



Kauno Technologijos Universitetas

Informatikos Fakultetas

T120B565 SAITYNO TAIKOMŲJŲ PROGRAMŲ PROJEKTAVIMAS

Inžinerinis projektas

Darbą atliko:

Ignas Pūras, IFF-1/8

Dėstytojai:

Lekt. Kiudys Eligijus

Lekt. Prakt. Baltulionis Simonas

Kaunas, 2024

1 Turinys

1	Turinys	2
2	Paveikslėlių sąrašas	3
3	Sprendžiamo uždavinio aprašymas	4
3.1	Sistemos paskirtis	4
3.2	Funkciniai reikalavimai	4
4	Sistemos architektūra	5
4.1	Sistemos sudedamosios dalys	5
5	Naudotojo sąsaja	6
5.1	Pagrindinis puslapis	6
5.2	Registravimosi puslapis	7
5.3	Prisijungimo puslapis	7
5.4	Pagrindinis puslapis prisijungus	8
5.5	Temų sąrašų puslapis	9
5.6	Temos detalių puslapis	11
5.7	Pranešimo detalių puslapis	12
6	API specifikacija	15
6.1	Topics API	15
6.2	Posts API	18
6.3	Comments API	21
6.4	Authentication API	25
7	Išvados	27

2 Paveikslėlių sąrašas

pav. 1 UML deployment diagrama	5
pav. 2 Pagrindinis forumo puslapis	6
pav. 3 registravimosi puslapis	7
pav. 4 prisijungimo puslapis	7
pav. 5 pagrindinis puslapis prisijungus	8
pav. 6 temų peržiūrėjimo puslapis	9
pav. 7 Temos sukūrimo puslapis	10
pav. 8 Temos detalių puslapis	11
pav. 9 Temos redagavimo puslapis	11
pav. 10 Pranešimo detalių puslapis	12
pav. 11 Pranešimo sukūrimo puslapis	12
pav. 12 Komentaro pridėjimo puslapis	13
pav. 13 Komentaro redagavimo puslapis	13
pav. 14 Pranešimo redagavimo puslapis	14
pav. 15 GET /api/topics	15
pav. 16 GET /api/topics/{topicId}	16
pav. 17 POST /api/topics	16
pav. 18 DELETE /api/topics/{topicId}	17
pav. 19 PUT /api/topics/{topicId}	17
pav. 20 POST /api/topics/{topicId}/posts	18
pav. 21 GET /api/topics/{topicId}/posts	18
pav. 22 PUT /api/topics/{topicId}/posts/{postId}	19
pav. 23 GET /api/topics/{topicId}/posts/{postId}	19
pav. 24 DELETE /api/topics/{topicId}/posts/{postId}	20
pav. 25 POST /api/topics/{topicId}/posts/{postId}/comments	21
pav. 26 GET /api/topics/{topicId}/posts/{postId}/comments	21
pav. 27 DELETE T /api/topics/{topicId}/posts/{postId}/comments/{commentId}	22
pav. 28 PUT /api/topics/{topicId}/posts/{postId}/comments/{commentId}	23
pav. 29 GET /api/topics/{topicId}/posts/{postId}/comments/{commentId}	24
pav. 30 POST /api/accounts	25
pav. 31 POST /api/login	25
pav. 32 POST /api/accessToken	26
pav. 33 POST /api/logout	26
pav. 34 GET /api/accounts/{userId}	27

3 Sprendžiamo uždavinio aprašymas

3.1 Sistemos paskirtis

Sistemos yra internetinis forumas, kuriame žmonės gali dalintis savo įpročių gerinimo ar kitais planais susijusiais su sportu, mityba, asmeniniu tobulėjimu, mokymosi ir t.t. Vartotojai turės galimybę kurti Temas, sukurtom temom kurti pranešimus, o pranešimuose parašyti komentarus. Panaikinus pranešimą, dingsta ir visi komentarai tame pranešime. Panaikinus temą taip pat panaikinami visi pranešimai ir komentarai esantys temoje.

3.2 Funkciniai reikalavimai

Vartotojai gali:

- Registruoti/susikurti naują paskyrą ir su ja prisijungti prie puslapio;
- Kurti Temas, sukurtom temom kurti pranešimus, o pranešimuose rašyti komentarus;
- Redaguoti savo sukurtas temas ir įkeltus pranešimus, parašytus komentarus;
- Trinti savo sukurtas temas, pranešimus, komentarus.

Administratorius gali:

- Viską ką ir vartotojas;
- Gali panaikinti ne tik savo bet ir paprastų vartotojų temas, pranešimus, komentarus;
- Gali redaguoti ne tik savo bet ir paprastų vartotojų temas, pranešimus, komentarus.

