Projekt

Aplikacja streamingowa kursów, instruktaży oraz poradników - DianomiTV

Marcel Kasprzycki, Witold Padula (P3)

Bazy Danych

Prowadzący: mgr inż. Nikodem Bulanda

June 13, 2023

Contents

1 Tytuł

Aplikacja streamingowa kursów, instruktaży oraz poradników - DianomiTV

2 Nazwa robocza

Dianomi

3 Cel

Zapewnienie dostępu użytkowników do materiałów wideo o szerokiej zakresie tematycznym z różnego rodzaju kursami, instruktażami oraz poradnikami. Aplikacja będzie oferowała rozległą bazę materiałów, która nie będzie ograniczona tematyką. Aplikacja będzie zapewniała rozwój intelektualny połączony z rozrywką. Zaoferowane będzie kilka płatnych planów subskrybcyjnych oraz jeden darmowy. Promowane będzie współdzielenie konta pomiędzy kilku użytkowników, poprzez system współdzielenia subskrybcji. ~~Aplikacja będzie zbierała informacje telemetryczne na temat aktywności użytkowników na stronie, oferując rekomendacje dotyczące materiałów wideo oraz informacje o możliwych, dalszych kierunkach rozwoju aplikacji.~~ Będzie zarządzana przed uprawnione podmioty, które posiadają odpowiedni dostęp do metod administracyjnych. ~~Dodatkowym elementem aplikacji będzie usługa pozwalająca na generowanie sztucznego, niemal identycznego do ludzkiego ruchu na stronie, pozwalającego na szybkie i efektywne zdobycie statystyk użytkowania strony.~~

4 Zakres

4.1 Analiza wymagań

Aplikacja powinna oferować usługę udostępniającą materiały wideo z szerokiej tematyce z różnego rodzaju kursami, instruktażami i poradnikami, które dostępne są po zalogowaniu do serwisu. Aplikacja powinna być zabezpieczona za pomocą najnowszych standardów, jednocześnie oferując bilans pomiędzy bezpieczeństwem użytkowników, samej aplikacji, a komfortem jej używania.

Pierwszym ekranem dostępnym dla użytkownika jest ekran logowania, który może być przełączony na ekran rejestracji. Do rejestracji wymagany jest adres e-mail oraz hasło powtórzone dwukrotnie, posiadające odpowiednią złożoność, użytkownik musi wyrazić również zgodę na regulamin serwisu. Adres pocztowy musi zostać potwierdzony przez użytkownika, przed możliwością uzyskania dostępu do aplikacji. W wypadku problemów z rejestrowaniem, użytkownik zostanie poinformowany o błędzie, który wystąpił w momencie wysłania formularza rejestracyjnego. Użytkownik, za pomocą adresu e-mail oraz wcześniejszego hasła może uzyskać dostep do serwisu. Alternatywnie, jeśli użytkownik już posiada konto w serwisie, może się zalogować. Strona oferuje również możliwość zresetowania zapomnianego hasła.

Po spędzeniu odpowiedniego czasu na stronie, będąc zalogowanym lub po obejrzeniu kilku dowolnych materiałów, użytkownik dostanie nieinwazyjny komunikat zachęcający do

wykupienia subskrybcji. Jeśli użytkownik posiada już subskrybcję, a nie dołączył jeszcze żadnego użytkownika, otrzyma nieinwazyjny komunikat zachęcający do zaproszenia znajomego i przyłączenia go do subskrybcji. W momencie wybrania materiału wymagającego subskrybcję użytkownik dostanie komunikat informujący o potrzebie wykupienia subskrybcji. W wypadku wygasającej subskrybcji, użytkownik dostanie stosowny komunikat, który będzie o tym informował. Próba komunikacji ze stroną, do której użytkownik nie posiada dostępu zostanie zakończona wyświetleniem odpowiedniej strony błędu HTTP 403.

Logując się do aplikacji, użytkownik otrzymuje dostęp do estetycznej i prostej strony głównej, oferującej katalog rekomendowanych materiałów do obejrzenia. Materiały mają być polecane na podstawie wcześniejszej aktywności użytkownika, używanego przez niego planu oraz ogólnych trendów oglądalności w obrębie aplikacji. Materiały mogą być swobodnie filtrowane odpowiednimi kategoriami, datą produkcji, datą dodania, oceną, studiem, popularnością oraz wiekiem odbiorcy. Materiały mogą być wyszukiwane po nazwie.

Wybierając materiał, użytkownik zostanie przekierowany na stronę odtwarzacza, która dostępna jest tylko i wyłącznie jesli użytkownik posiada stosowną subskrybcję, oraz jest zalogowany. W centrum strony znajduje się odtwarzacz z wyborem jakości materiału, poziomem głośności oraz oferuje swobodne kontrolowanie punktu, w którym materiał jest odtwarzany. Na stronie znajduje się dodatkowo sekcja z informacjami o materiale, czyli tytułem, opsiem, kategorią, wyświetleniami, ocenami oraz komentarzami. Użytkownik może swobodnie komentować oraz pozostawiać oceny. Komentarze mogą zostać zreportowane w związku z naruszeniem regulaminu. Kolejną elementem strony jest sekcja z polecanymi użytkownikowi materiałami, które wyświetlają się na podstawie wybieranych przez niego wcześniej kategorii. Każde wideo obejrzane przez użytkownika posiada informację o czasie, który poświęcił na jego oglądanie oraz o ostatnim punkcie, w którym je zakończył.

Otwierając menu w panelu bocznym, użytkownik uzyskuje dostęp do panelu konfiguracji profilu, gdzie może wprowadzić zmiany do swojego konta, dotyczące adresu e-mail, hasła, aktywnej subskrybcji oraz dodatkowych kont dziedziczących subskrybcję. Kolejnym elementem menu w panelu bocznym jest strona z historią oglądanych materiałów.

Kolejnym istotnym elementem jest panel administratorski, który oferuje dodatkowe narzędzia pozwalające na administrację witryną. Administrator z odpowiedniego panelu posiada dostęp do listy użytkowników, informacji o ich kontach oraz ma możliwość ich filtrowania oraz wyszukiwania. Administrator może modyfikować dane użytkowników, ich aktywne subskrybcje, czy podglądać ich historię oglądania, płatności, komentarze oraz polubienia. Panel administratorski pozwala również na dodawanie oraz usuwanie użytkowników oraz na ręczne potwierdzanie ich adresów e-mail. Użytkownicy mogą być banowani, a pozostawione przez nich komentarze mogą być usuwane. Bany, które użytkownik może otrzymać zakładają określony czas lub stały ban na konkretny email. Administrator otrzymuje również dostęp do biblioteki udostępnionych materiałów i podobnie jak w przypadku użytkowników może manipulować poszczególnymi materiałami, grupami czy komentarzami skojarzonymi z tymi materiałami.

