

# Projekt Goodreads

Projekt stara si! imitowa" podzbi-r funkcjonalno#ci serwisu Goodreads.com. Projekt jest napisany w j!zyku Javascript. Celem projektu jest nauka programowania w j!zyku Javascript oraz nauka r-\$nych technologii webowych.

## Funkcjonalno!ci

Projekt posiada nast!puj%ce funkcjonalno#ci:

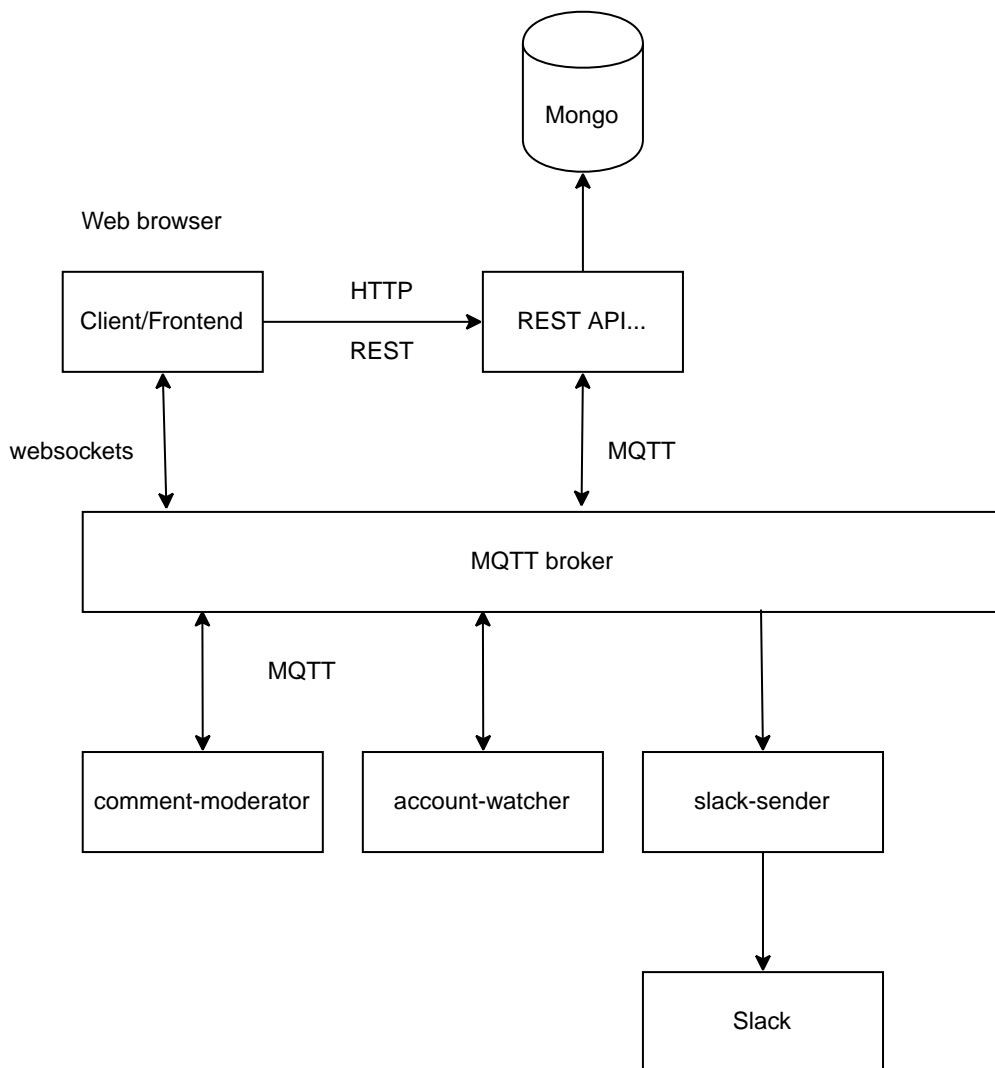
- ¥ rejestracja i logowanie
- ¥ role u\$ytkownik-w (admin, moderator, user)
- ¥ dodawanie/usuwanie/edycja/pobieranie ksi%\$ek
- ¥ dodawanie/usuwanie/edycja/pobieranie w\$asnych p-ek/list w bibliotece<sup>[1]</sup>
- ¥ dodawanie/usuwanie ksi%\$ek do w\$asnej kopii predefiniowanej listy: Want to read, Currently reading, Read
- ¥ dodawanie/usuwanie ksi%\$ek do/z dowolnej indywidualnie utworzonej p-ki/listy
- ¥ nadawanie/usuwanie/zmiana oceny ksi%\$ce
- ¥ dodawanie/usuwanie/edycja komentarzy do ksi%\$ki
- ¥ wyszukiwanie po fragmencie tekstu ksi%\$ek, komentarzy, u\$ytkownik-w
- ¥ statystyki per u\$ytkownik i globalne (najlepiej/najgorzej oceniane ksi%\$ki, najcz!#ciej dodawane ksi%\$ki, liczba ksi%\$ek przeczytanych, obecnie czytanych, chc! przeczyta",)

