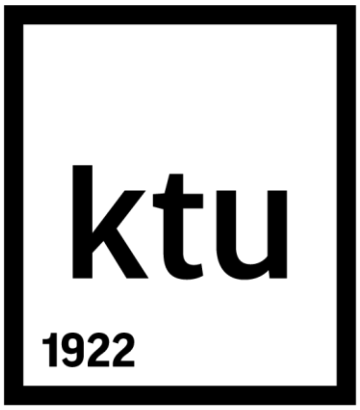
KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS



Saityno taikomųjų programų projektavimas
T120B165
Galutinė projekto ataskaita
Projektas „Naudoti stalo žaidimai“

Atliko:
Matas Suslavičius IFF 9/4

Priėmė:
Dėstytojas Tomas Blažauskas

KAUNAS 2022

Turinys

1. **Sprendžiamo uždavinio aprašymas 3**
2. **Sistemos architektūra 4**
3. **Naudotojo sąsajos projektas..... 5**
4. **API specifikacija..... 17**
5. **Išvados 22**

1. Sprendžiamo uždavinio aprašymas

1. Sistemos paskirtis

Projekto tikslas – suteikti galimybę vartotojams parduoti bei pirkti naudotus stalo žaidimus.

Veikimo principas – kuriamą platformą sudaro dvi dalys: internetinė aplikacija, kuria naudosis žmonės, norintys sukurti arba peržiūrėti stalo žaidimų skelbimus, administratorius bei aplikacijų programavimo sąsaja.

Vartotojas, norėdamas parduoti žaidimą, prisiregistruos prie internetinės aplikacijos ir galės pasirinkto stalo žaidimo puslapyje patalpinti savo naudoto žaidimo skelbimą. Vartotojas, norintis pirkti žaidimą, prisiregistruos prie internetinės aplikacijos, pasirinks norimą žaidimą ir iš atsidariusio konkretaus žaidimo skelbimų sąrašo galės pasirinkti jį domintantį skelbimą. Taip pat pirkėjas galės palikti prie skelbimo klausimą pardavėjui bei peržiūrėti kitų vartotojų užduotus klausimus. Administratorius galės šalinti skelbimus kurie yra netinkami, šalinti vartotojų paskyras kurios elgiasi piktaviliškai, trinti netinkamus klausimus.

2. Funkciniai reikalavimai

Neregistruotas sistemos naudotojas galės:

1. Peržiūrėti platformos reprezentacinį puslapį.
2. Registruotis prie internetinės aplikacijos.
3. Prisijungti prie internetinės aplikacijos.

Registruotas sistemos naudotojas galės:

1. Atsijungti nuo internetinės aplikacijos.
2. Peržiūrėti konkretaus žaidimo skelbimus.
3. Sukurti skelbimą. Sukurtą skelbimą redaguoti, šalinti.
4. Peržiūrėti konkretų skelbimą.
5. Peržiūrėti klausimus užduotus skelbimo autoriui.
6. Palikti klausimą po skelbimu. Klausimą redaguoti, šalinti.
7. Atsakyti į užduotus klausimus skelbime.

Administratorius galės:

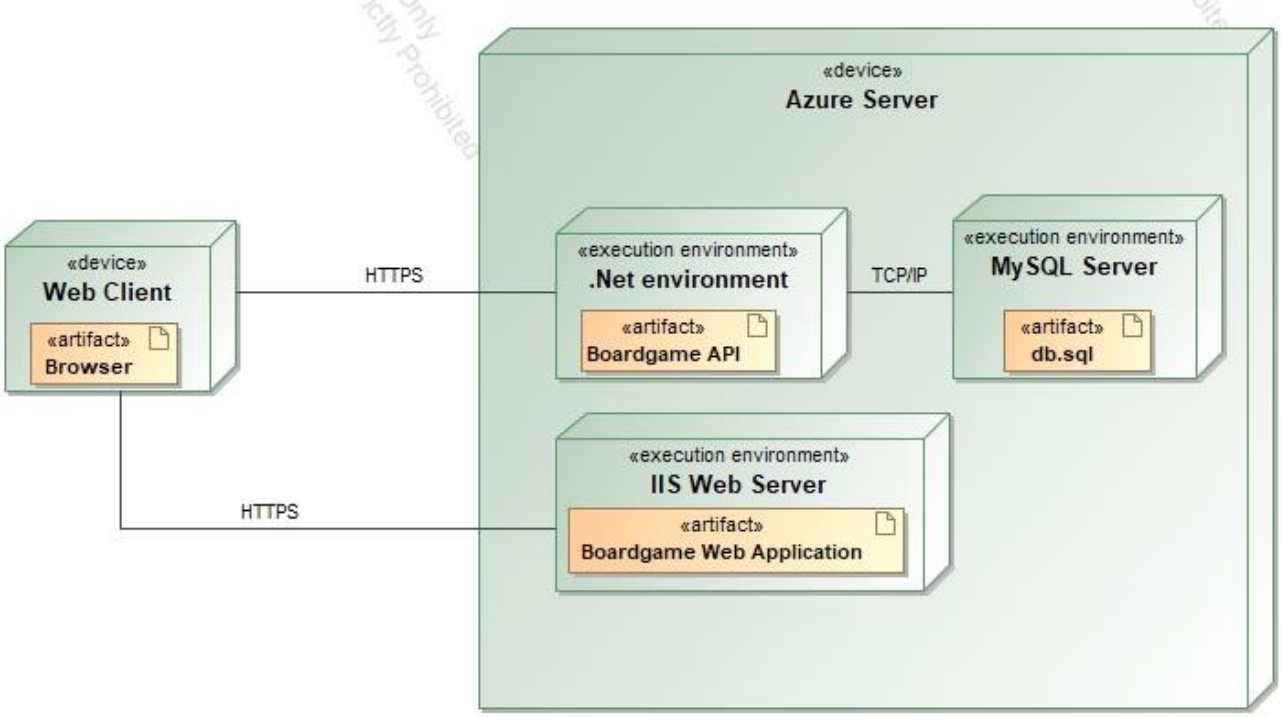
1. Šalinti naudotojus.
2. Šalinti netinkamus skelbimus.
3. Šalinti netinkamus klausimus.

2. Sistemos architektūra

Sistemos sudedamosios dalys:

- Kliento pusė (ang. Front-End) – naudojant React.js;
- Serverio pusė (angl. Back-End) – naudojant C# .NET. Duomenų bazė – Azure SQL.