~~Istnieje równie dodatkowy panel ze statystykami, który pozwala na sprawdzanie ruchu na stronie, tendencji użytkowników, ogólnych zainteresowań, popularności poszczególnych materiałów, kategorii czy tagów. Statystyki użytkowania strony maj za zadanie proponowa~~

~~dalsze możliwe etapy rozwoju treści, czy te samej aplikacji. Panel statystyk będzie nadto wskazywa problematyczne miejsca, na których powinno si skupi uwag w celu zapewnienia stabilności aplikacji oraz jej bezpieczeństwa.~~

~~Istotnym elementem rozwojowym będzie zastosowany w aplikacji system generowania ruchu, który będzie przypomina ten wykonywany przez operacje użytkowników. System ten pozwoli na wygenerowanie syntetycznego ruchu, który pozwoli na wybranie przyszłego kierunku rozwoju aplikacji. System ten pozwoli równie na odpowiednie obciążenie serwera, które pozwoli oceni wydajno systemu. Narzędzie nie będzie bezpośrednio zaimplementowane w aplikację, ale będzie dodatkowym zewnętrznym interfejsem pozwalającym na dostosowanie parametrów ruchu, który miałby pojawi si na stronie.~~

4.2 Wymagania niefunkcjonalne

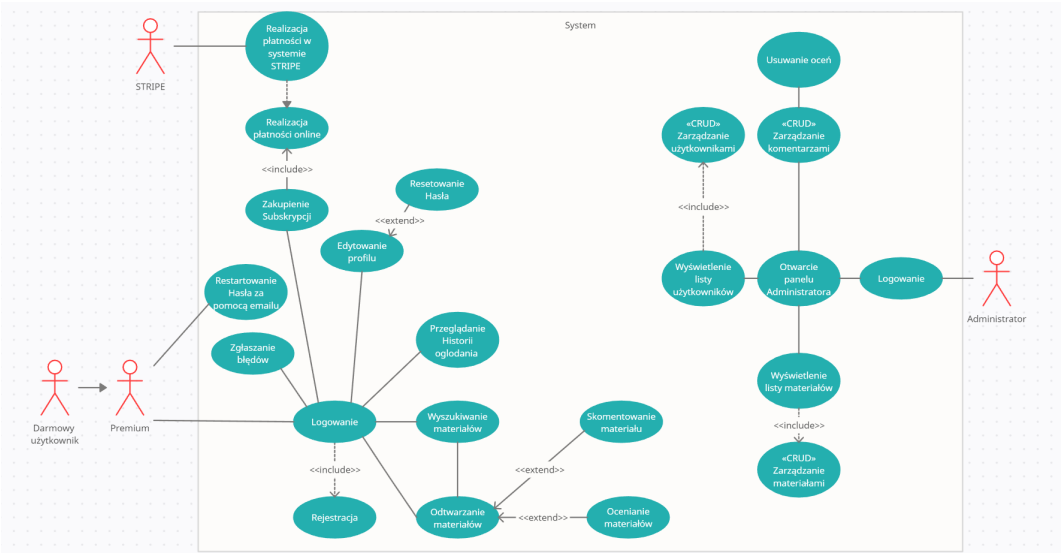
- Prosty i responsywny interfejs użytkownika.
- Zapewnienie bezpieczeństwa danych aplikacji
- Udostępnienie płynnego oglądania materiałów wideo na stronie
- Zapewnienie systemu autoryzacji użytkowników w aplikacji
- Umożliwienie odzyskania konta użytkownikom, którzy zapomnieli danych autoryzacyjnych
- Zastosowanie systemu ograniczającego dostęp do treści za pomocą bariery subskrybcyjnej
- Zachęcenie użytkowników do wykupowania subskrybcji
- Zachęcanie użytkowników do korzystania z serwisu i pochłaniania treści
- Umożliwienie użytkownikom dzielenia się swoimi reakcjami na materiały
- Dostęp administracyjny do aplikacji
- ~~Zapewnienie metryk oglądalności~~
- Udostępnienie kanału komunikacji o błędach
- ~~Zapewnienie statystyk pozwalających na określenie kierunków rozwoju bez aktywnego ruchu użytkowników~~

4.3 Wymagania funkcjonalne

- Zastosowanie modelu MVC oraz frameworka SPA dla aplikacji, w celu odciążenia serwera.
- Implementacja reverse proxy z certyfikatami SSL/TLS.
- Wykorzystanie zewnętrznego systemu przechowywania materiałów z dostępem z poziomu aplikacji
- Implementacja systemu autoryzacji użytkowników za pomocą loginu i hasła, pozwalającego na rejestrację w serwisie, z wykorzystaniem tokenów JWT.
- Dodanie systemu przypominającego hasła, który bazuje na potwierdzonych przez użytkowników adresach e-mail
- Wykonanie systemu pozwalającego na pobieranie opłat od użytkowników w zamian za czasową subskrybcję, oraz kontrolującego posiadane przez nich subskrybcje
- Wykorzystanie nieinwazyjnych powiadomień dla użytkowników informujących o możliwości wykupienia subskrybcji

- Implementacja systemu pozwalającego na współdzielenie subskrybcji pomiędzy użytkownikami za pomocą adresu e-mail
- Dodanie systemu rekomendującego użytkownikom materiały na podstawie oglądanych przez nich treści, popularnych materiałów na stronie
- Użytkownicy mogę reagować na materiał za pomocą oceny pozytywnej i negatywnej oraz dodawać własne komentarze
- Użytkownicy mogą zgłaszać problemy za pomocą odpowiedniej zakładki dostępnej w panelu bocznym
- Dodanie panelu administracyjnego pozwalającego użytkownikom z odpowiednimi uprawnieniami na administrację użytkownikami, materiałami oraz aktywnościami dodawanymi przez użytkowników
- ~~Dodanie panelu ze statystykami użycia dla administratorów, który pozwala na podgląd najchętniej używanych przez~~
- ~~Dodatkowa aplikacja/modu pozwalający na generowanie sztucznego ruchu sieciowego – jak najbardziej zbliżonego do rzeczywistego ruchu wykonywanego przez użytkowników~~

4.4 Diagram przypadków użycia i diagram przepływu



4.5 Wybór technologii

- **JavaScript** - Jest to jezyk skryptowy oraz wieloparadygmatowych, służący do tworzenia prostych skryptów na stronach internetowych. Jest to jeden z trzech podstawowych narzędzi do tworzenia stron internetowych zaraz po HTMLu oraz CSSie, pomimo to nie jest tylko używany na stronach ale także np w plikach pdf.