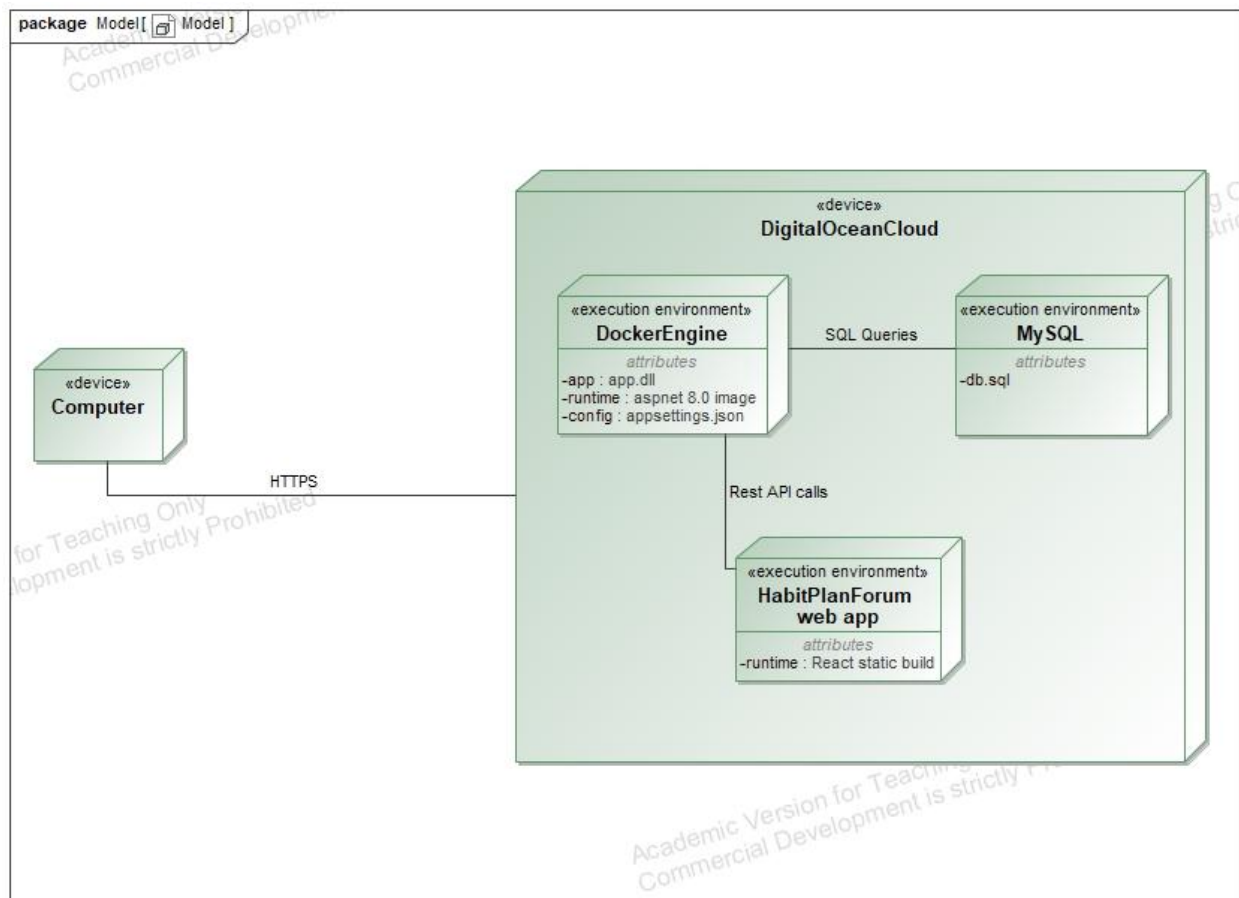
4 Sistemos architektūra

4.1 Sistemos sudedamosios dalys

- Kliento pusė – kuriama naudojant JavaScript React.js biblioteką;
- Serverio pusė – kuriama su .net c# entity framework;
- Duomenų bazė – MySQL.

Programa buvo talpinama puslapyje (<https://www.digitalocean.com/>). Per šį puslapį yra paleista duomenų bazė, kliento pusė ir serverio pusė. Kodas pasiekiamas GitHub saugykloje:

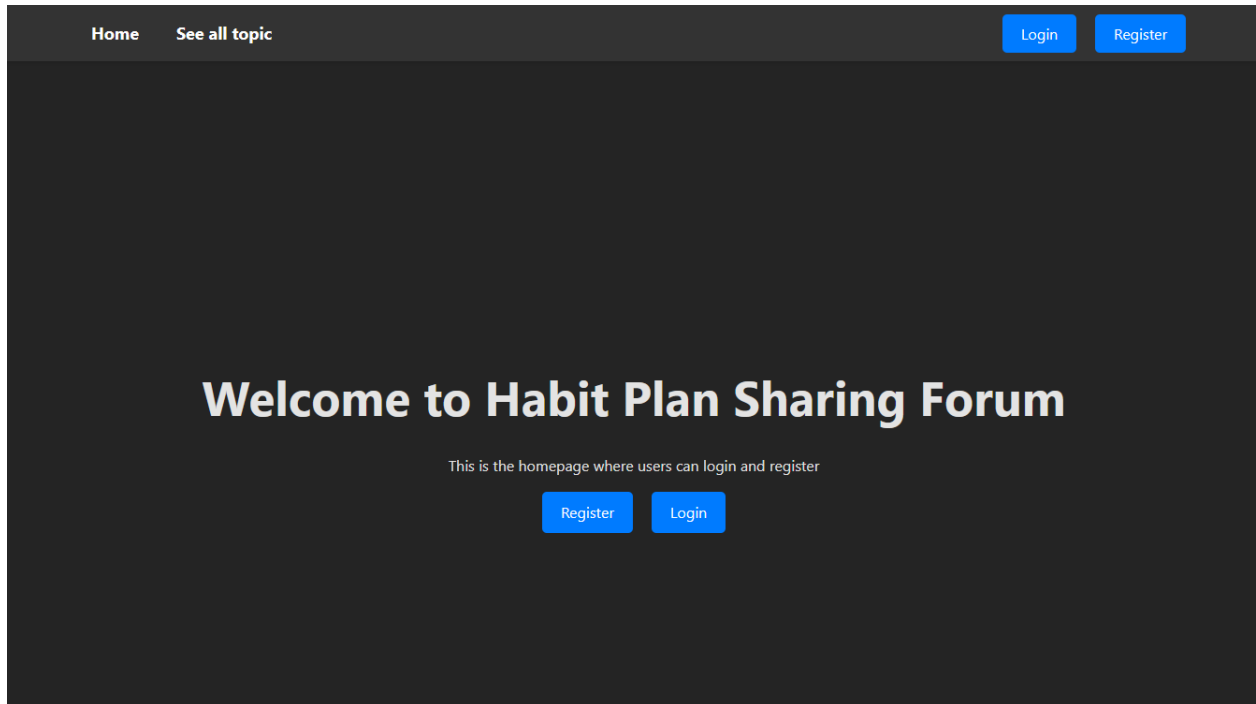
<https://github.com/ignpur/Habbit-Plan-Forum>.



