PAINT Projekt - dokumentacja

Jakub Ciarciński, Eryk Jakubowski

1. Temat Projektu:

Forum dyskusyjne do gry League of Legends.

2. Zespół:

- Kierownik projektu - Jakub Ciarciński,
- Programista frontend - Jakub Ciarciński,
- Programista backend - Jakub Ciarciński, Eryk Jakubowski,
- Programista bazy danych - Eryk Jakubowski.

3. Opis szczegółowych założeń funkcjonalnych:

- Logowanie:
    - Użytkownik może zalogować się do systemu, podając swoje dane uwierzytelniające (email i hasło).
    - System weryfikuje dane użytkownika w bazie danych i, w przypadku poprawnej autoryzacji, tworzy sesję.
    - W przypadku błędnego logowania użytkownik otrzymuje stosowny komunikat.
    - Obsługiwane jest również przechowywanie stanu logowania za pomocą tokenów (np. JWT).
- Przeglądanie wątków:
    - Użytkownik ma dostęp do listy wszystkich wątków w systemie, z możliwością filtrowania według kategorii.
    - Wątki są posortowane według daty utworzenia.
    - Szczegóły każdego wątku można przeglądać na osobnej stronie, gdzie widoczne są komentarze i oceny.
- Tworzenie nowych wątków do istniejących kategorii:
    - Zalogowani użytkownicy mogą tworzyć nowe wątki, wybierając kategorię, w której wątek ma się znajdować.
    - Każdy wątek wymaga podania tytułu oraz opcjonalnie treści początkowej.
    - Po utworzeniu wątek automatycznie staje się widoczny dla innych użytkowników w odpowiedniej kategorii.
- Dodawanie komentarzy pod wątkiem:
    - Użytkownicy mogą dodawać komentarze do istniejących wątków.
    - System obsługuje dynamiczne odświeżanie liczby komentarzy w wątku.
- Odpowiedzi do komentarzy:
    - Możliwość tworzenia odpowiedzi na istniejące komentarze, tworząc drzewiastą strukturę komentarzy.
    - Każda odpowiedź ma przypisanego autora, czas utworzenia oraz odwołanie do głównego komentarza.
- Możliwość obserwowania wątków:

    - Użytkownik może "obserwować" wybrane wątki.
    - Funkcja obejmuje także przeglądanie listy obserwowanych wątków w osobnym panelu.
- Możliwość zamknięcia wątku przez autora:
    - Autor wątku ma prawo zamknąć wątek, uniemożliwiając dodawanie kolejnych komentarzy.
- Możliwość oceny wątku/komentarza w skali +/-:
    - Każdy użytkownik może głosować na wątki i komentarze w skali "lubię/nie lubię" (+/-).
    - Liczba głosów jest widoczna w czasie rzeczywistym.
    - Użytkownik może zmienić swoją ocenę, ale nie może oddać więcej niż jednego głosu.

4. Opis szczegółowych założeń architektonicznych:

System jest oparty na trzech głównych komponentach:

- Backend (Flask + PrismaSQL):
    - Flask odpowiada za obsługę żądań API.
    - PrismaSQL zapewnia komunikację z bazą danych, upraszczając zarządzanie schematami i migracjami.
    - REST API z punktami końcowymi do obsługi wątków, użytkowników, komentarzy i ocen.
- Frontend (React):
    - React służy jako warstwa wizualna systemu, odpowiedzialna za interakcję z użytkownikiem.
    - Dynamiczne renderowanie wątków, komentarzy i ocen.
    - Komunikacja z backendem odbywa się za pomocą REST API.
- Baza danych (MySQL + Prisma):
    - MySQL przechowuje dane aplikacji, w tym użytkowników, wątki, komentarze, głosy i powiadomienia.
    - Prisma jest używana jako ORM (Object-Relational Mapping), co upraszcza zarządzanie strukturą bazy oraz wykonywanie złożonych zapytań.

5. Opis wykorzystania narzędzi programowych w powiązaniu z ww. założeniami funkcjonalnymi i architektonicznymi:

- Flask:
    - walidacja danych oraz obsługa punktów końcowych API.
- React:
    - Tworzenie dynamicznego, responsywnego interfejsu użytkownika.
    - Obsługa widoków wątków, komentarzy oraz funkcji interaktywnych (np. głosowanie).
- Prisma:
    - ORM dla MySQL, umożliwiający prostą konfigurację modeli i migracji danych.
    - Wykorzystywany do złożonych zapytań dotyczących wątków, komentarzy i ocen.
- MySQL:

    - Przechowywanie danych aplikacji.
- JWT:
    - Realizacja uwierzytelniania i autoryzacji w systemie.

6. Opis realizacji:

- Proces realizacji:
    - Postawienie szkieletu frontendu i backendu,
    - Postawienie bazy danych,
    - Modelowanie bazy danych przez PrismaSQL oraz SQLAlchemy,
    - Połączenie bazy danych z backendem,
    - Zastosowanie ngrok aby udostępnić backend członkom zespołu,
    - Rozwijanie frontentu i dodawanie endpointów w backendzie w zależności od potrzeb na bieżąco,
    - Dodanie przykładowych danych do bazy danych,
    - Upiększanie interfejsu oraz testowanie use case'ów,
- Problemy:
    - Problemy z udostępnieniem lokalnej bazy danych członkom zespołu, rozwiązane przez zastosowanie linku ngrok udostępniającego backend połączony z bazą danych,
- Modyfikacje rozwiązań:
    - Skorzystanie z SQLAlchemy do tworzenia zapytań bazy danych, ograniczeń i relacji do tabel w bazie danych,
    - Zrezygnowanie z Auth JWT na rzecz przechowywania danych sesji w local storage w celu uproszczenia implementacji,
- Modyfikacje założeń:
    - Rezygnacja z zamykania wątków - uznaliśmy tę funkcjonalność za mało użyteczną,
    - Rezygnacja z systemu odpowiadania na komentarze w celu utrzymania przejrzystości aplikacji,