- **React** - jest to open-sourcowy JavaScriptowy framework oraz biblioteka wykorzystywany do tworzenia interfejsów graficznych aplikacji mobilnych. Stworzony został przez programiste Facebooka Jordana Walkera, ispiracją dla tego frameworka było roszerzenie do jezyka PHP -XHP. Jedną z głównych cech tego frameworka jest wirtualny DOM.React przechowuje cały DOM aplikacji w pamięci, po zmianie stanu wyszukuje różnice między wirtualnym i prawdziwym DOM i aktualizuje zmiany. Drugą z cech szczególnych React jest język JSX. Jest on nakładką na JavaScript, która dodaje możliwość wstawiania kodu HTML (lub komponentów React) bezpośrednio w kodzie, zamiast ciągu znaków.

- **JWT** – JSON Web Token (JWT) to otwarty standard (RFC 7519), który definiuje zwarty i niezależny sposób bezpiecznego przesyłania informacji między stronami jako obiekt JSON. Te informacje można zweryfikować i zaufać, ponieważ są podpisane cyfrowo. JWT mogą być podpisywane przy użyciu tajnego klucza za pomocą algorytmu HMAC lub pary kluczy publiczny/prywatny przy użyciu RSA lub ECDSA.

- **GoLang** – wieloparadygmatowy język programowania opracowany przez pracowników firmy Google: Roberta Griesemera, Roba Pike'a oraz Kena Thompsona. Łączy w sobie łatwość pisania aplikacji charakterystyczną dla języków dynamicznych (np. Python, Lisp), jak również wydajność języków kompilowanych (np. C, C++). Głównymi założeniami twórców tego języka miała być prostota i połączenie wszystkiego co najlepsze ze świata programowania. Warto też zauważyć, że Go kompiluje się niezmiernie szybko w porównaniu do innych języków kompilowanych

- **Fiber** – jest to framework sieciowy inspirowany frameworkiem Express.js, oparty na Fasthttp, najszybszym silniku HTTP dla GoLanga. Stworzony, aby ułatwić prace oraz przyspieszyć programowanie z myślą o zerowej alokacji pamięci i wydajności. Umożliwia tworzenie API, które korzystają z dobrodziejstw języka Go, jednocześnie oferując intuicyjne opakowanie, pozwalające na łatwe uruchamianie aplikacji.

- **Docker** – Jest to otwarte oprogramowanie stworzone przez Solomona Hykesa. Docker został stworzony do wirtualizacji na poziomie systemu operacyjnego inaczej mówiąć pozwala nam na umieszczenie bibliotek, plików konfiguracyjnych, małych baz dan, itp. do małych oraz lekkich kontenerów.

- **Traefik** – Traefik to otwarto-źródłowy router brzegowy dla aplikacji, który sprawia, że publikowanie usług jest przyjemnym i łatwym doświadczeniem. Odbiera on żądania w imieniu systemu i uczy się, które komponenty są odpowiedzialne za ich obsługę. To co wyróżnia Traefik, oprócz jego wielu funkcji, to fakt, że automatycznie wykrywa

właściwą konfigurację dla usług. Magia dzieje się, gdy Traefik sprawdza infrastrukturę, gdzie znajduje odpowiednie informacje i odkrywa, która usługa obsługuje dane żądanie.

- **Minio S3** – każda aplikacja, a już przede wszystkim aplikacja streamingowa wymaga odpowiedniego systemu przechowywania plików. Minio S3 oferuje interfejs kompatybilny z narzędziami AWS, co czyni go świetnym zamiennikiem dla osób, które chcą mieć całą kontrolę nad swoją infrastrukturą. Minio oferuje również metody ograniczania dostępu oraz integracji z własnymi rozwiązaniami autoryzacyjnymi.

- **PostgreSQL**, także Postgres – obok MySQL i SQLite, jeden z najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych. Początkowo opracowywany na Uniwersytecie Kalifornijskim w Berkeley i opublikowany pod nazwą Ingres. Postgres jest jednym z niewielu systemów zarządzania bazami danych które oferują relacyjny model bazy danych łącznie z obiektowym dając najlepsze rozwiązania z obu modeli. Dzięki temu że PostgreSQL jest pod licencją typu open source to jest nie tylko darmowy ale również pozwala na dowolną modyfikacje kodu źródłowego dając możliwość dopasowania go do swoich wymagań. Będąc nieustannie usprawniany od prawie 20 lat, PostgreSQL jest jednym z najbardziej niezwodnych i funkcjonalnych systemów zarządzania bazą danych.

- **Stripe** - Stripe jest irlandzko-amerykańską firmą świadczącą usługi finansowe i oprogramowanie jako usługę z podwójną siedzibą w South San Francisco, Kalifornia, Stany Zjednoczone i Dublin, Irlandia. Firma oferuje przede wszystkim oprogramowanie do przetwarzania płatności oraz interfejsy programowania aplikacji dla stron internetowych e-commerce i aplikacji mobilnych.

5 Scenariusze

Scenariusz nr 1	
Tytuł	Rejestracja użytkownika
Aktorzy	Użytkownik, Aplikacja, Bramka SMTP
Warunki wejścia	Aktywna strona rejestracji, użytkownik nie posiada konta w serwisie
Przebieg	a) Aplikacja wyświetla formularz rejestracji b) Użytkownik wypełnia formularz logowania podając nazwe użytkownika, hasło , powtórzenie hasło oraz email nastepnie formularz przesyłany jest do aplikacji. c) Aplikacji sprawdza dane pod kątem poprawności oraz duplikacji ich w bazie danych d) Użytkownik otrzymuje od serwera wiadomość z linkiem weryfikacyjnym oraz potwierdza swój email
Zakończenie	Użytkownik zostaje zalogowany oraz otrzymuje wiadomość powitalną
Zakończenie alternatywne 1	Błąd, ponieważ użytkownik istnieje już w bazie
Zakończenie alternatywne 2	Błąd, ponieważ wprowadzone dane są błędne
Zakończenie alternatywne 3	Przekierowanie do strony błędu

Scenariusz nr 2	
Tytuł	Logowanie
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	Aktywna strona logowania, użytkownik posiada konto w serwisie
Przebieg	a) Aplikacja wyświetla formularz logowania zawierający pola na email oraz hasło b) Użytkownik wprowadza swój email oraz hasło do konta w formularzu logowania c) Dane są weryfikowane pod kątem poprawności
Zakończenie	Użytkownik zostaje przekierowany na stronę główną aplikacji.
Zakończenie alternatywne	Użytkownik zostaje przekierowany ponownie na stronę logowania i otrzymuje komunikat o błędzie.
Zakończenie alternatywne 1	Błąd, ponieważ użytkownik nie istnieje/wprowadził błędne dane
Zakończenie alternatywne 2	Błąd, ponieważ wystąpił wewnętrzny problem
Zakończenie alternatywne 3	Użytkownik został zbanowany, wyświetla się pop-up z informacją.