pav. 1 UML deployment diagrama

5 Naudotojo sąsaja

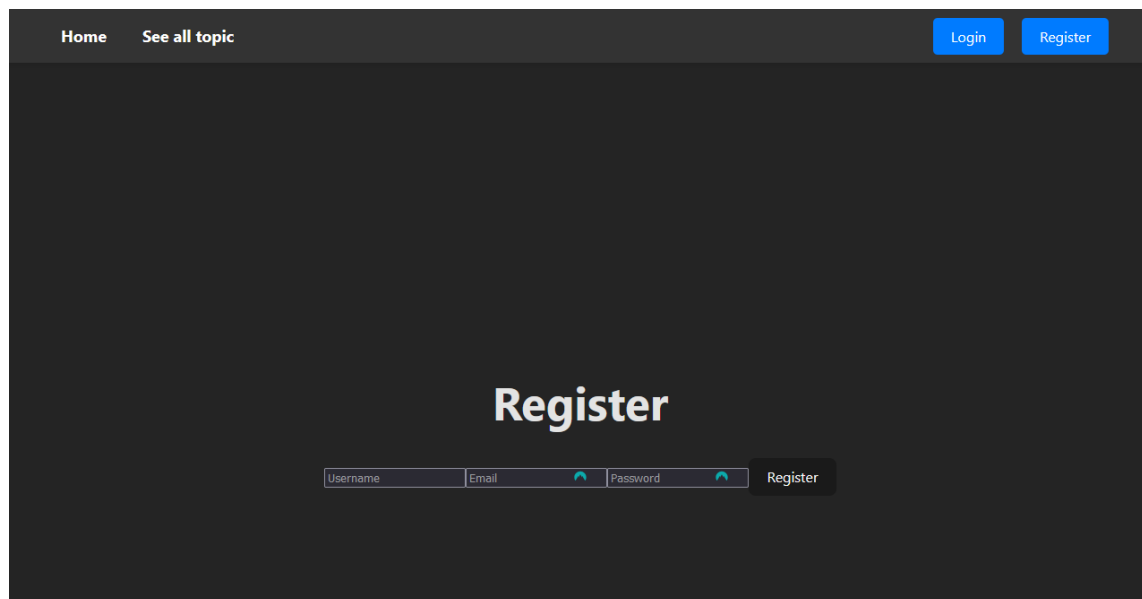
5.1 Pagrindinis puslapis



pav. 2 Pagrindinis forumo puslapis

Pagrindiniame puslapyje vartotojai gali susikurti paskyra arba prisijungti.

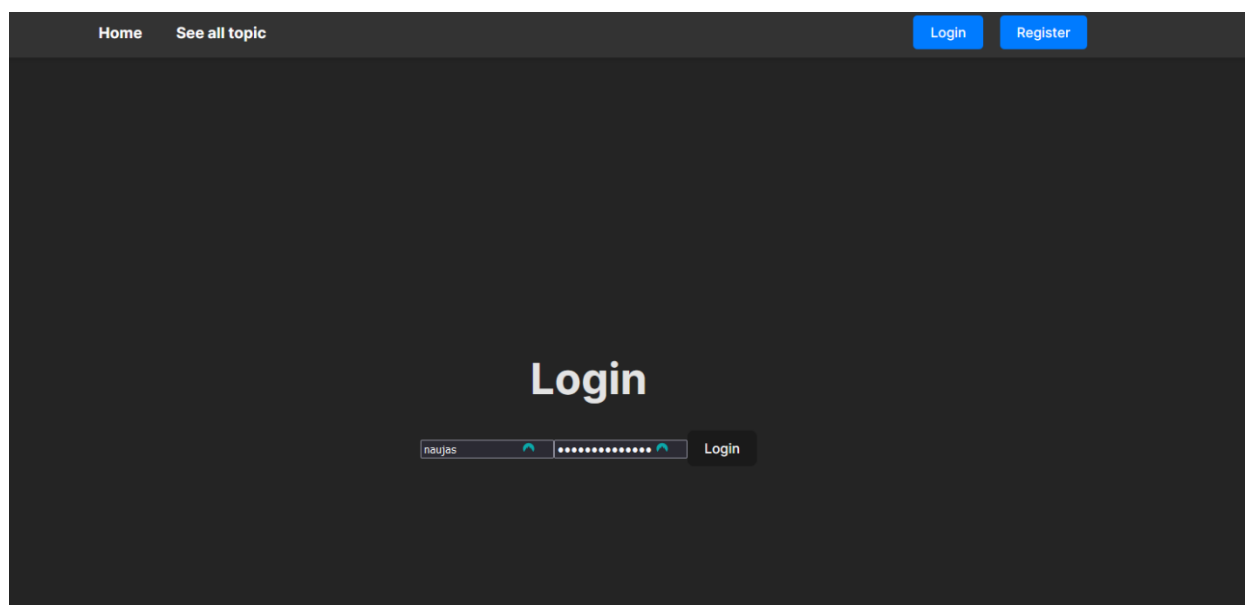
5.2 Registravimosi puslapis



pav. 3 registravimosi puslapis

Registravimosi puslapyje nauji vartotojai turi būtinai užpildyti visus laukelius ir tada paspaudus mygtuką „register“ gali registruotis.