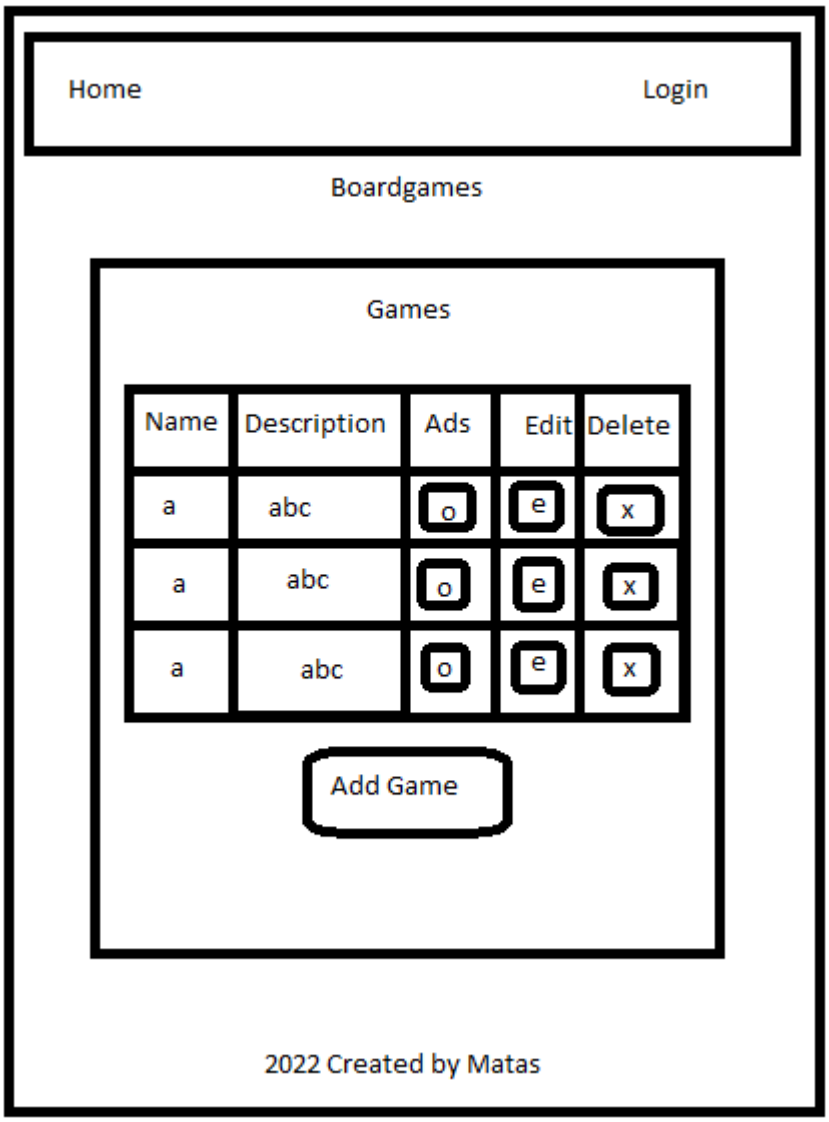
Pavaizduota kuriamos sistemos diegimo diagrama. Sistemos talpinimui yra naudojamas Azure serveris. Kiekviena sistemos dalis yra diegiama tame pačiame serveryje. Internetinė aplikacija yra pasiekiama per HTTPS protokolą. Šios sistemos veikimui yra reikalingas Boardgame API, kuris pasiekiamas per aplikacijų programavimo sąsają. API vykdo duomenų mainus su duomenų baze - tam naudojama ORM sąsaja.



3. Naudotojo sąsajos projektas

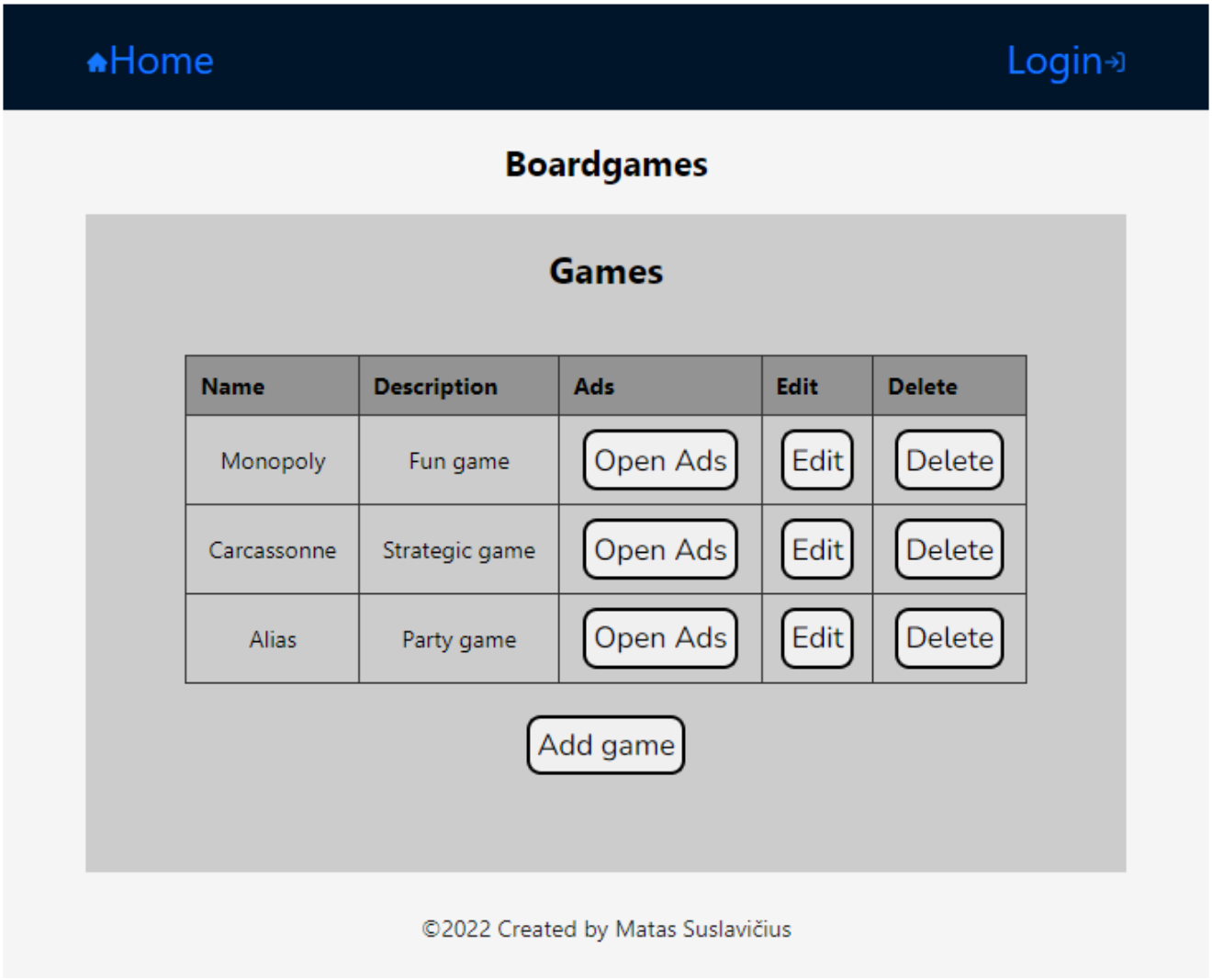
Prieš pradėdamas kurti projekto kliento pusę susiprojektavau kliento pusės sąsajos langų „wireframes“. Remdamasis šiais suprojektuotais „wireframes“ realizavau juos naudojant React.js.
Suprojektuoti langai bei jų realizacijos:

Pagrindinis puslapis – žaidimų sąrašas:



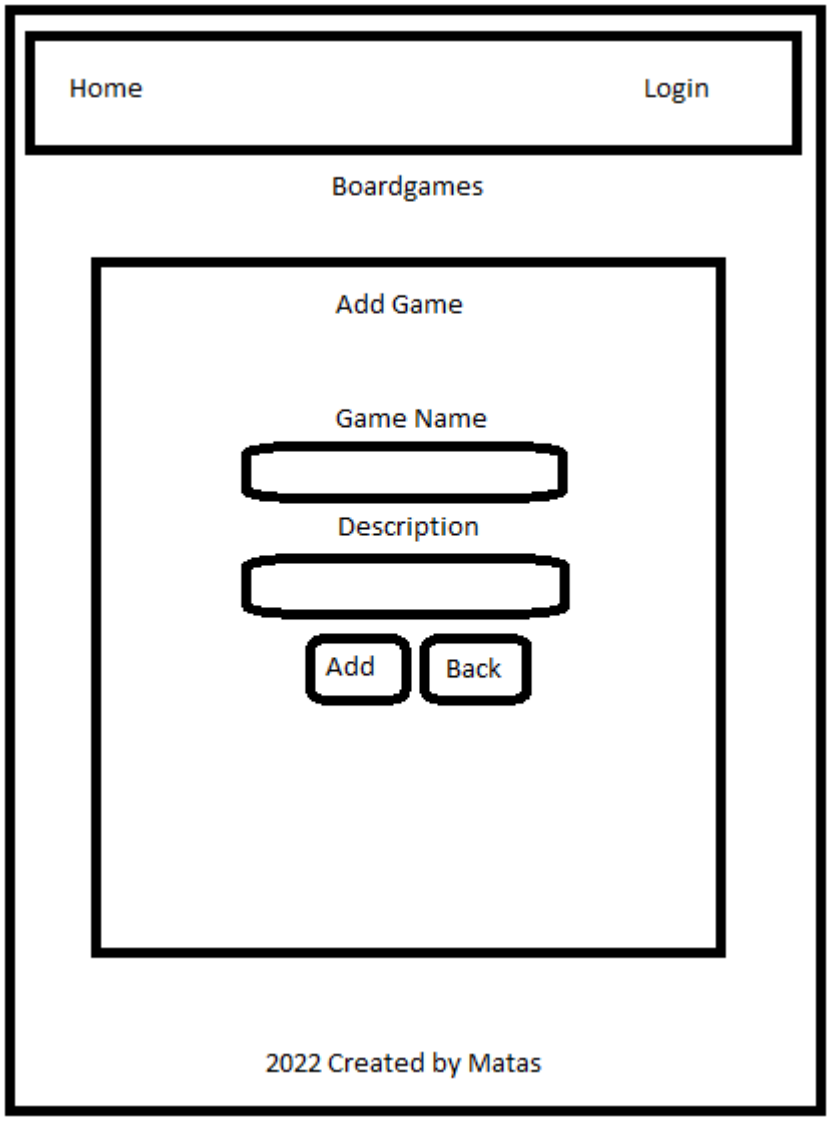
Pav. 1 pagrindinio lango wireframe

Realizuotas pagrininis puslapis:



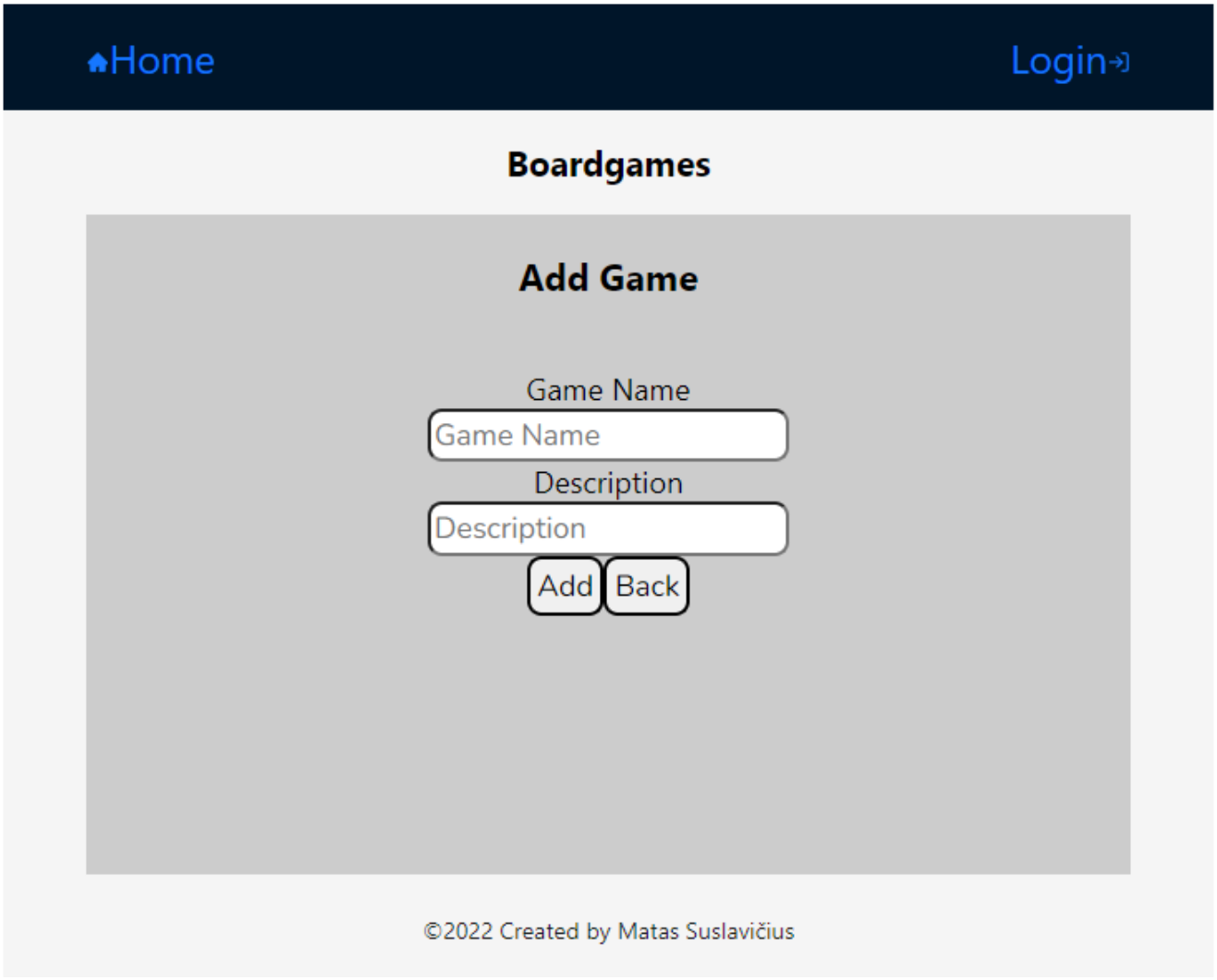
Pav. 2 realizuotas pagrindinis langas

Naujo žaidimo pridėjimo lango wireframe:



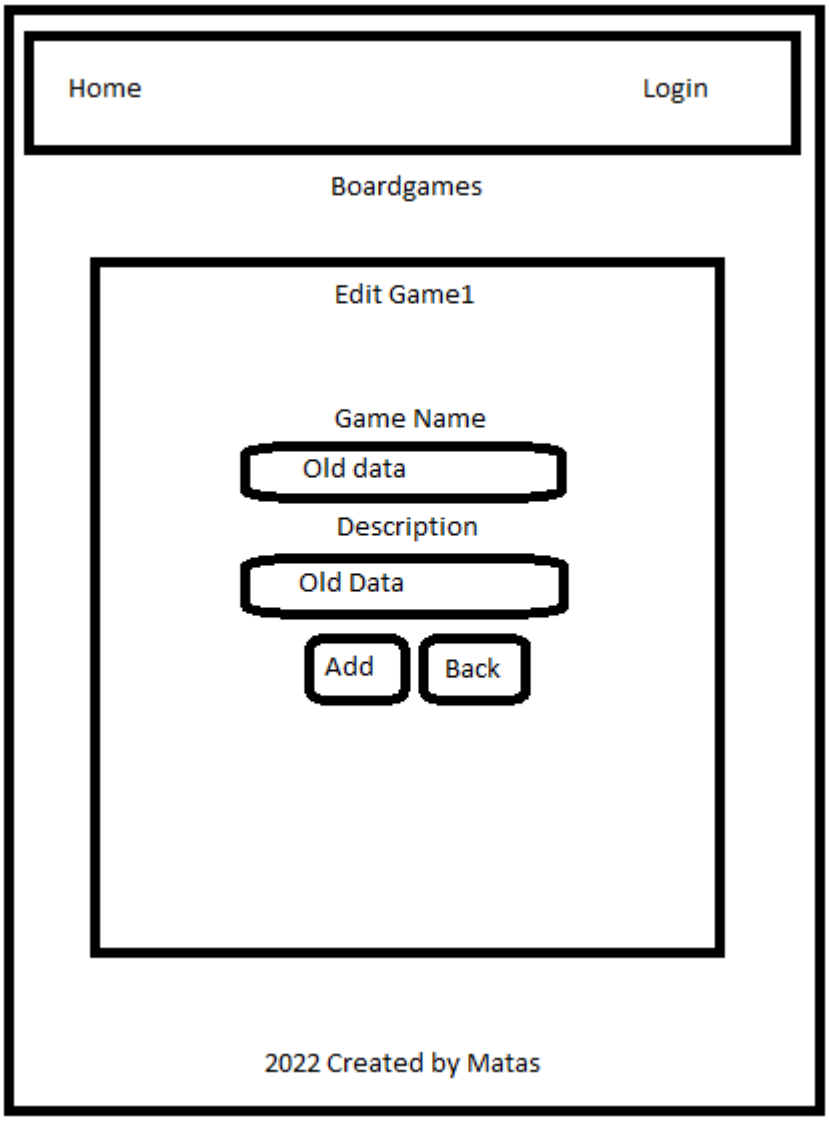
Pav. 3 Naujo žaidimo sukūrumo wireframe

Realizuotas naujo žaidimo sukūrimas:



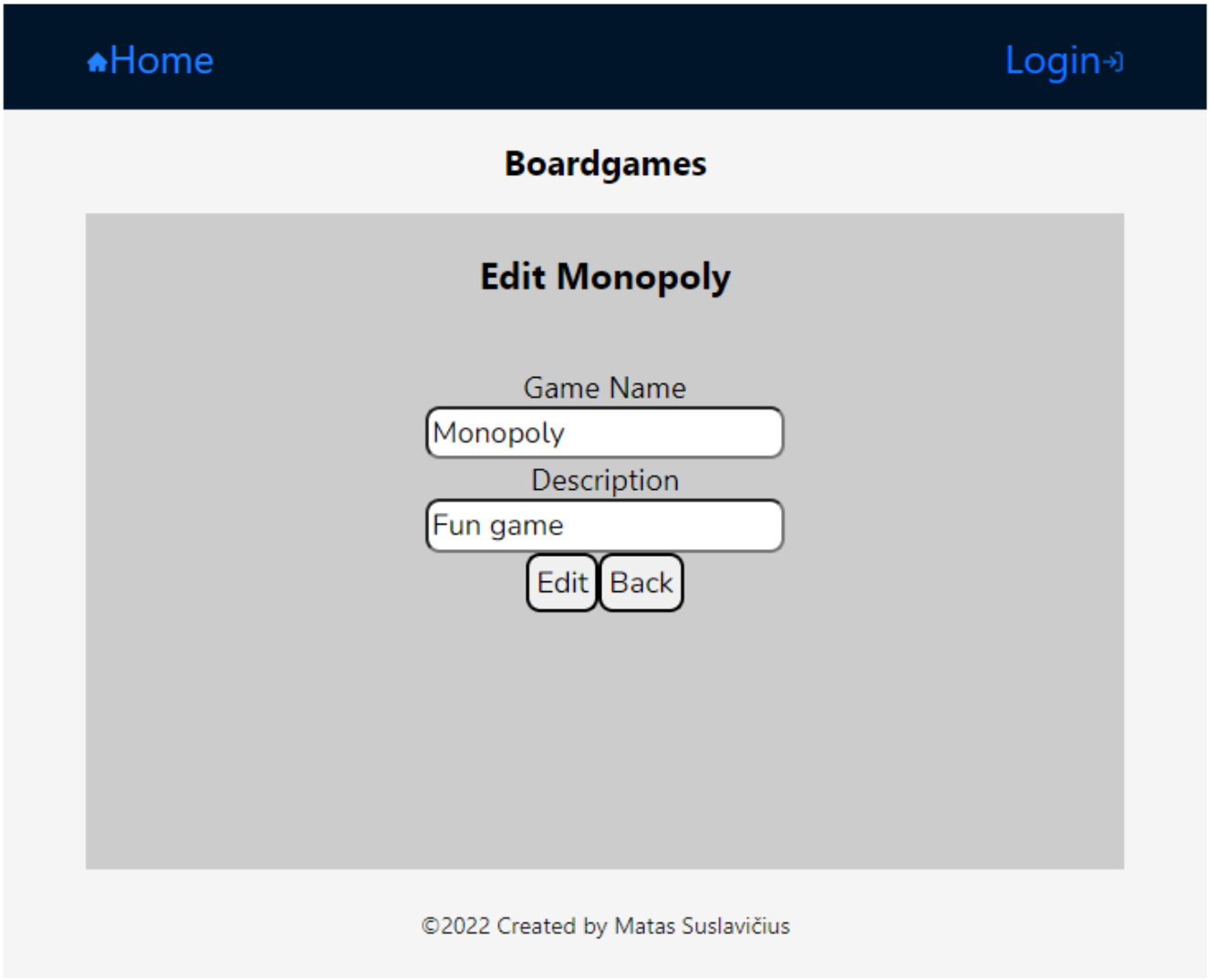
Pav. 4 Realizuotas naujo žaidimo sukūrimas