Scenariusz nr 3	
Tytuł	Resetowanie hasła
Aktorzy	Użytkownik, Aplikacja, Bramka SMTP
Warunki wejścia	Użytownik posiada konto oraz ma dostęp do skrzynki e-mail z nim skojarzonym.
Przebieg	a) Użytkownik przechodzi na ekran logowania i wybierz opcje ”nie pamiętam hasła”. b) Aplikacja przenosi użytkownika na formularz z opcją wpisania emaila. c) Wprowadza adresu skrzynki pocztowej swojego konta. d) Potwierdza chęć resetu hasła przyciskiem ”Zresetuj hasło” e) Użytkownik odbiera wiadomość, wykorzystuje link oraz zmienia swoje hasło użytkownika.
Zakończenie	Użytkownik pomyślnie zmienia swoje hasło na nowe.
Zakończenie alternatywne 1	Użytkownik użył linku, który został przedawniony, otrzymuje komunikat.
Zakończenie alternatywne 2	Użytkownik otrzymuje informacje o wewnętrznym problemie aplikacji w komunikacie.

Scenariusz nr 4	
Tytuł	Wykupienie subskrybcji.
Aktorzy	Użytkownik, Aplikacja, System Stripe
Warunki wejścia	Użytkownik jest zalogowany
Przebieg	a) Użytkownik otwiera boczne menu hamburger b) Wybranie opcji **Subskrybcja** c) Po przekierowaniu użytkownik jedną subskrybcję z listy i klika jej cenę d) Użytkownik zostaje przekierowany na stronę płatności Stripe e) Użytkownik dostaje informację o udanej płatności w formie wiadomości e-mail
Zakończenie	Uzytkownik zostaje przekierowany na stronę z informacją o poprawnie zakończonej płatności
Zakończenie alternatywne 1	Użytkownik dostaje informację o nieudanej płatności i dostaje możliwość jej ponowienia
Zakończenie alternatywne 2	Użytkownik już posiada subskrybcję i wyświetla się czas jej trwania oraz data jej zakupienia na przycisku subskrybcji jak i stronie do jej zakupu oraz możliwość dodania użytkownika do swojej subskrybcji

Scenariusz nr 5	
Tytuł	Dodanie użytkownika do swojej subskrybcji.
Aktorzy	Użytkownik, Aplikacja,
Warunki wejścia	a) Użytkownik jest zalogowany b) Użytkownik posiada subskrybcję
Przebieg	a) Użytkownik otwiera boczne menu hamburger b) Wybranie opcji **Subskrybcja** c) Użytkownik zostaje przekierowany na stronę z informacją na temat swojej subskrybcji (czas zakupu, ile zostało, kto jest dodatkowo przypisany do konta) d) Użytkownik wybiera przycisk **Dodaj** e) Wyświetla się pop-up proszący o wpisanie adresu email użytkownika, którego użytkownik chce dodać f) Użytkownik potwierdza wprowadzony e-mail za pomocą **Dodaj**
Zakończenie	Użytkownik zostaje dodany do listy oraz otrzymuje subskrybcję konta rodzica
Zakończenie alternatywne 1	Podany e-mail nie istnieje, zwracany jest błąd.
Zakończenie alternatywne 2	Użytkownik zostaje przekierowany na stronę scenariusza (błędu) z informacją o błędzie, który wystąpił.

Scenariusz nr 6	
Tytuł	Wyszukiwanie materiału
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	Użytkownik jest zalogowany
Przebieg	a) Użytkownik wybiera opcje wyszukiwania na stronie głównej. b) Użytkownik wprowadza nazwę interesującego go materiału do paska wyszukiwania lub wybiera interesującą go kategorię c) (Opcjonalnie) Użytkownik wprowadza filtry dotyczące zawartości, ocen, daty utworzenia etc.
Zakończenie	Na ekranie pojawia się materiał zgodny z filtrami wprowadzonymi przez użytkownika.
Zakończenie alternatywne	Użytkownik zostaje poinformowany, że nie istnieje materiał, który spełnia podane przez niego warunki.

Scenariusz nr 7	
Tytuł	Odtworzenie materiału
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada zakupioną subsrybcję • Finalizacja scenariusza nr 4
Przebieg	a) Użytkownik wybiera materiał z wyszukanych lub ze strony głównej. b) Użytkownik zostaje przekierowany na stronę z materiałem c) Liczba wyświetleń dla materiału rośnie o 1, jeśli poprzedni oglądany materiał nie był tym docelowym
Zakończenie	Materiał może teraz zostać odtworzony.
Zakończenie alternatywne	Film nie uruchamia się, a strona informuje o problemach i prosi, aby odświezyć stronę.

Scenariusz nr 8	
Tytuł	Odtworzenie materiału Premium
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada zakupioną subsrybcję • Finalizacja scenariusza nr 4
Przebieg	a) Użytkownik wybiera materiał z dostępnego katalogu. b) Serwer przekierowuje na podstronę odtwarzacza dla materiału c) Liczba wyświetleń dla materiału rośnie o 1, jeśli poprzedni oglądany materiał nie był tym docelowym
Zakończenie	Wyświetlona jest strona odtwarzacza z wybranym materiałem
Zakończenie alternatywne 1	Użytkownik otrzymuje pop-up z informacją o braku uprawnień do tego materiału
Zakończenie alternatywne 2	Użytkownik zostaje przekierowany według scenariusza (strona błędu) z kodem błędu 403 (Forbidden)
Zakończenie alternatywne 3	Film nie wczytuje się, a strona informuje o problemach i prosi, aby ją odświeżyć.

Scenariusz nr 9	
Tytuł	Operacje na materiale wideo
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Finalizacja scenariusza (strona materiał/materiał premium)
Przebieg	a) Użytkownik rozpoczyna/zatrzymuje odtwarzanie za pomocą przycisku Start/Stop b) użytkownik może wybrać głośność dla materiału za pomocą ikony głośnika po prawej stronie dolnego paska c) Wybierając moment na pasku postępu materiału, użytkownik może dowolnie wybrać moment odtwarzania. d) Za pomocą ikony zębatki po prawej stronie dolnego paska można wybrać jakość materiału wideo.
Zakończenie	Użytkownik dostaje możliwość swodobnego odbioru materiału
Zakończenie alternatywne 1	Użytkownik otrzymuje wiadomość pop-up z prośbą odświeżenia strony w związku z zaistniałym błędem
Zakończenie alternatywne 2	Użytkownik zostaje przekierowany według scenariusza (strona błędu) z błędem HTTP 403 lub 500

Scenariusz nr 10	
Tytuł	Dodanie oceny dla materiału
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Finalizacja scenariusza (strona odtwarzacza)
Przebieg	a) Użytkownik przenosi kursor w sekcję poniżej materiału wideo b) Użytkownik wybiera docelową reakcję, którą chce zamieścić. (ręka w górę, lub w dół) c) (Opcjonalnie) Reakcja może zostać usunięta, albo zamieniona na alternatywną
Zakończenie	Reakcja zostaje przypisana do materiału
Zakończenie alternatywne	Reakcja została już przypisana i została wyświetlona