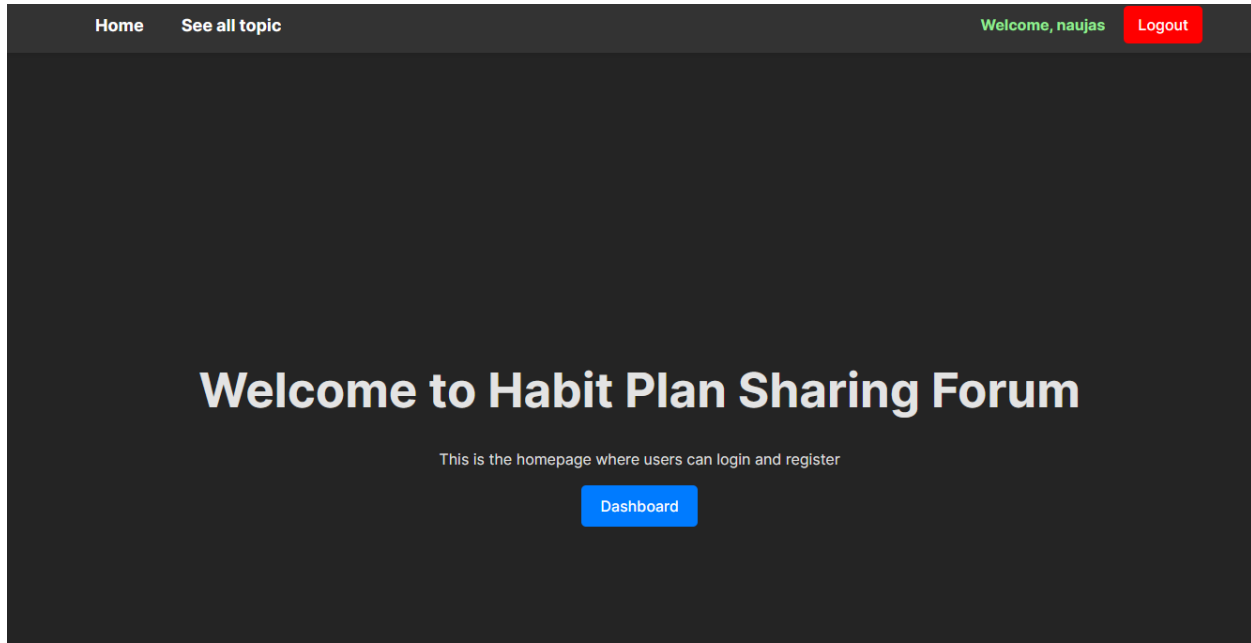
5.3 Prisijungimo puslapis



pav. 4 prisijungimo puslapis

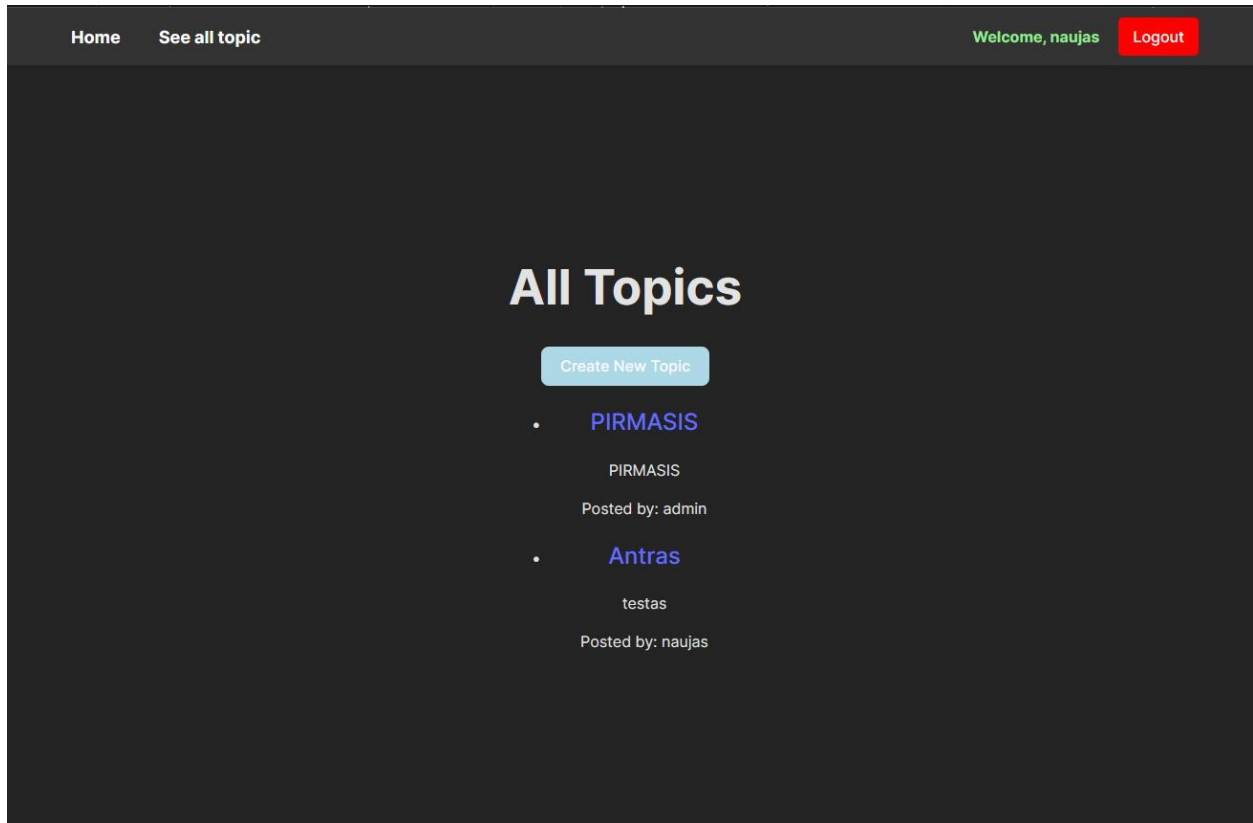
Prisijungimo puslapyje vartotojai ar administratorius turi suvesti savo prisijungimo vardą ir slaptažodį ir paspaudus mygtuką „login“ gali prisijungti prie forumo.