7. Opis techniczny produktu:

- Backend:
    - Endpointy takie jak:
        - /api/login - do logowania użytkownika
        - /api/categories - do wyświetlania kategorii
        - /api/threads - do wyświetlania wszystkich wątków
        - /api/<string:category_name>/threads - do wyświetlania wątków danej kategorii, tworzenia wątku
        - /api/<string:user_name>/followed_threads - Obserwowane wątki przez użytkownika
        - /api/threads/<int:thread_id>/comments - Komentarze pod wątkiem, tworzenie komentarza pod wątkiem
        - /api/users - Do rejestrowania nowego użytkownika
        - /api/comments/<int:comment_id>/vote - do głosowania na komentarz

        - /api/threads/<int:thread_id>/follow - do obserwowania wątku przez użytkownika
    - PrismaSQL i SQLAlchemypozwoliły na wygodne zamodelowanie tabel jak:
        - User - przechowuje dane użytkownika
        - Category - reprezentuje daną kategorie
        - Thread - przechowuje wątki
        - Comment - przechowuje komentarze
- Baza danych:
    - Tabele główne:
        - User
        - Category
        - Thread
        - Comment
    - Relacje:
        - Wiele do wielu jak:
            - User >-< Comment(upvoters/downvoters)
            - User >-< Thread(followers)
        - Jeden do wielu:
            - Thread -< Comment
            - Category -<Thread
            - User -< Comment(author)
            - User -< Thread(author)
- Frontend:
    - Stworzono podstrony zarządzane przez react-router-dom:
        - /home albo /
        - /thread/:threadId
        - /login
        - /register
        - /new_thread
        - /categories
        - /category/threads/:catId
        - /followed
    - Stworzono formularze z walidacją argumentów z użyciem biblioteki yup.
    - Skorzystano z frameworku Tailwind do stylowania graficznego interfejsu.
- Komunikacja:
    - Frontend z backendem poprzez zapytania typu GET i POST do endpointów.
    - Backend z bazą danych przy użyciu kwerend tworzonych poprzez SQLAlchemy

8. Opis instalacji produktu:

Do uruchomienia projektu wystarczy zainstalować i uruchomić frontend:

1. W folderze \client\client\:
    - npm install
    - npm install axios
    - npm install web-utils
    - npm install web-vitals

- npm start

instalacja backendu nie jest wymagana (działa cały czas na linku ngrok połączonym z backendem, link jest aktualny i zapisany w zmiennej “proxy” w frontendzie), ale jest możliwa instalacja i uruchomienie własnego serwera podążając kolejno krokami:

1. Instalacja mysql (należy zapamiętać hasło) i postawienie bazy danych my_flask_app:
   - sudo systemctl start mysql
   - sudo systemctl enable mysql
   - mysql -u root -p
   - CREATE DATABASE my_flask_app;
2. Zainstalowanie potrzebnych rozszerzeń w folderze server:
   - npm install prisma --save
   - pip install Flask prisma
   - pip install Flask-cors
   - npx prisma migrate dev
   - npx prisma generate
3. Uruchomienie kodów pythonowych:
   - models.py
   - seed.py
   - app.py
4. Zalogowanie się do ngrok - Online in One Line, zainstalowanie ngrok.exe w folderze server i wpisanie komendy:
   - ngrok http 5000

   Link powinien być widoczny Forwarding {https://7433-83-24-176-170.ngrok-free.app} Następnie zaktualizować “proxy”:w frontendzie w client\client\package.json i teraz backend jest połączony z bazą, a frontend z backendem.

9. Opis użytkowania produktu:

Nagłówek:

- Kliknięcie w logo Forum of Legends prowadzi do strony domowej.
- Przyciski Zaloguj się i Utwórz konto prowadzą do odpowiednich formularzy.
- Przycisk Wyloguj jest widoczny jedynie dla zalogowanego użytkownika.

Strona domowa:

- Przycisk Kategorie prowadzi do podstrony z wszystkimi kategoriami wątków.
  - Można wybrać kategorię z listy aby wyświetlić listę wszystkich wątków danej kategorii.
- Przycisk utwórz wątek:
  - Dla niezalogowanego użytkownika przekierowuje do strony z logowaniem.
  - Dla zalogowanego użytkownika przenosi do strony z formularzem zakładania nowego wątku.
    - Można wybrać kategorię z listy.
- przycisk Obserwowane:

    - Dla niezalogowanego użytkownika przekierowuje do strony z logowaniem.
    - Dla zalogowanego użytkownika przenosi do strony z listą obserwowanych przez niego wątków.

Listy wątków:

Aplikacja posiada wiele ekranów zawierających listy wątków, zasady ich działania są analogiczne:

- Kliknięcie w wybrany wątek przenosi do strony z listą komentarzy do tego wątku.

Listy komentarzy:

- Kliknięcie w gwiazdkę przez zalogowanego użytkownika powoduje dodanie wątku do ulubionych.
- Kliknięcie w strzałkę w górę lub w dół przy komentarzu przez zalogowanego użytkownika powoduje dodanie oceny pozytywnej lub negatywnej do tego komentarza.
- Na dole dyskusji:
    - Niezalogowany użytkownik widzi komunikat, że należy się zalogować aby dołączyć do dyskusji.
    - Zalogowany użytkownik ma możliwość dodania własnego komentarza do wątku.