Scenariusz nr 11	
Tytuł	Dodanie komentarza do materiału
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Finalizacja scenariusza (strona odtwarzacza)
Przebieg	a) Użytkownik przenosi kursor w sekcję poniżej opisu materiału zatytułowaną **Komentarze** b) Użytkownik w polu tekstowym wprowadza komentarz, który chce zamieścić na stronie. c) Użytkownik wybiera przycisk **Zamieść**
Zakończenie	Reakcja zostaje przypisana do materiału
Zakończenie alternatywne	Reakcja została już przypisana i została wyświetlona

Scenariusz nr 12	
Tytuł	Filtrowanie komentarzy pod materiałem
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Finalizacja scenariusza (strona odtwarzacza)
Przebieg	a) Użytkownik przenosi kursor w sekcję poniżej opisu materiału zatytułowaną **Komentarze** b) Za pomocą pola wyboru użytkownik wybiera sposób filtrowania (najlepiej ocenanie, najnowsze)
Zakończenie	Komentarze zostają przefiltrowane przy pomocy wybranej opcji

Scenariusz nr 13	
Tytuł	Reportowanie komentarza
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	a) Użytkownik jest zalogowany b) Finalizacja scenariusza (odtworzenie materiału)
Przebieg	a) Użytkownik scrolluje do sekcji z komentarzami, znajdującej się pod opisem materiału b) Po znalezieniu komentarza do zreportowania, użytkownik wciska **ikonę wykrzynika** w górnym rogu docelowego komentarza. c) Użytkownikowi pokazane zostaje okienko z polem tekstowym proszące o wprowadzenie powodu d) Użytkownik wybiera przycisk **Wyślij**.
Zakończenie	Okienko znika. Pojawia się powiadomienie o wysłaniu wiadomości.
Zakończenie alternatywne1	Wiadomość jest zbyt krótka, przycisk **Wyślij** jest niedostępny.
Zakończenie alternatywne 2	Użytkownik zamyka okienko przyciskiem **X**, nic więcej się nie pojawia.

Scenariusz nr 14	
Tytuł	Edycja swojego profilu użytkownika
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany
Przebieg	a) Użytkownik otwiera boczne menu hamburger b) Użytkownik wybiera opcję **Profil** c) Użytkownik zostaje przekierowany na stronę z danymi profilowymi d) Zostają wyświetlone informacje użytkownika (adres e-mail) oraz historia jego oglądania e) Użytkownik wybiera **pole z adresem e-mail** albo **placeholderem hasła** w celu ich zmiany (E-mail musi zsotać potwierdzony, co skutkuje odpowiednim zielonym znaczkiem) f) Użytkownik potwierdza zmianę za pomocą przycisku **Zapisz**
Zakończenie	Zmiany zostały zapisane na profilu użytkownika.
Zakończenie alternatywne 1	Użytkownik opuścił stronę profilu, a zmiany zostały anulowane.
Zakończenie alternatywne 2	Wyświetlenie komunikaty pop-up z błędem, który uniemożliwił zapisanie zmian

Scenariusz nr 15	
Tytuł	Komunikat o wygasającej subskrybcji
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada subskrybcję • Subskrybcja wygasa w ciągu 7 dni
Przebieg	a) Użytkownik przegląda serwis i konsumuje treści. b) Po czasie 5m od wejścia powinien pojawić się modal informujący o możliwości przedłużenia subskrybcji.
Zakończenie	Modal poprawnie się wyświetlił. Użytkownik może go zamknąć.

Scenariusz nr 16	
Tytuł	Przeglądanie historii oglądania
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany
Przebieg	a) Użytkownik otwiera boczne menu hamburger b) Użytkownik wybiera opcję **Profil** c) Użytkownik zostaje przekierowany na stronę z danymi profilowymi oraz historią
Zakończenie	Użytkownik może zobaczyć swoje statystyki dotyczące materiałów (ilosć wyświetleń, czas spędzony na oglądaniu, reakcje) oraz może kliknąć dany materiał, żeby przejść na stronę z materiałem.
Zakończenie alternatywne 1	Historia przeglądania jest pusta
Zakończenie alternatywne 2	Wystąpił błąd w momencie przejścia na profil, użytkownik zostaje przekierowany do scenariusza (błąd na stronie) z kodem błędu 403.

Scenariusz nr 17	
Tytuł	Przejście do panelu administracyjnego
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie
Przebieg	a) Użytkownik otwiera boczne menu hamburger b) Użytkownik wybiera opcję **Administracja** c) Użytkownik zostaje przekierowany do panelu administracyjnego
Zakończenie	Administrator posiada pełen dostęp do panelu administracyjnego
Zakończenie alternatywne 1	Użytkownik zostaje przekierowany do strony w scenariuszu (błąd na stronie) z kodem błędu 403.

Scenariusz nr 18	
Tytuł	Wyświetlenie listy użytkowników
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Administartor wybiera opcję "Zarządzanie użytkownikami" z menu bocznego. b) Aplikacja uruchamia menu z listą użytkowników. c) Administrator w specjalnym panelu wyszukuje użytkownika po nazwie, bądz filtrując po dacie utworzenia konta, ostatnim logowaniu itd. d)
Zakończenie	Administrator może teraz usuwać, banować czy modyfikować ustawienia skojarzone z użytkownikiem.
Zakończenie alternatywne 1	Informacja o braku użytkowników o określonych filtrach

Scenariusz nr 19	
Tytuł	Nieodwracalna operacja dla konta użytkownika
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (Wyświetlenie listy użytkowników)
Przebieg	a) Administrator wybiera jedną z dostępnych opcji (zmiana adresu e-mail, hasła) b) Po wybraniu opcji, administrator zostaje powitany oknem pop-up, które pozwala na wpisanie wymaganych informacji (dwa pola w celu potwierdzenia poprawności wprowadzanych danych) c) Administrator wybiera przycisk OK
Zakończenie	Użytkownik zostaje zmodyfikowany.
Zakończenie alternatywne 1	Wprowadzone informacje nie spełniają wymagań, pojawia się informacja o błędzie w obrębie wyskakującego okienka.

Scenariusz nr 20	
Tytuł	Wyświetlenie listy materiałów wideo
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Administrator wybiera opcję ”Materiały wideo” z menu bocznego. b) Użytkownik przekierowany jest na stronę z listą materiałów c) Komentarze mogą być filtrowane po kategorii filmu, ilości reportów, ocenach, czasie dodania, czy treści.
Zakończenie	Wyświetlenie listy materiałów, zgodnie z filtrami, które wybrał użytkownik.
Zakończenie alternatywne 1	Informacja o braku materiałów o podanych filtrach.