5.4 Pagrindinis puslapis prisijungus



pav. 5 pagrindinis puslapis prisijungus

5.5 Temų sąrašų puslapis



pav. 6 temų peržiūrėjimo puslapis

Šiame puslapyje prisijungę vartotojai gali paspaudę mygtuką „Create New Topic“ sukurti naują temą. O paspaudę ant temos pavadinimo (pvz.: „PIRMASIS“) bus atverčiamas tos temos puslapis, kuriame galima matyti visus sukurtus pranešimus pagal tą temą.

Create a New Topic

Title

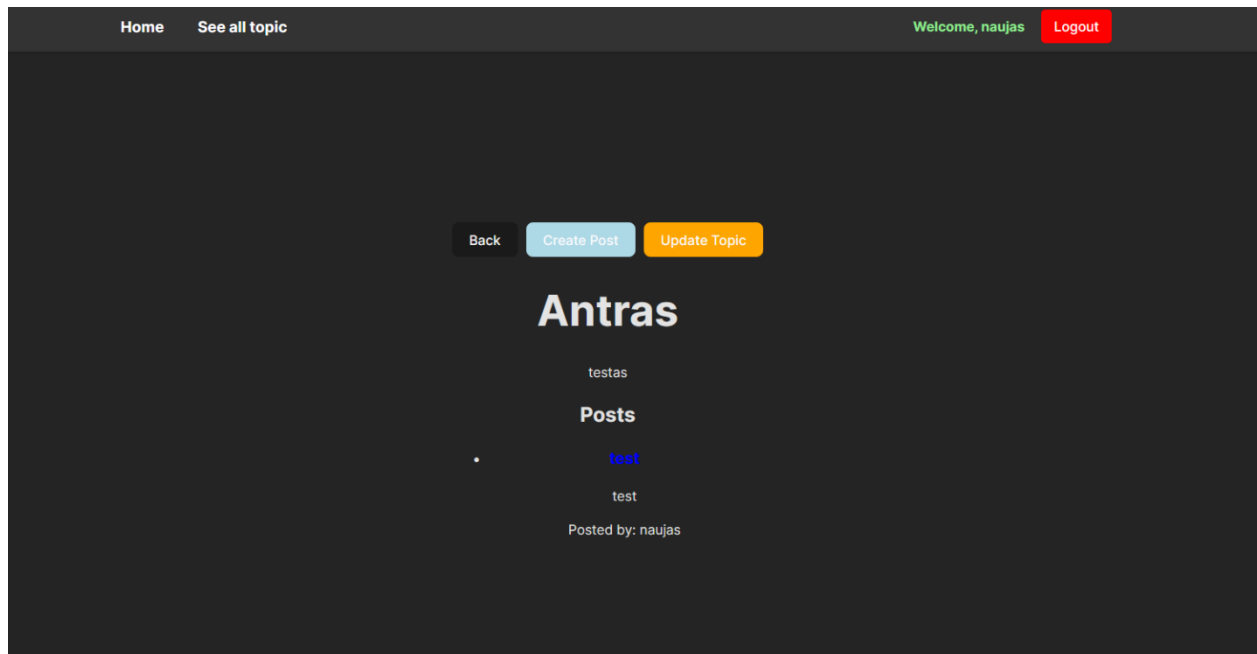
Description

Create Topic

Cancel

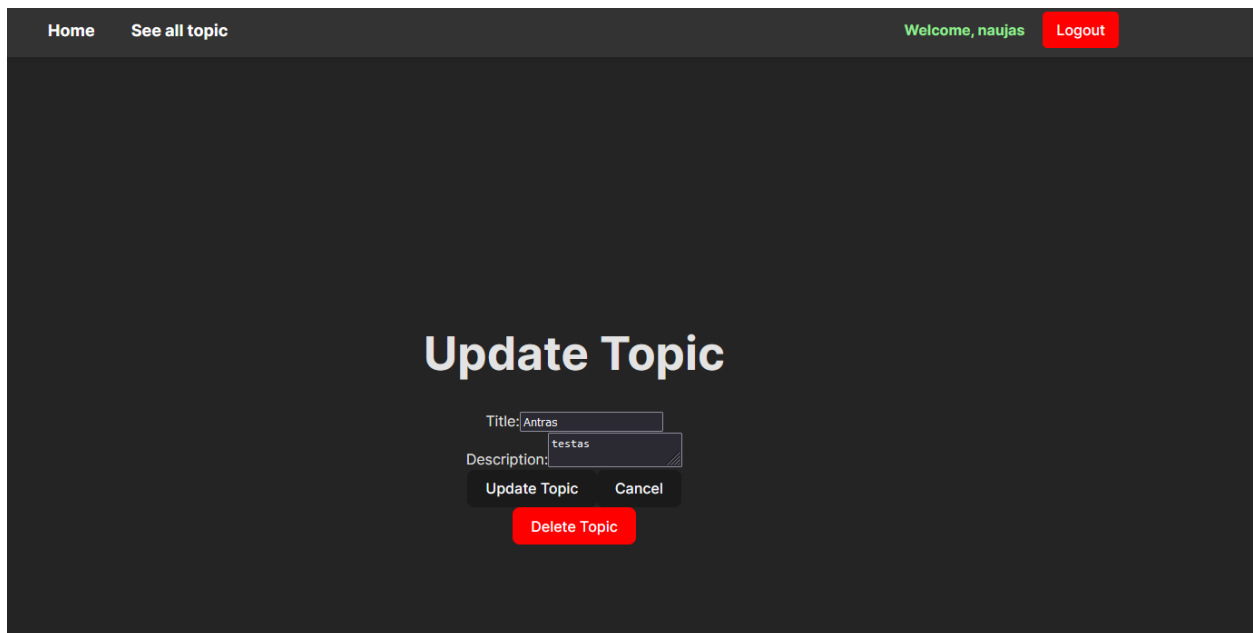
pav. 7 Temos sukūrimo puslapis

5.6 Temos detalių puslapis



pav. 8 Temos detalių puslapis

Veikia panašiai, kaip ir temų puslapis. Tik šiame puslapyje matomi visi komentarai parašyti pasirinktame pranešime, paspaudus ant to komentaro pavadinimo, kuris yra mėlynas bus atverčiamas tas komentaras. Jeigu paspausite mygtuką „Create Post“, galima bus sukurti dar viena pranešimą pasirinktai temai. Paspaudus mygtuką „Update topic“, jeigu esate administratorius arba šios temos autorius galite ją redaguoti.



pav. 9 Temos redagavimo puslapis

[Home](#) [See all topic](#) Welcome, naujas [Logout](#)

Create a Post

Title:

Content:

Create Post

Cancel

pav. 11 Pranešimo sukūrimo puslapis

5.7 Pranešimo detalių puslapis

[Home](#) [See all topic](#) Welcome, naujas [Logout](#)

[Back](#)

