": "Lithuania Updated" }
```

- **Response example:**

```
{ "id": 1, "name": "Lithuania Updated" }
```

DELETE /api/countries/{id}

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

```
{ "message": "Country deleted successfully." }
```

- **2. Cities**

GET /api/countries/{countryId}/cities

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

```
[
  { "id": 1, "name": "Vilnius" },
  { "id": 2, "name": "Kaunas" }
]
```

POST /api/countries/{countryId}/cities

- **Response codes:** 200 OK, 400 Bad Request, 404 Not Found
- **Request example:**

```
{ "name": "Klaipėda" }
```

- **Response example:**

{ "id": 3, "name": "Klaipėda" }

GET /api/countries/{countryId}/cities/{cityId}

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

{ "id": 1, "name": "Vilnius" }

PUT /api/countries/{countryId}/cities/{cityId}

- **Response codes:** 200 OK, 400 Bad Request, 404 Not Found
- **Request example:**

{ "name": "Vilnius Updated" }

- **Response example:**

{ "id": 1, "name": "Vilnius Updated" }

DELETE /api/countries/{countryId}/cities/{cityId}

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

{ "message": "City deleted successfully." }

- **3. Activities**

GET /api/countries/{countryId}/cities/{cityId}/activities

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

```
[
  { "id": 1, "name": "Gedimino Tower", "description": "Famous landmark", "rating": 4.5 },
  { "id": 2, "name": "Old Town Tour", "description": "Guided tour", "rating": 4.8 }
]
```

POST /api/countries/{countryId}/cities/{cityId}/activities

- **Response codes:** 200 OK, 400 Bad Request, 404 Not Found
- **Request example:**

{ "name": "National Museum", "description": "Museum of Lithuania", "rating": 5 }

- **Response example:**

{ "id": 3, "name": "National Museum", "description": "Museum of Lithuania", "rating": 5 }

GET /api/countries/{countryId}/cities/{cityId}/activities/{activityId}

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

{ "id": 1, "name": "Gedimino Tower", "description": "Famous landmark", "rating": 4.5 }

PUT /api/countries/{countryId}/cities/{cityId}/activities/{activityId}

- **Response codes:** 200 OK, 400 Bad Request, 404 Not Found
- **Request example:**

{ "name": "Gedimino Tower Updated", "description": "Landmark updated", "rating": 4.6 }

- **Response example:**

{ "id": 1, "name": "Gedimino Tower Updated", "description": "Landmark updated", "rating": 4.6 }

DELETE /api/countries/{countryId}/cities/{cityId}/activities/{activityId}

- **Response codes:** 200 OK, 404 Not Found
- **Request:** nėra body
- **Response example:**

{ "message": "Activity deleted successfully." }

5. Išvados

Šio projekto metu sukurta internetinė kelionių rekomendacijų platforma, veikianti kaip vieno puslapio aplikacija (SPA) naudojant React ir Vite. Back-end dalis sukurta su ASP.NET Core, o duomenys saugomi PostgreSQL duomenų bazėje. Serveris ir front-end bendrauja per REST API, aprašytą OpenAPI standartu.
Sistema leidžia vartotojams naršyti šalis, miestus ir veiklas bei registruoti naujas vietas. Projekto metu įgyta patirtis su front-end ir back-end integracija, duomenų bazių valdymu bei API dokumentavimu.