Scenariusz nr 21	
Tytuł	Dodanie mateirału wideo
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (wyświetlenie listy materiałów wideo)
Przebieg	a) Użytkownik wybiera przycisk ”Dodaj” b) Wyświetlone jest okienko pop-up, które wymaga wprowadzenia informacji tj. Tytuł, Opis, Kategoria, czyPremium, dodanie miniaturki. c) Użytkownik dodaje plik z materiałem wideo
Zakończenie	Materiał zostaje dodany i umieszczony na liście materiałów.
Zakończenie alternatywne 1	Wprowadzone informacje nie przeszły walidacji. Użytkownik otrzymuje informację o błędnie wprowadzonych danych
Zakończenie alternatywne 2	Podczas wysyłania materiału wystąpił błąd, użytkownik zostaje o tym poinformowany, a operacja dodawania materiału zostaje przerwana.

Scenariusz nr 22	
Tytuł	Edycja materiału wideo
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Administrator wybiera opcję ”Videos” b) Administrator filtruje listę wideo lub wybiera opcję edycji na jednym z już wyświetlonych. c) Administrator zostaje przeniesiony na stronę edycji materiału z wyświetlonym formularzem o elementach takich jak na stronie dodawania (już wypełnionych) d) Administrator zatwierdza wprowadzone zmiany przyciskiem ”Save”
Zakończenie	Materiał zostaje zedytowany.
Zakończenie alternatywne	Wyświetlony zostaje komunikat błędu.

Scenariusz nr 23	
Tytuł	Usuwanie materiału wideo
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Administrator wybiera opcję ”Videos” b) Administrator filtruje listę wideo lub wybiera opcję usunięcią na jednym z już wyświetlonych. c) Administrator potwierdza chęć usunięcia materiału i wybiera ”OK”
Zakończenie	Materiał zostaje usunięty.
Zakończenie alternatywne	Wyświetlony zostaje komunikat błędu.

Scenariusz nr 24	
Tytuł	Zarządzanie komentarzami
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Administartor wybiera opcję ”Komentarze” z menu bocznego. b) Aplikacja uruchamia sekcję z najnowszymi komentarzami na stronie. c) Komentarze mogą być filtrowane po kategorii filmu, ilości zgłoszeń, ocenach, czasie dodania, czy treści.
Zakończenie	Wyświetlana jest lista komentarzy, które zostały umieszczone na stronie.
Zakończenie alternatywne 1	Informacja o braku komentarzy przy określonych filtrach.

Scenariusz nr 25	
Tytuł	Zarządzanie komentarzami
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (wyświetlenie listy komentarzy)
Przebieg	a) Użytkownik zaznacza komentarz lub komentarze, dla których chce dokonać operacji b) Z górnego paska nad listą wybiera jedną z opcji (Usuń, ukryj, zbanuj użytkownika) c) Komentarze mogą być filtrowane po kategorii filmu, ilości zgłoszeń, ocenach, czasie dodania, czy treści.
Zakończenie	Wyświetlana jest lista komentarzy, które zostały umieszczone na stronie.
Zakończenie alternatywne 1	Informacja o braku komentarzy przy określonych filtrach.

Scenariusz nr 26	
Tytuł	Wyświetlenie listy subskrybcji
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	• Administartor wybiera opcję "Zarządzanie subskrybcjami" z menu bocznego. • Administrator zostaje przekierowany do podstrony z wszystkimi pakietami • Administrator może filtrować zakupione pakiety po adresie e-mail, dacie zakupu czy pozostałym czasie subskrybcji
Zakończenie	Wyświetla się lista zakupionych w aplikacji pakietów
Zakończenie alternatywne 1	Historia subskrybcji jest pusta, ponieważ nikt nie zakupił pakietów

Scenariusz nr 27	
Tytuł	Zarządzanie subskrybcjami
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Użytkownik jest zalogowany • Użytkownik posiada uprawnienia administratorskie • Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	• Administartor wybiera opcję ”Zarządzanie subskrybcjami” z menu bocznego. • Administrator zostaje przekierowany do podstrony z wszystkimi pakietami • Administrator może filtrować zakupione pakiety po adresie e-mail, dacie zakupu czy pozostałym czasie subskrybcji • Administrator wybiera pakiet.
Zakończenie	Administrator może anulować pakiet, potwierdzić jego płatność, albo dodać nowy pakiet (przedłużyć subskrybcję) dla użytkownika
Zakończenie alternatywne 1	Historia subskrybcji jest pusta, ponieważ nikt nie zakupił pakietów

Scenariusz nr 28	
Tytuł	Komunikat o braku uprawnień
Aktorzy	Użytkownik, Aplikacja
Warunki wejścia	• Wykonanie operacji, która skutkuje błędem HTTP lub użytkownik nie posiada wystarczających uprawnień
Przebieg	a) Przekierowanie na dedykowaną **stronę błędów**
Zakończenie	Wyświetlenie opisu błędu oraz przycisku powrotu do strony głównej.

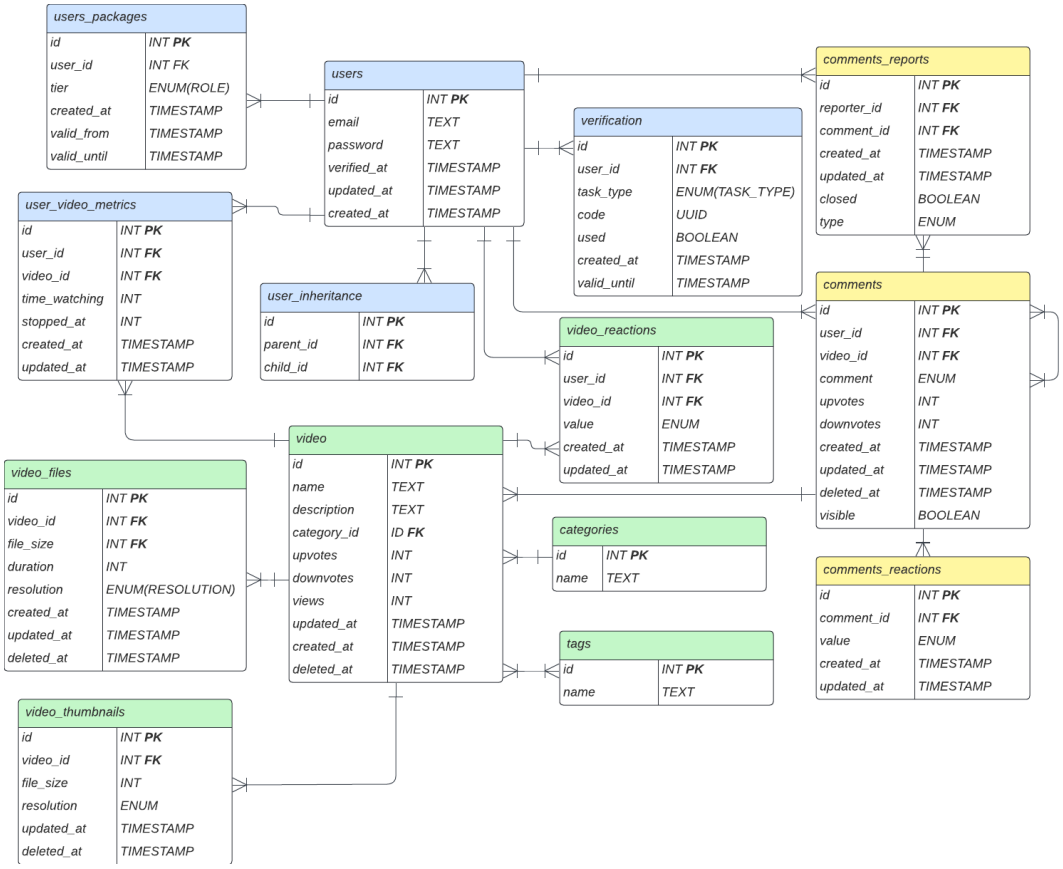
Scenariusz nr 29 (Odroczony)	
Tytuł	Przegląd statystyk oglądalności
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Otwarcie zakładki **Raporty** z menu bocznego b) Wybór opcji **Statystyki** c) Wybranie źródła danych (materiał, materiały lub kategoria) d) Doprecyzowanie oczekiwanych inforamcji (średnie wyświetlenia, wszystkie wyświetlenia, pozytywne/negatywne reakcje, komentarze) e) Dostosowanie przedziału czasowego f) Strona zostaje automatycznie odświeżona
Zakończenie	Wyświetlenie wykresu pokazującego statystyki oglądalności materiału zmieniającego się w czasie
Zakończenie alternatywne	Wyświetlenie informacji o niewsytarczającej ilości statystyk