test

test

Created by: naujas

Add Comment

Update Post

Comments

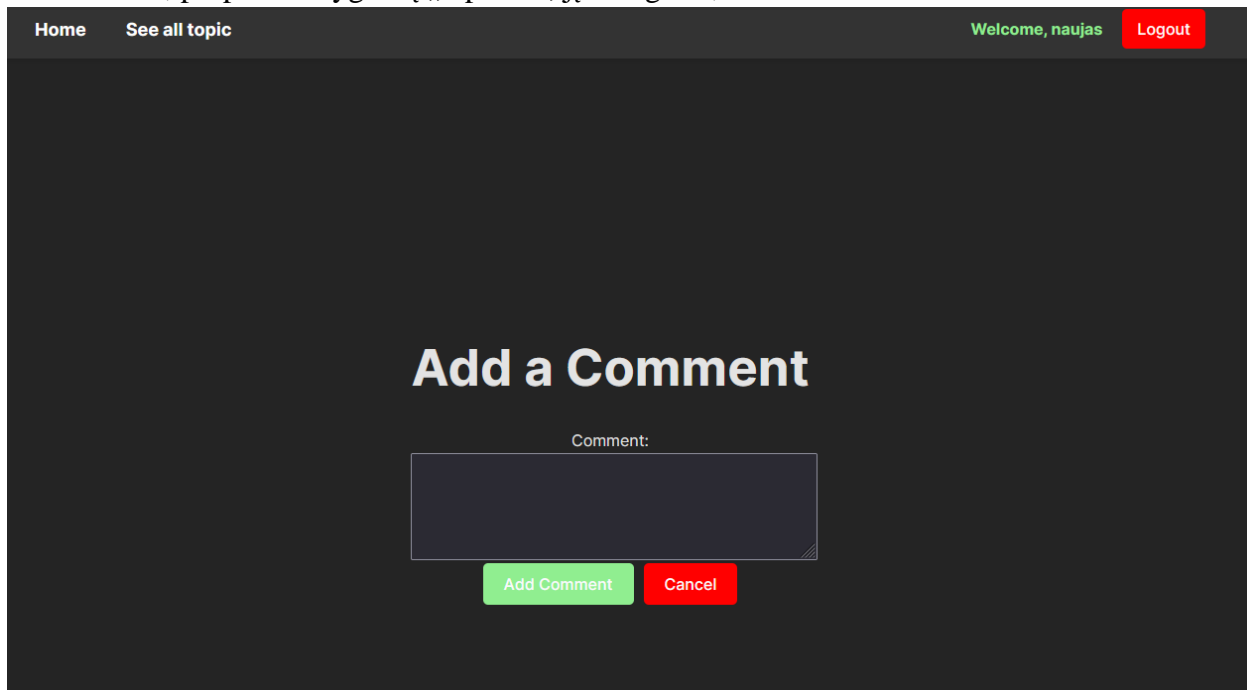
Pirmas

Posted by: naujas at 12/19/2024, 10:12:40 PM

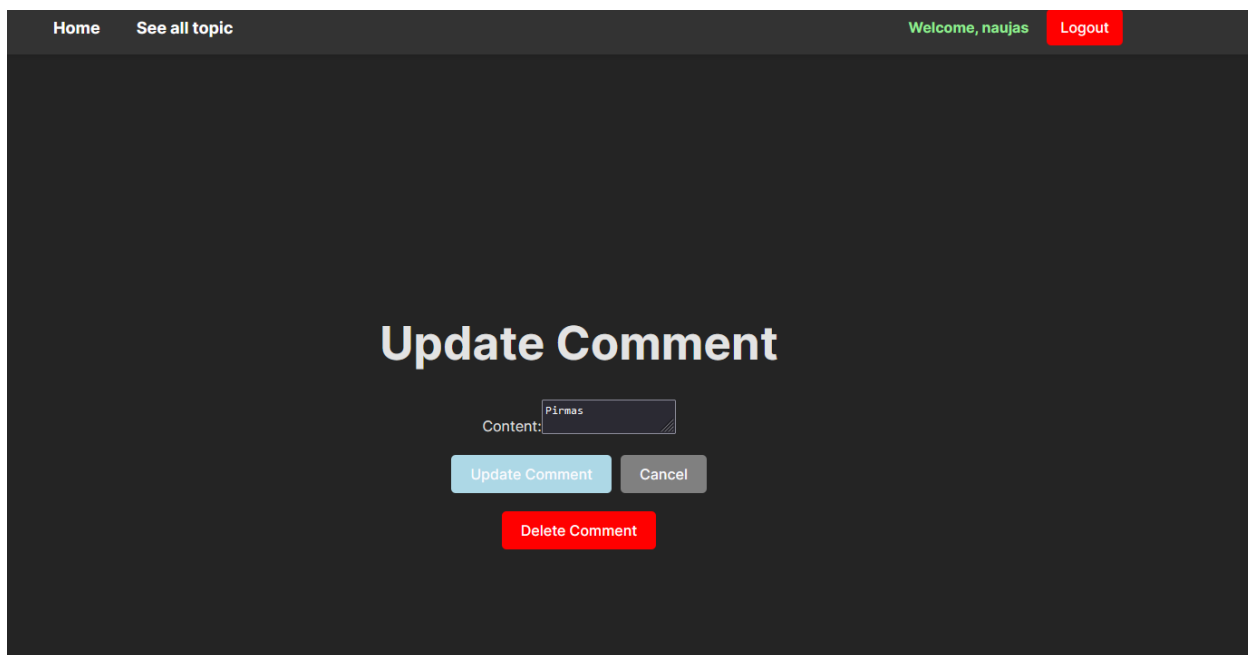
Update

pav. 10 Pranešimo detalių puslapis

Atsidarius pranešimų detalių puslapį galima matyti visus komentarus esančius pasirinktame pranešime. Galima redaguoti pranešimą, jeigu esate autorius arba administratorius, taip pat leidžia pridėti komentarą pranešime ir jeigu esate administratorius arba autorius komentaro, paspaude mygtuką „Update“, jį redaguoti,



pav. 12 Komentaro pridėjimo puslapis



pav. 13 Komentaro redagavimo puslapis

Update Post

Title:

Content:

[Update Post](#)

[Cancel](#)

[Delete Post](#)

pav. 14 Pranešimo redagavimo puslapis

6 API specifikacija

6.1 Topics API

GET

/api/topics

Retrieve all topics

^

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	<div>A list of topics</div> <div>Media type</div> <div>application/json</div> <div>Controls Accept header</div> <div>Example Value Schema</div> <div><pre>[{ "id": 0, "title": "string", "createdAt": "2024-12-19T22:41:44.621Z" }]</pre></div>	No links
500	Internal server error	No links

pav. 15 GET /api/topics

POST

/api/topics Create a new topic

^

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "title": "string"
}
```

Responses

Code	Description	Links
201	Topic created successfully	No links
422	Unprocessable Entity - Invalid input	No links
500	Internal server error	No links

pav. 17 POST /api/topics

GET

/api/topics/{topicId} Get a topic by ID

^

Parameters