Žaidimo redagavimo lango wireframe:



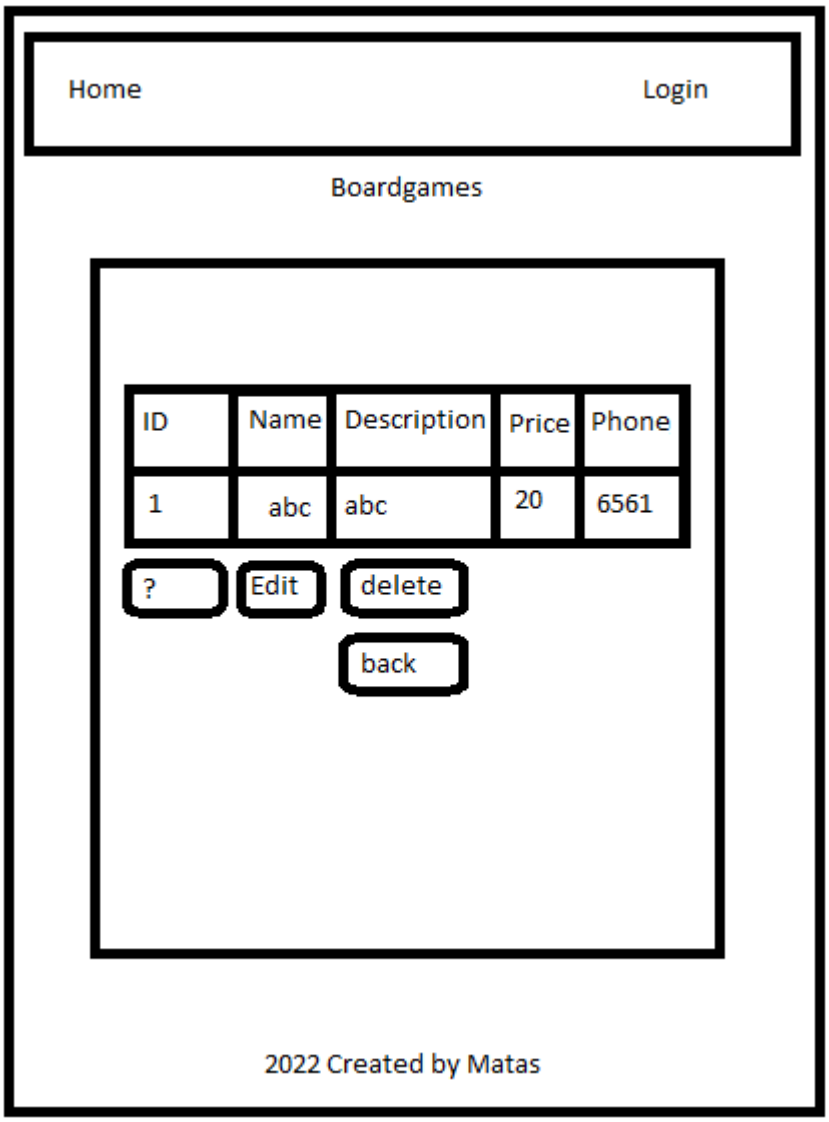
Pav. 5 Žaidimo redagavimo lango wireframe

Realizuotas žaidimo redagavimas:



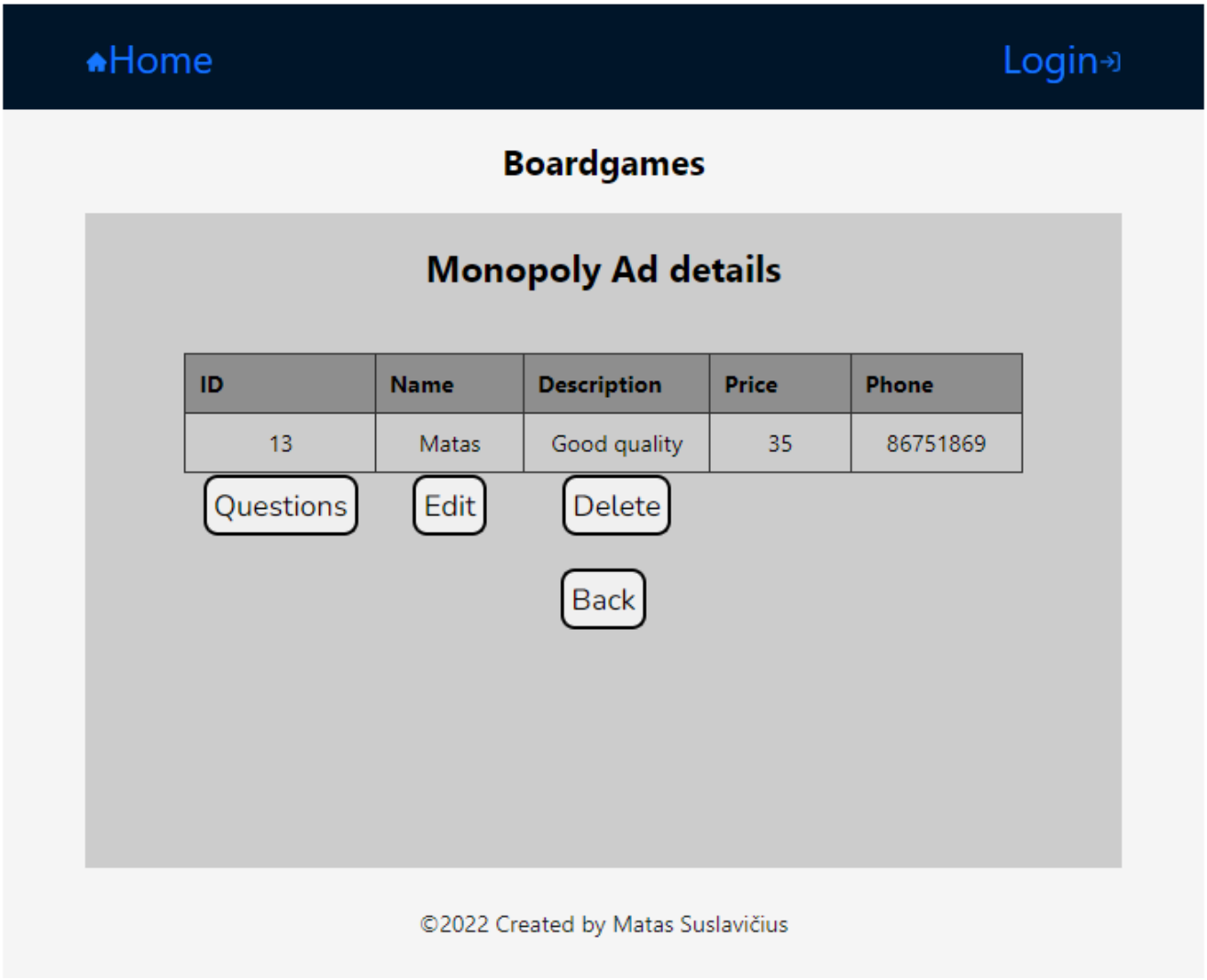
Pav. 6 realizuotas žaidimo sukūrimas

Detalaus skelbimo peržiūrėjimo lango wireframe:



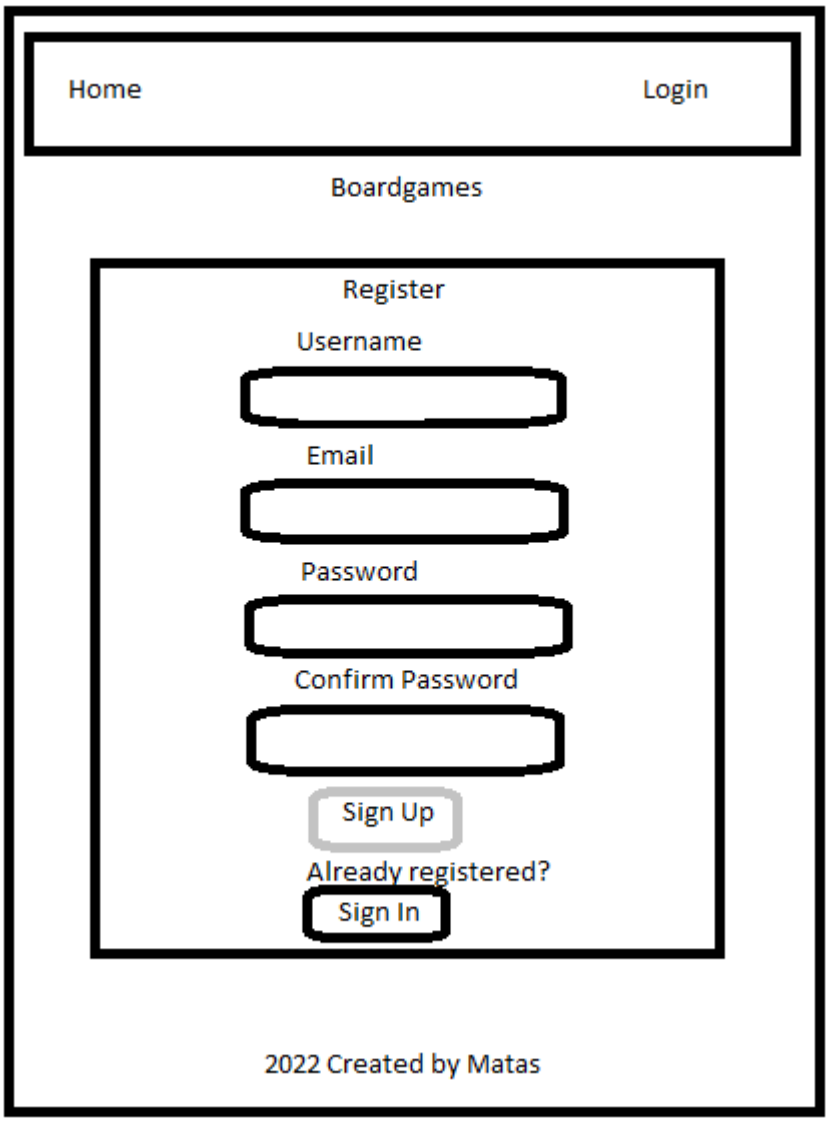
Pav. 7 Detalaus skelbimo lango wireframe

Realizuota detali skelbimo peržiūra:



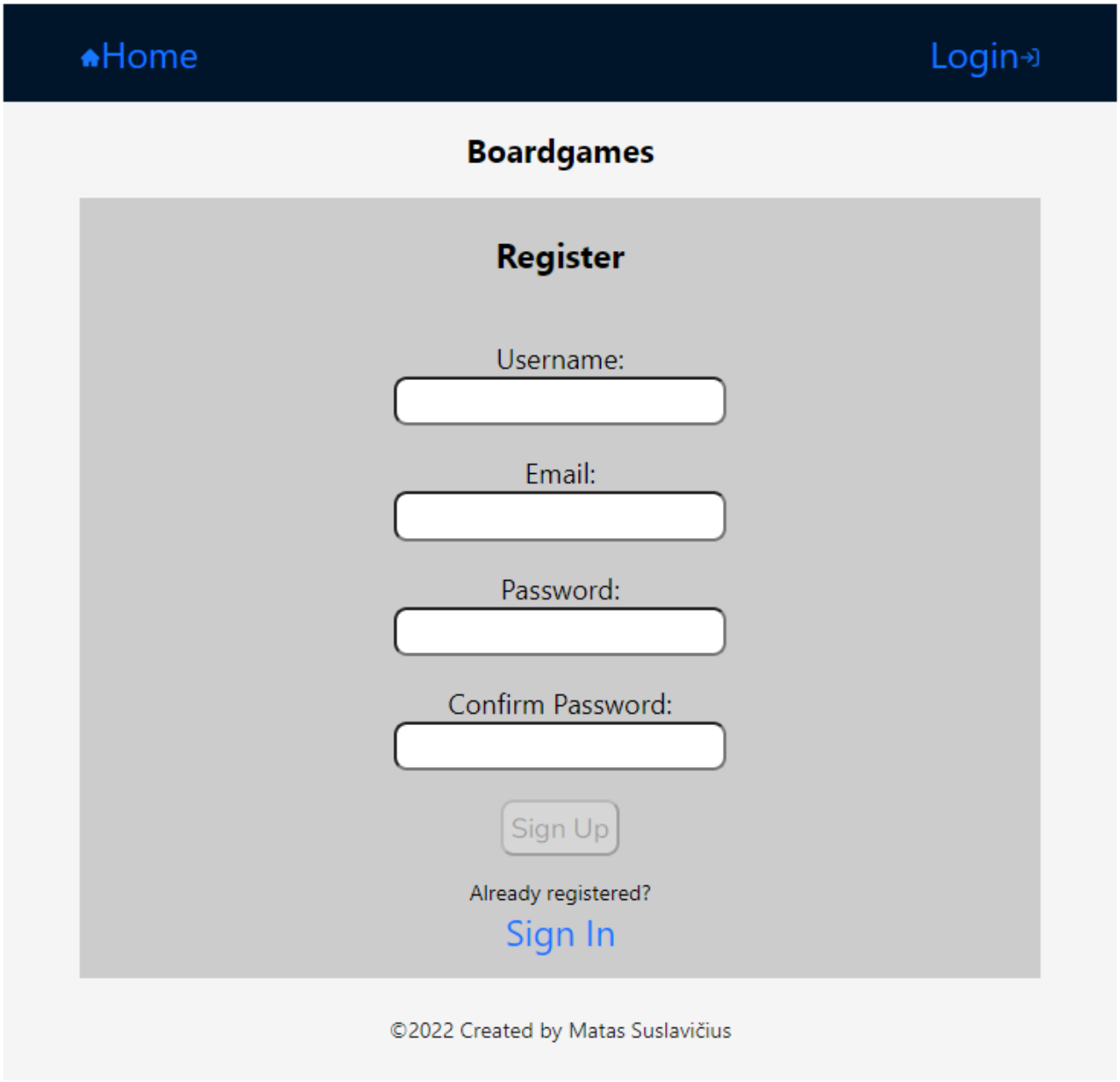
Pav. 8 Realizuota detali skelbimo peržiūra