Scenariusz nr 30 (Odroczony)	
Tytuł	Przegląd popularnych materiałów
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Otwarcie zakładki **Raporty** z menu bocznego b) Wybór opcji **Popularne** c) Wybranie kategorii materiałów d) Dostosowanie przedziału czasowego e) Automatyczne odświeżenie strony
Zakończenie	Wyświetlenie popularnych materiałów dla określonego przedziału czasowej
Zakończenie alternatywne	Wyświetlenie komunikatu o zbyt małej ilości danych statystycznych

Scenariusz nr 31 (Odroczony)	
Tytuł	Podgląd aktywności dla strony
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Finalizacja scenariusza nr. 28
Przebieg	a) Otwarcie zakładki **Raporty** z menu bocznego b) Wybór opcji **Aktywność** c) Wyświetlenie listy anoniomwych sesji z listami kroków d) Dostosowanie przedziału czasowego e) Automatyczne odświeżenie strony
Zakończenie	Wyświetlenie popularnych materiałów dla określonego przedziału czasowej
Zakończenie alternatywne	Wyświetlenie komunikatu o zbyt małej ilości danych statystycznych

Scenariusz nr 32 (Odroczony)	
Tytuł	Generowanie syntetycznego ruchu sieciowego dla użytkownika
Aktorzy	Administrator, Aplikacja
Warunki wejścia	• Finalizacja scenariusza (przejście do panelu administracyjnego)
Przebieg	a) Otwarcie zakładki **Generowanie ruchu** z menu bocznego b) Wybranie parametrów, na które położony ma zostać nacisk (długość oglądania, czas sesji, zainteresowania, ilość użytkowników) c) Wybranie przycisku **Generowanie** i potwierdzenie przyciskiem OK na wyskakującym okienu pop-up. d) Wyświetlenie okienka ładowania.
Zakończenie	Wyświetlenie komunikatu o poprawnie wykonanym zadaniu.
Zakończenie alternatywne	Wyświetlenie okienka pop-up z informacją o błędzie.

6 Diagram ERD



7 Estymacja czasowa

- **Rozbudowa dokumentacji - 8h**
- **Przygotowanie narzędzi potrzebnych do wykonania programu - 2h**
- **Wykonanie aplikacji po stronie backendu odpowiedzialnej za logowanie - 6h**
- **Stworzenie graficznego układu panelu logowania - 2h**
- **Stworzenie graficznego układu panelu rejstracji - 2h**
- **Implementacja logowania przy pomocy tokenu JWT 12h**
- **Impelementacja szkieletu strony głównej aplikacji - 5h**
- **Impelementacja szkieletu panelu użytkownika - 4h**
- **Impelementacja szkieletu panelu administracyjnego - 8h**
- **Projekt graficzny dla systemu resetowania hasła - 2h**
- **Implementacja systemu resetowania hasła - 2h**
- **Implementacja systemu weryfikacji adresu e-mail -5h**
- **Projekt strony odtwarzacza wideo - 5h**
- **Implementacja odtwarzacza wideo - 5h**
- **Projekt panelu zarządzania użytkownikami -10h**
- **Implementacja funkcjonalności zarządzania użytkownikami** *(crud)* **- 8h**
- **Projekt panelu zarządzania materiałami wideo - 3h**
- **Implementacja funkcjonalności zarządzania wideo** *(crud)* **- 12h**
- **Implementacja systemu zarządzania kategoriami - 4h**
- **Implementacja okienka z propozycją zakupu - 2h**
- **Projekt ekranu kupowania subskrybcji - 3h**
- **Projekt komponentu historii subskrybcji - 3h**
- **Wykonanie logiki systemu subskrybcji - 4h**
- **Podłączenie systemu płatniczego STRIPE - 3h**
- **Projekt + Implementacja zakładki do wysyłania błędów i skarg - 6h**
- **Implementacja systemu dziedziczenia subkrybcji - 4h**
- **Stworzenie działającej wyszukiwarki materiałów - 4h**

- **Wykonanie systemu filtrującego materiały -4h**
- **Wykonanie profilu użytkownika - 3h**
- **Wykonanie logiki dla profilu użytkownika - 2h**
- **Stworzenie wyświetlania historii oglądania - 4h**
- **Implementacja systemu historii oglądania przez użytkownika -4h**
- **Logika systemu wyświetlania materiałów w tej samej kategorii - 1h**
- **Aktualizacja interfejsu rekomendowanych materiałów w odtwarzaczu - 1h**
- **Stworzenie panelu odpowiedzialnego za administrowanie ocenami i komentarzami - 8h**
- **Dodanie strony graficznej dla komentarzy i ocen do odtwarzacza** - textbf4h
- **Wykonanie logiki systemu dodawania komentarzy do materiałów - 4h**
- ~~Zaimplementowanie panelu z statystykami - **8h**~~
- ~~Zaimpelementowanie funkcji odpowiedzialnej za generowanie raportów **-4h**~~
- ~~Konfiguracja serwera ElasticSearch - **8h**~~
- ~~Utworzenie systemu anonimowego zbierania statystyk dla sesji użytkowników - **16h**~~
- ~~Zaprojektowanie i implementacja panelu z list statystyk - **8h**~~
- ~~Utworzenie systemu uczącego si na zbieranych danych - **24h**~~
- ~~Implementacja systemu generowania statystyk użytkownania stony - **16h**~~