Try it out

Name	Description
topicId <small>* required</small> integer (path)	<input type="text" value="topicId"/>

Responses

Code	Description	Links
200	Topic details	No links
404	Topic not found	No links
500	Internal server error	No links

pav. 16 GET /api/topics/{topicId}

PUT

/api/topics/{topicId}

Update a topic by ID

^

Parameters

Try it out

Name	Description
topicId * required	<input type="text" value="topicId"/>
integer	
(path)	

Request body * required

application/json

^

Example Value

Schema

```
{
  "title": "string"
}
```

Responses

Code	Description	Links
204	Topic updated successfully	No links
403	Forbidden	No links
404	Topic not found	No links
500	Internal server error	No links

pav. 19 PUT /api/topics/{topicId}

DELETE

/api/topics/{topicId}

Delete a topic by ID

| ^

Parameters

Try it out

Name	Description
topicId * required	<input type="text" value="topicId"/>
integer	
(path)	

Responses

Code	Description	Links
204	Topic deleted successfully	No links
403	Forbidden	No links
404	Topic not found	No links
500	Internal server error	No links

pav. 18 DELETE /api/topics/{topicId}

6.2 Posts API

GET `/api/topics/{topicId}/posts` Get all posts for a topic

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	List of posts for the topic	No links
404	No posts found for the topic	No links
500	Internal server error	No links

pav. 21 GET /api/topics/{topicId}/posts

POST `/api/topics/{topicId}/posts` Create a post for a topic

Parameters Try it out

No parameters

Request body required application/json

Example Value | **Schema**