Dodatkowo:

- ¥ obserwacja zak\$adanych kont przez administratora
- ¥ integracja ze Slackiem (informacje o nowych u\$ytkownikach)
- ¥ komentarz do ksi%\$ki mo\$e zosta" zablokowany przez comment-moderatora (wystarczy, \$e w komentarzu znajdzie si! s\$owo fuck lub shi t, lub free)

# Architektura rozwi" zania

Goodreads like project



Aplikacja sk&ada si! z nast!puj&cych komponent&w:

- ¥ frontend - aplikacja webowa napisana w j!zyku Javascript przy wykorzystaniu React, b!d&ca interfejsem u&ytownika i administratora (dost!p do panelu administratora tylko dla administratora)
- ¥ backend - aplikacja webowa napisana w j!zyku Javascript - Node.js, b!d&ca serwerem REST API. Obecnie w API zdefiniowane s& nast!puj&ce endpointy:
  - ' /auth - autoryzacja u&ytownik&w, zak&adanie kont, weryfikacja zalogowania (3 metody)
  - ' /users - operacje na u&ytownikach (pe&n CRUD)<sup>[2][3]</sup>
  - ' /books - operacje na ksi&skach (pe&n CRUD)<sup>[2]</sup>
  - ' /comments - operacje na komentarzach (pe&n CRUD)<sup>[2]</sup>. Komentarze zablokowane przez comment-moderatora/moderatora nie s& widoczne dla u&ytownik&w chyba &e maj& rol! admin lub moderator
  - ' /shelves - operacje na p&ekach/listach (pe&n CRUD)<sup>[2]</sup>
  - ' /book-details - operacje na ocenach, dodawanie/usuwanie z p&ek/list (pe&n CRUD)

- ' `/stats` - statystyki - globalne i per użytkownik (GET - 4 różne zestawienia)
- ' `/dbs` - operacje na bazie danych (DELETE - usuwanie kolekcji MongoDB - dostępne tylko dla administratora)
- ' Dodatkowo serwer backend łączy się z brokerem MQTT w celu wysyłania wiadomości do apletów/microserwisów, oraz potrafi przyjąć zlecenia z systemu kolejkowego w celu zablokowania komentarza ocenionego przez `comment-moderator`.

¥ baza danych - MongoDB

¥ system kolejkowy - MQTT

¥ aplety/microserwisy - działające na zasadzie pub/sub, które reagują na zdarzenia z systemu kolejkowego MQTT. W projekcie zaimplementowane są następujące aplety:

- ' `slack-sender` - wysyłanie wiadomości do Slacka
- ' `account-watcher` - monitorowanie zakładanych kont użytkowników, oraz wysyłanie przez MQTT wiadomości do apletu `slack-sender` o utworzeniu nowego konta
- ' `comment-moderator` - automatyczne blokowanie komentarzy zawierających słowa `fuck`, `shit`, `free` (przykład rozwiązania - można się integrować z zewnętrznymi usługami, np. z usługą moderacji komentarzy). Po zablokowaniu komentarza wysłana jest wiadomość do admina na Slacka (następna integracja pomiędzy microserwisami przez MQTT). Zależnie od klasyfikacji treści zwracany jest inny powód blokady komentarza (np. `forbidden words` lub `spam`).

Obecnie wykorzystywane są następujące tematy (`topic`):

- ¥ `goodreads/user/created` - wysłane z backendu do apletu `account-watcher` - informacja o utworzeniu nowego konta użytkownika
- ¥ `goodreads/user/login` - logowanie użytkownika<sup>[4]</sup>
- ¥ `goodreads/book-details/created` - dodanie książki do powieści<sup>[4]</sup>
- ¥ `goodreads/book-details/updated` - zmiana informacji o powieściach, ratingu dla książki<sup>[4]</sup>
- ¥ `goodreads/slack/send` - wysłane z apletu `account-watcher` lub `comment-moderator` do apletu `slack-sender` - informacja o utworzeniu nowego konta użytkownika
- ¥ `goodreads/comments/created` - wysłane z backendu do apletu `comment-moderator` - informacja o utworzeniu nowego komentarza
- ¥ `goodreads/comments/updated` - wysłane z backendu do apletu `comment-moderator` - informacja o zmianie komentarza
- ¥ `goodreads/comments/blocked` - wysłane z apletu `comment-moderator` do backendu - informacja o zablokowaniu komentarza. Backend zapisuje powód blokady w bazie danych.

