

Dokumentacja Projektu

Aplikacja webowa w języku JavaScript

Projekt Webowe Biblioteki Programistyczne semestr VI

Wojciech Kozieł



**Politechnika
Śląska**

1. Założenia projektu

Projekt zakłada napisanie aplikacji webowej z wykorzystaniem języka JavaScript. Aplikacja będzie służyła do zarządzania zadaniami dodanymi przez użytkownika podzielonymi na trzy kategorie: do zrobienia, w trakcie i gotowe. Aplikacja napisana z wykorzystaniem **Node.js** i wykorzystywać będzie bazę **MongoDB**. W aplikacji zawarty będzie system logowania użytkowników.

2. Technologie

Express

Express stanowi standardową platformę serwerową dla Node.js i został użyty do stworzenia szkieletu aplikacji internetowej.

MongoDB

Biblioteka została wykorzystana do nawiązywania połączenia z bazą MongoDB. Baza wykorzystywana jest do zarządzania systemem użytkowników oraz zapisywaniem kart zadań. Każdy nowy użytkownik może dodawać swoje zadania, które są widoczne po ponownym zalogowaniu.

Mongoose

Biblioteka została wykorzystana aby poprawić łączność z bazą MongoDB oraz zapewnić walidację danych przesyłanych do bazy. Z faktu iż MongoDB korzysta z kolekcji, ważne jest zapewnienie struktury odpowiedzialnej za tworzenie rekordów w bazie danych do czego zostały wykonane Schema zawarte w bibliotece mongoose.

EJS

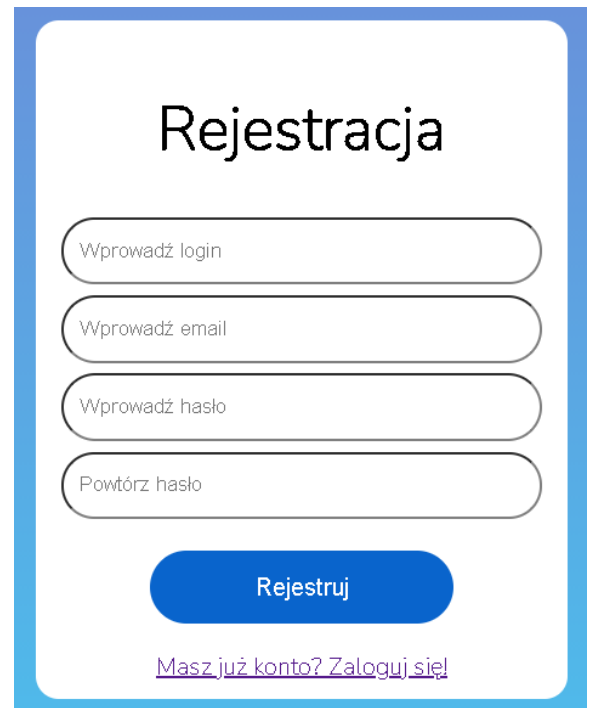
Został wykorzystany do tworzenia szablonów strony. Pozwala on na zawarcie skryptów JavaScript w kodzie HTML co pozwala na szybkie generowanie szablonowych części strony. W projekcie generowane są karty z zadaniami do których EJS sprawdził się idealnie pozwalając na warunkowe pojawianie się poszczególnych elementów w poszczególnych kartach.

3. Opis projektu

Stroną startową jest strona logowania użytkowników, która pozwala również na przejście do formularza rejestracji:



The login form is titled "Logowanie" in a large, bold, black font. Below the title are two input fields: "Wprowadź login" and "Wprowadź hasło", both with rounded rectangular borders. Below these fields is a blue button with the text "Zaloguj" in white. At the bottom, there is a link that says "Nie masz konta? Kliknij tutaj!" in purple text.



The registration form is titled "Rejestracja" in a large, bold, black font. Below the title are four input fields: "Wprowadź login", "Wprowadź email", "Wprowadź hasło", and "Powtórz hasło", all with rounded rectangular borders. Below these fields is a blue button with the text "Rejestruj" in white. At the bottom, there is a link that says "Masz już konto? Zaloguj się!" in purple text.

Logowanie odbywa się za pomocą nazwy użytkownika oraz hasła. W czasie rejestracji podajemy nazwę użytkownika, która musi być unikatowa, adres e-mail oraz hasło.

```
app.get('/login', (req, res) => {
  res.render('login', {message: ""})
})

app.post('/login', async (req, res) => {
  try {
    const login = await queries.findUser({
      name: req.body.name,
      password: req.body.password
    })

    if (login){
      user = req.body.name
      res.redirect('/cards')
    }
    else
      res.render('login', {message: "Logowanie nie powiodło się"})

  } catch (error) {
    console.error(error)
    res.render('login', {message: "Logowanie nie powiodło się"})
  }
})
```

Ekran główny aplikacji składa się z menu bocznego oraz trzech kolumn grupujących zadania.

W menu wyróżniamy sekcje z nazwą użytkownika, sekcję ze statystykami zadań - ile zadań znajduje się w każdej z kolumn oraz przyciski przekierowujące do formularzy dodawania zadania, zmiany hasła oraz ponownego logowania. Karty zgrupowane są w trzy kolumny - do zrobienia, w trakcie oraz zrobione. Każda z kart po najechaniu kursorem ujawnia menu, pozwalające na przeniesienie pomiędzy poszczególnymi kolumnami oraz usunięcie.



Dodawanie zadań odbywa się poprzez przeznaczony do tego formularz. Podajemy treść zadania oraz przewidywany termin realizacji:

Dodaj zadanie

Wprowadź treść zadania

dd.mm.rrrr

Dodaj zadanie

[Powrót do strony głównej](#)

Kod:

```
app.get('/addCard', async (req, res) =>{
  res.render('addCard', {message: undefined})
})


app.post('/addCard', async (req, res) =>{
  if (user === '') res.redirect('/login')
  try {
    const card = await queries.createCard({
      name: user,
      text: req.body.text,
      date: req.body.date
    })
    if (card){
      res.render('addCard', {message: 'Zdanie zostało dodane'})
    } else {
      res.render('addCard', {message: 'Nie udało się utworzyć zadania'})
    }
  } catch (e) {
    console.error(e)
  }
})
```

```

    res.render('addCard', {message: 'Nie udało się utworzyć
zadania'})
  }
})

```

W ramach aplikacji przewidziana jest również możliwość zmiany hasła użytkownika:



Łączenie aplikacji z bazą:

```

const mongoose = require('mongoose')

const url =
'mongodb+srv://admin:q12w34ee@cluster0.ktllp.mongodb.net/ToDoApp?retryW
rites=true&w=majority'

mongoose.connect(url, {
  useNewUrlParser: true,
  useCreateIndex: true,
  useUnifiedTopology: true
})

```

Schema użytkownika:

```

const userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    unique: true
  },
  email: {
    type: String,

```

```
        required: true,
      },
      password: {
        type: String,
        required: true
      }
    })
  const User = mongoose.model('User', userSchema)
```

Schemat zadania:

```
const cardSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  column: {
    type: Number,
    default: 1,
  },
  text: {
    type: String,
    required: true
  },
  date: {
    type: Date,
    required: true
  }
})
const Card = mongoose.model('Card', cardSchema)
```