```
"string"
```

Responses

Code	Description	Links
201	Post created successfully	No links
400	Bad Request	No links
500	Internal server error	No links

pav. 20 POST /api/topics/{topicId}/posts

GET
/api/topics/{topicId}/posts/{postId}
Get a post by ID

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Post details	No links
404	Post not found	No links
500	Internal server error	No links

pav. 23 GET /api/topics/{topicId}/posts/{postId}

PUT
/api/topics/{topicId}/posts/{postId}
Update a post by ID

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

"string"

Responses

Code	Description	Links
204	Post updated successfully	No links
403	Forbidden	No links
404	Post not found	No links
500	Internal server error	No links

pav. 22 PUT /api/topics/{topicId}/posts/{postsId}

DELETE

`/api/topics/{topicId}/posts/{postId}` Delete a post by ID

Parameters

Try it out

No parameters

Responses

Code	Description	Links
204	Post deleted successfully	No links
403	Forbidden	No links
404	Post not found	No links
500	Internal server error	No links

pav. 24 DELETE /api/topics/{topicId}/posts/{postId}

6.3 Comments API

GET `/api/topics/{topicId}/posts/{postId}/comments` Get all comments for a post

Parameters Try it out

No parameters

Responses

Code	Description	Links
200	List of comments for the post	No links
500	Internal server error	No links

pav. 26 GET `/api/topics/{topicId}/posts/{postId}/comments`

POST `/api/topics/{topicId}/posts/{postId}/comments` Create a comment for a post

Parameters Try it out

No parameters

Request body required application/json

Example Value | **Schema**

```
"string"
```

Responses

Code	Description	Links
201	Comment created successfully	No links
400	Bad Request	No links
500	Internal server error	No links

pav. 25 POST `/api/topics/{topicId}/posts/{postId}/comments`

DELETE

/api/topics/{topicId}/posts/{postId}/comments/{commentId} Delete a comment by ID

^

Parameters

Try it out

Name	Description
topicId * required	
integer	
(path)	
	topicId
postId * required	
integer	
(path)	
	postId
commentId * required	
integer	
(path)	
	commentId

Responses

Code	Description	Links
204	Comment deleted successfully	No links
403	Forbidden	No links
404	Comment not found	No links
500	Internal server error	No links

pav. 27 DELETE T/api/topics/{topicId}/posts/{postId}/comments/{commentId}

PUT

/api/topics/{topicId}/posts/{postId}/comments/{commentId} Update a comment by ID

^

Parameters

Try it out

Name	Description
topicId required integer (path)	<input type="text" value="topicId"/>
postId required integer (path)	<input type="text" value="postId"/>
commentId required integer (path)	<input type="text" value="commentId"/>

Request body required

application/json

Example Value | Schema

```
{  "content": "string"} 
```

Responses

Code	Description	Links
204	Comment updated successfully	No links
403	Forbidden	No links
404	Comment not found	No links
500	Internal server error	No links

pav. 28 PUT /api/topics/{topicId}/posts/{postId}/comments/{commentId}

GET

/api/topics/{topicId}/posts/{postId}/comments/{commentId}

Get a comment by ID

^

Parameters

Try it out

Name	Description
topicId * required integer (path)	<input type="text" value="topicId"/>
postId * required integer (path)	<input type="text" value="postId"/>
commentId * required integer (path)	<input type="text" value="commentId"/>

Responses

Code	Description	Links
200	<div>Comment details</div> <div>Media type</div> <div><div>application/json</div></div> <div>Controls Accept Header</div> <div>Example Value</div> <div>Schema</div> <div><pre>{ "id": 0, "content": "string"}</pre></div>	No links
403	Forbidden	No links
404	Comment not found	No links
500	Internal server error	No links

pav. 29 GET /api/topics/{topicId}/posts/{postId}/comments/{commentId}

6.4 Authentication API

POST `/api/accounts` Register a new user ^

Parameters Try it out

No parameters

Request body required application/json ▾

Example Value | Schema

```
{
  "userName": "string",
  "email": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
201	User registered successfully	No links
422	Unprocessable Entity - Username already exists	No links

pav. 30 POST /api/accounts

POST `/api/login` Login a user ^

Parameters Try it out

No parameters

Request body required application/json ▾

Example Value | Schema

```
{
  "userName": "string",
  "password": "string"
}
```

Responses

Code	Description	Links
200	Login successful	No links
422	Unprocessable Entity - Username or password incorrect	No links

pav. 31 POST /api/login

POST

/api/accessToken

Refresh access token

^

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Access token refreshed successfully	No links
422	Unprocessable Entity - Invalid or expired refresh token	No links

pav. 32 POST /api/accessToken

POST

/api/logout

Logout a user

^

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Logout successful	No links

pav. 33 POST /api/logout

GET

/api/accounts/{userId} Get username by userId

^

Parameters

Try it out

Name	Description
userId * required string (path)	<input type="text" value="userId"/>

Responses

Code	Description	Links
200	Username retrieved successfully	No links
404	User not found	No links

pav. 34 GET /api/accounts/{userId}

7 Išvados

Projektas realizuotas dalinai sėkmingai, naudojantis Entity framework (backend) ir React (frontend) karkasus. Realizuota JWT autentifikacija ir autorizacija, kuri kontroliuoja registruojančius ir prisijungusių vartotojų leidžiamas galimybes. Sistema buvo sėkmingai patalpina debesyse ir visi gali pasiekti šią sistemą.