## Technologie użyte w projekcie Goodreads.

¥ backend - API REST serwer

- ' operacje CRUD na bazie danych (8 routów)
- ' autoryzacja użytkowników przy pomocy JWT - sesja w headerze HTTP w postaci

podpisanego tokena JWT

- ' wykorzystanie systemu kolejkowego MQTT - wysyłanie zdarzeń z serwera do kolejki, reakcja mikroserwisów na zdarzenia z kolejki, komunikacja między mikroserwisami

¥ frontend - aplikacja webowa napisana w języku Javascript - React

- ' websockets - do komunikacji z brokerem MQTT
- ' wykorzystanie biblioteki React Router do obsługi routingu
- ' wykorzystanie biblioteki Redux do zarządzania stanem aplikacji

¥ baza danych - MongoDB

- ' wykorzystanie biblioteki Mongoose do komunikacji z bazą danych
- ' operacje na bazie - find, insert, update, delete, aggregate
- ' wyszukiwanie w bazie danych jest wykonane na dwa różne sposoby:
  - ) wyszukiwanie za pomocą wyrażenia regularnych w kilku wybranych polach np. `book.title`, `books.author` - gdzie pola te są połączone za pomocą operatora `$or`. Dzieje się tak w przypadku gdy chcemy przeszukać konkretne pola w bazie danych. Tworzony RegExp jest odpowiednikiem `/.search_text./i` - zwraca wszystkie dokumenty w których występuje szukany tekst w dowolnym miejscu tekstu (case insensitive)
  - ) wyszukiwanie w całym dokumencie za pomocą wyrażenia szukania full-text search (komentarze) - nie ma wtedy opcji wyszukiwania tylko w wybranych polach
- ' operacja na samej bazie danych - usuwanie kolekcji (dostępne tylko dla administratora)

¥ system kolejkowy - MQTT

- ' połączenie z frontendu do MQTT przez websockets (broker Mosquitto wystawia swoje API przez websockets). Websockets opakowują protokół MQTT aby umożliwić standardową komunikację z frontendem
- ' połączenie z backendu do MQTT przez natywny protokół MQTT

## Bezpieczeństwo

### Uwierzytelnianie

Uwierzytelnianie jest wykonywane przez endpoint API `/auth/login`. Wymagane jest podanie loginu i hasła użytkownika. W przypadku poprawnego uwierzytelnienia serwer zwraca token JWT. Token ten jest zapisywany w localStorage przeglądarki. W przypadku każdego zapytania do API serwer sprawdza czy w nagłówku HTTP jest token JWT. Jeśli tak to sprawdza czy token jest poprawny. Jeśli tak to zwraca dane zapytania. Jeśli nie to zwraca błąd 401 Unauthorized. W tokenie JWT zapisywane są dane użytkownika (email, role, username).

Przy zakładaniu użytkownika (endpoint `POST /auth/signup`) serwer sprawdza czy podany email nie jest już zajęty. Jeśli jest to zwraca błąd 409 Conflict. Jeśli nie jest to tworzy nowego użytkownika w bazie danych.

Hasło użytkownika w bazie danych jest hashowane przy pomocy biblioteki `bcrypt` wraz z dodanym

saltem. Hasło jest hashowane przy rejestracji użytkownika oraz przy logowaniu. W przypadku logowania serwer porównuje hashowane hasło z logowania z hashowanym hasłem zapisanym w bazie danych. Hasło jest niemożliwe do "odzyskania" z hashowanej wersji ponieważ funkcja hash jest jednostronna.

Hashowane hasło z bazy danych nie jest zwracane w żadnym zapytaniu do API.

## Autoryzacja

Autoryzacja jest wykonywana przez middleware. Zależy od endpointu oraz zapytania sprawdzane są różne uprawnienia.

Na przykład:

- dla endpointu `GET /users` umożliwiające otrzymanie listy wszystkich użytkowników - uprawnienia pozwalają na pobranie tej listy przez użytkownika z rolą `admin`, ale zwracają błąd przy roli `user`. Przy próbie pobrania danych jednego użytkownika (endpoint `GET /users/:id`) - rola `user` może pobrać tylko swoje dane, a rola `admin` może pobrać dane dowolnego użytkownika. Jeśli uprawnienia nie są spełnione zwracany jest błąd 403 Forbidden.
- dla endpointu `PATCH /users/:id` umożliwiające zmianę danych użytkownika - rola `user` pozwala zmienić tylko część swoich danych (password, username, avatar\_url etc.). Rola `admin` pozwala także na zmianę atrybutu `role` dla każdego użytkownika. Użytkownik z oczywistych powodów nie ma uprawnień do zrobienia tego samodzielnie.