Registracijos formos wireframe:



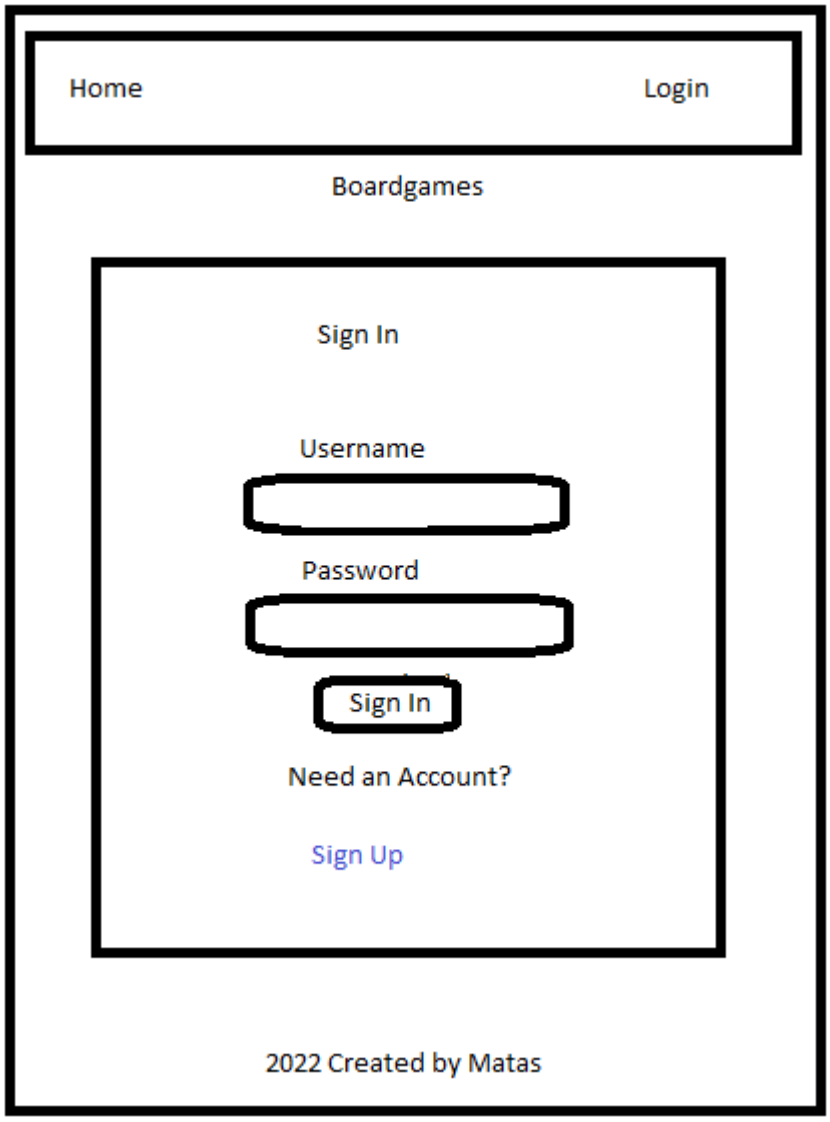
Pav. 9 registracijos formos wireframe

Realizuota registracijos forma:



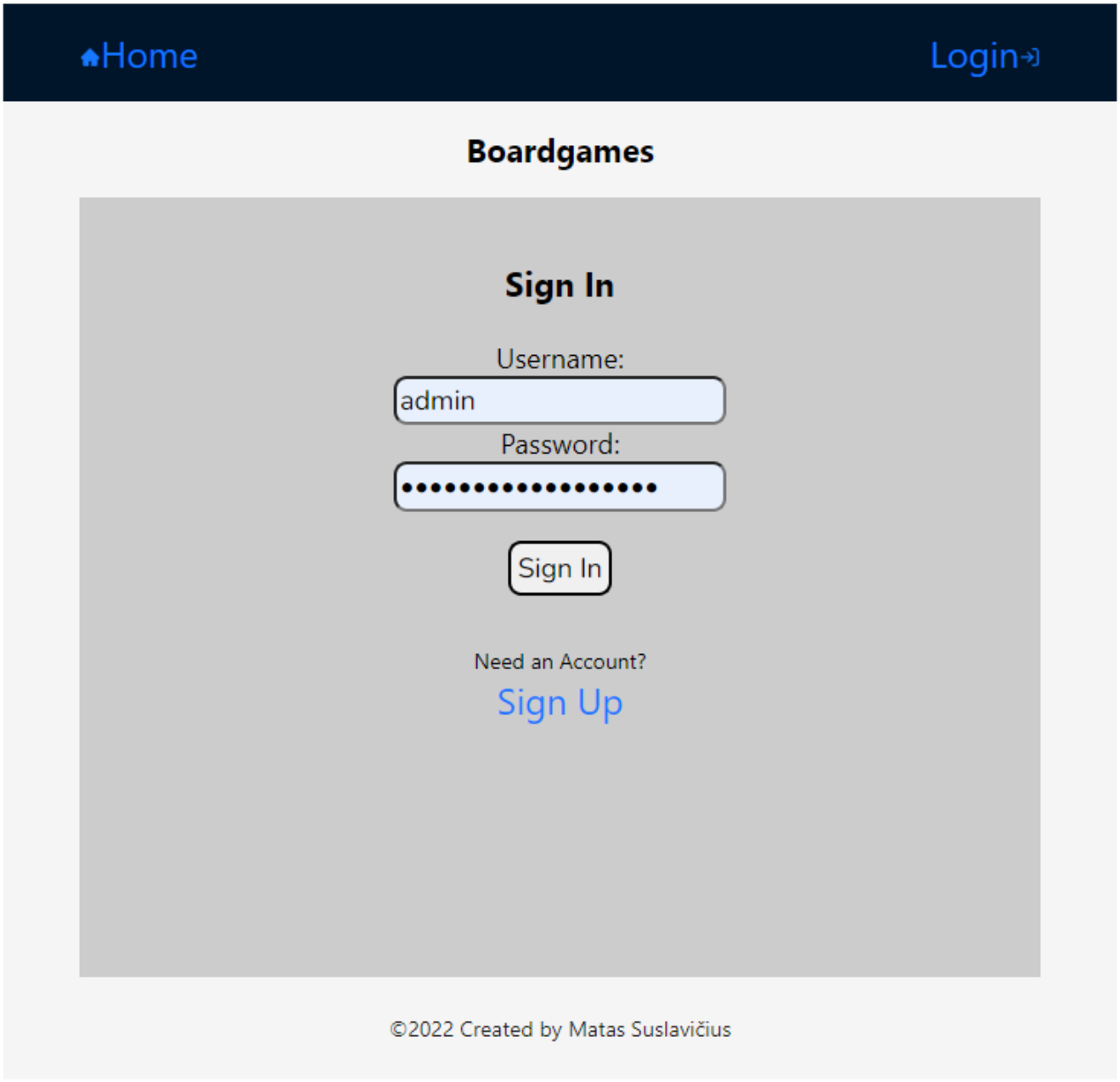
Pav. 10 Realizuota registracijos forma

Prisijungimo formos wireframe:



Pav. 11 Prisijungimo formos wireframe

Realizuota prisijungimo forma:



Pav. 12 Realizuota prisijungimo forma

4. API specifikacija

Vystant projektą buvo sukurta 17 skirtingų API metodų: 2 skirti vartotojų autentifikacijai, ir po 5 kiekvienam objektui – žaidimui, skelbimui, klausimui.

Remiantis Twitter API specifikacijos pavyždžiu apračiau vieno objekto API metodus, bei API metodus skirtus varotojų autentifikacijai:

(GET) Get All Questions

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/games/2/ads/13/questions

Resource Information

Response formats	JSON
Requires authentication?	No

Parameters

Name	Required	Description	Default Value	Example

Example Response

```
Response code - 200
[
    {
        "id": 36,
        "author": "Customer",
        "body": "Maybe cheaper?"
    },
    {
        "id": 37,
        "author": "Mark",
        "body": "Can you send to Vilnius?"
    }
]
```

(GET) Get Question

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/games/2/ads/13/questions/36

Resource Information

Response formats	JSON
Requires authentication?	No

Parameters

Name	Required	Description	Default Value	Example

Example Response

```
Response code - 200
{
    "id": 36,
    "author": "Customer",
    "body": "Maybe cheaper?"
}
```

(POST) Create Question

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/games/2/ads/13/questions

Resource Information

Response formats	JSON
Requires authentication?	Yes

Parameters

Name	Required	Description	Default Value	Example
author	required	Question creator name		`Tom`
body	required	Creator question about specific Ad		`Maybe cheaper?`

Example Body

```
{
  "author": "new question author",
  "body": "new question"
}
```

Example Response

```
Response code - 201
{
    "id": 40,
    "author": "new question author",
    "body": "new question"
}
```

(PUT) Update Question

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/games/2/ads/13/questions/40

Resource Information

Response formats	JSON
Requires authentication?	Yes

Parameters

Name	Required	Description	Default Value	Example
author	required	Question creator name		`Tom`
body	required	Creator edited question about specific Ad		`Maybe cheaper?`

Example Body

```
{
  "author": "Tom",
  "body": " Maybe cheaper?"
}
```

Example Response

```
Response code - 200
{
    "id": 40,
    "author": "Tom",
    "body": "Maybe cheaper?"
}
```

(DELETE) Delete Question

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/games/2/ads/13/questions/40

Resource Information

Response formats	JSON
Requires authentication?	Yes

Parameters

Name	Required	Description	Default Value	Example

Example Response

```
Response code - 204
```

(POST) Register

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/register

Resource Information

Response formats	JSON
Requires authentication?	No

Parameters

Name	Required	Description	Default Value	Example
userName	Required	Username of the user who is doing the registration.		Matas3
email	Required	Email of the user who is doing the registration.		matas@matas.com
password	Required	Password of the user who is doing the registration. Must contain lowercase letter, uppercase letter, number, special symbol		Verystrong1!

Example Body

```
{
    "userName" : "Matas3",
    "email" : "matas@matas.com",
    "password" : "Verystrong1!"
}
```

Example Response

```
Response code - 201
{
    "id": "a892b8e2-88e9-472d-b002-302fb945a91d",
    "userName": "Matas3",
    "email": "matas@matas.com"
}
```

(POST) Login

Resource URL:
https://boardgamesapi1.azurewebsites.net/api/login

Resource Information

Response formats	JSON
Requires authentication?	No

Parameters

Name	Required	Description	Default Value	Example
userName	Required	Username of the user who wants to log in.		Matas3
password	Required	Password of the user who wants to log in.		Verystrong1!

Example Body

```
{
    "userName" : "Matas3",
    "password" : "Verystrong1!"
}
```

Example Response

```
Response code - 200
{
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1lIjoiTWF0YXMzIiwianRpIjoiOWZmYzUzZTQtM2M5MC00ODNkLTkxNGUtYTUyNWVjZGJjY2Q5IiwidXNlcklkIjoiYTg5MmI4ZTItODhlOS00NzJkLWIwMDItMzAyZmI5NDVhOTFkIiwiaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2xlIjoiU2ltcGxlVXNlciIsImV4cCI6MTY3MTE0Nzg5NSwiaXNzIjoiTWF0YXMiLCJhdWQiOiJUcnVzdGVkQ2xpZW50In0.GtVqHI7MbegVy53Wc5gc4Rgb8mUIOrILotC_BIgCU38"
}
```

Realizuotų API metodų galimi atsako kodai:

Metodas	Galimi atsakymo kodai
GET	200
POST	200, 201, 400, 401
PUT	200, 400, 401
DELETE	201, 401

5. Išvados

Atliekant projektą buvo įgytos žinios apie serverio (Back-end) kūrimą naudojant C# .NET Core 7. Taip pat buvo praplėstos žinios apie kliento pusės realizaciją (Front-end) naudojant React.js. Kurtas projektas buvo sėkmingai patalpintas į saityną panaudojus „Azure“ debesų technologijas. Projektui sėkmingai pavyko pritaikyti vartotojų autentifikaciją ir autorizaciją naudojant JWT technologinį sprendimą. Realizuojant projektą buvo sukurta 17 API metodų ir kiekvienas iš jų pritaikytas naudotojo sąsajos sprendime.