## Cookies

Przy logowaniu ustawiane jest cookie o nazwie `last_login` z czasem życia 1 rok. Cookie to jest używane do wyświetlania informacji o ostatnim logowaniu użytkownika.

Przykładowy response:

```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Set-Cookie: last_login=1675021103569; Max-Age=31536000; Path=/; Expires=Mon, 29 Jan 2024 19:38:23 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 219
ETag: W/"db-ip1L8lUuBPkV4zFfYn7y074Zd98"
Date: Sun, 29 Jan 2023 19:38:23 GMT
Connection: close
```

```
{
  "message": "OK",
  "token":
    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyaWljb2kiOiJW5pYUBsb2Nhbmhvc3QiLCJyb2xiOiJ0eXNlciJ9LCJpYXQiOiJlbnV4cCI6MTY3NTYyNTkwM30. aQJ8D9xM6ANaIX0AaqZX5RwYY4G7wEF_dISI-o0ppo0"
}
```

## Infrastruktura

Infrastruktura została zbudowana na bazie dockera i pliku docker-compose.yml. W pliku tym zdefiniowane są następujące kontenery:

- ✚ **mongo** - baza danych MongoDB
- ✚ **mongo-express** - GUI do bazy danych MongoDB
- ✚ **mosquitto** - broker MQTT

Wszystkie te usługi udostępniają swoje porty na zewnątrz, aby można było do nich połączyć z zewnątrz. Wszystkie kontenery są połączone w jedną sieć, aby móc sobie komunikować między sobą. Taka definicja pozwala na łatwy lokalny development gdyż elementy infrastruktury są dostępne na lokalnym komputerze w dockerze.

## Testy

- ✚ W trakcie developmentu używany jest plik **route.rest** zawierający wywołania każdej metody API. Jest to plik wykorzystywany przez plugin VSCode **Rest Client**. Jest on klientem API i umożliwia on testowanie API wprost ze środowiska VSCode. Jest to odpowiednik Postmana, ale wprost w środowisku VSCode.
- ✚ W aplikacji przygotowany jest system testów umożliwiający podejście BDD (Behavioural Driven Development). Testy zostały napisane w języku Gherkin, który jest językiem opisującym zachowanie aplikacji. Testy zostały napisane w plikach **\*.features/** znajdujących się w podkatalogu **features/**. W testach skorzystano z biblioteki **cucumber-js** umożliwiającej pisanie testów w języku naturalnym oraz **pactum** - ułatwiającej test REST API. Przykładowy scenariusz testowy:

```

Ê Scenario: Get comments
Ê     Given I make a "GET" request to "/comments"
Ê     When I receive a response
Ê     Then I expect response should have a status 200
Ê     And I expect response should have a json like
Ê         ""
Ê         [
Ê         {
Ê         "book_id": ${BookId},
Ê         }
Ê         ]
Ê         ""

```

W tym scenariuszu testowym sprawdzamy czy po wykonaniu %dania GET na adres /comments otrzymamy odpowiedz\* z kodem 200 oraz czy w odpowiedzi znajduje si! tablica z obiektami, kt-rych pola book\_id maj% warto#" zapisan% w zmiennej BookId z poprzedniego %dania. Uruchomienie test-w odbywa si! za pomoc% komendy npm run test w katalogu ./src.

Inny przyk&ad testu:

```

Ê Scenario Outline: Check valid authorization
Ê     Given I make a "<method>" request to "<endpoint>"
Ê     When I receive a response
Ê     Then I expect response should have a status <status>

Ê     Examples:
Ê         | method | endpoint      | status |
Ê         | GET    | /users        | 403    |
Ê         | GET    | /comments     | 200    |
Ê         | GET    | /books        | 200    |
Ê         | GET    | /book-details | 200    |
Ê         | GET    | /shelves      | 200    |
Ê         | GET    | /stats        | 200    |

```

[1] administrator mo&e dodawa"/usuwa"/edytowa" standardowe p-&ki dost! pne dla ka&dego u&ytownika: np. Want to read, Currently reading, Read

[2] pe&en CRUD oznacza: get all, get by id, create (post), update (patch), delete )

[3] r-&ne uprawnienia dla r-&nych r-l u&ytownik-w

[4] obecnie nic nie nas&uchuje na tym topicu