7.1 Wymagania MVP:

a) Działający formularz rejestracji.

b) Działający formularz logowania.

c) Możliwość resetowania hasła.

d) Potwierdzenie adresu e-mail.

e) Możliwość wyświetlania oraz odtwarzania materiałów na stronie.

f) Możliwość filtrowania materiałów na stronie.

g) Dodawanie komentarzy i ocen przez użytkowników pod materiałami.

h) Możliwość zakupu subskrybcji.

i) Możliwość współdzielenia subskrybcji pomiędzy użytkownikam

j) Działający system rekomendacji materiałów konkretnemu użytkownikowi.

k) Wyświetlanie historii oglądanych materiałów przez użytkownika.

l) Przeglądanie historii płatności danego użytkownika przez Administratora.

m) Działający system administracji dostępnymi na witrynie elementami.

n) Podstawowa możliwość administrowanie kontami użytkowników przez administratora (Dodawanie, usuwanie, banowanie)

o) Podstawowa możliwość administrowania materiałami (Dodawanie, usuwanie, edycja materiału)

p) Podstawowa możliwość administrowania komentarzami (Dodawanie, usuwanie, zgłaszanie)

q) ~~Zbieranie informacji o ruchu na stronie i działania użytkowników.~~

r) ~~Zbieranie informacji o materiałach na stronie obejrzanych przez użytkowników.~~

s) ~~Generowanie raportów dotyczących działa użytkowników na stronie.~~

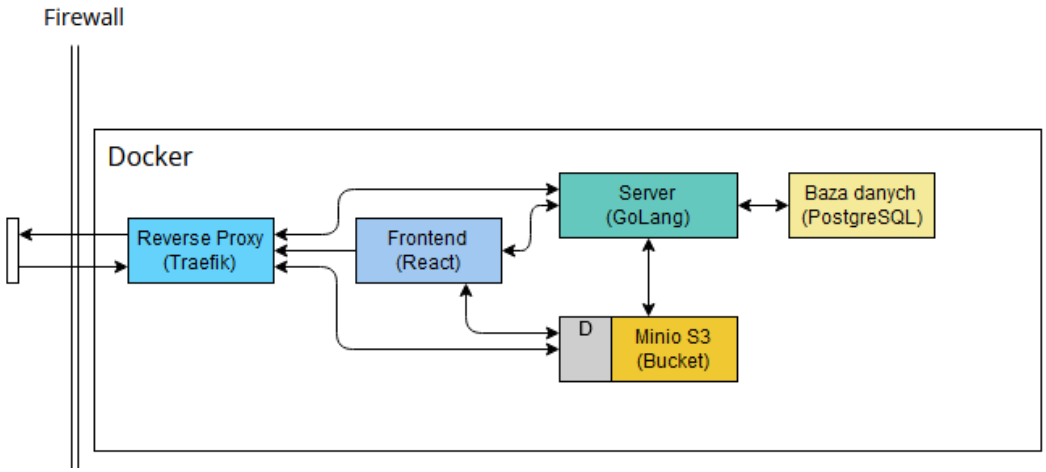
t) ~~Generowanie raportów dotyczących materiałów dostępnych na stronie.~~

8 Implementacja

8.1 Architektura aplikacji

W aplikacji zastosowana została architektura klient-serwer, która zakłada utworzenie osobnych aplikacji dla użytkownika końcowego oraz backendu. Zastosowanie architektury klient-serwer pozwala na znaczące odciążenie serwera oraz zwiększenie responsywności aplikacji kosztem większego zużycia zasobów po stronie klienta. Aplikacja uruchomiona jest za pomocą technologii konteneryzacyjnej Docker i składa się z 5 znaczących kontenerów, w skład których wchodzi:

- **Reverse Proxy** - odpowiedzialne za udostępnianie wspólnego punktu dostępowego dla wszystkich elementów, do których użytkownik musi mieć dostęp
- **Aplikacja kliencka** - napisana przy pomocy biblioteki React, udostępnia użytkownikowi interfejs, który do poprawnej pracy musi wykonywać zapytania HTTP do aplikacji serwerowej
- **Aplikacja serwerowa** - napisana w języku Go, odpowiada za funkcjonalność aplikacji i udostępnia interfejs w postaci Rest API, które oferuje operacje za pomocą zapytań HTTP.
- **Aplikacja na pliki** - wykorzystująca interfejs S3, aplikacja uruchomiona jest w alternatywnym do AWS S3 oprogramowaniu Minio S3, które pozwala na hostowanie plików we własnej przestrzeni. Przechowywane są tutaj duże pliki tymczasowe oraz docelowe materiały, do których użytkownik może uzyskać dostęp.
- **Baza danych** - PostgreSQL, którego zadaniem jest przechowywać informacje o płatnościach, użytkownikach, materiałach oraz pozwala na agregację pewnych informacji dotyczących, np. czasu oglądania.



Grafika 1: Diagram architektury aplikacji

8.2 Elementy reprezentowane przez tabele

Diagram ERD tabel oraz elementy, które się w nich znajdują można zaobserwować w sekcji nr. 4.4. Zastosowana struktura bazy danych celuje w częściową optymalizację obciążenia, kosztem integralności danych (zwłaszcza w wypadku ocen i komentarzy, które rezygnują z klauzuli COUNT(*) w wypadku liczenia ilości materiałów).

users	Główna tabela przechowująca informacje o użytkownikach, na tej podstawie umożliwiona jest autoryzacja użytkowników.
users_packages	W tej tabeli przechowywane są informacje o uprawnieniach użytkowników w formie pakietów z określonym okresem ważności, jeden użytkownik może posiadać wiele pakietów.
user_video_metrics	Tutaj przechowywane są informacje o czasie spędzonym przez użytkowników na oglądaniu materiałów, kiedy się zatrzymał i jakiego materiału to dotyczyło
user_inheritance	Przechowywane są tutaj współdzielone subskrybcje, które nawiązują do kont użytkowników - głównego (rodzica) i pobocznych (dzieci).
verification	Tabela przechowująca kody weryfikacyjne dla elementów takich jak resetowanie hasła, czy weryfikacja adresu e-mail. Kazdy kod posiada informację kiedy został utworzony, użyty, okres ważności oraz wartość, którą musi podać użytkownik w celu autoryzacji.
video	Istotna table przechowująca informacje o materiałach znajdujących się w systemie. W tej tabeli przechowywane są głównie metadane, które określają tytuł, opis, odzew widowni czy określają kategorię.
video_files	Tutaj przechowywana jest lokalizacja do fizycznych materiałów oraz metadane dla plików, wpisy odwołują się do tabeli ‘video‘.
video_thumbnails	W tej tabeli przechowywane są informacje o lokalizacji oraz metadane dla grafik materiałów, rekordy nawiązują do tabeli ‘video‘.
categories	Tabela przechowująca nazewnictwo dla poszczególnych kategorii, do których odwołuje się tabela ‘video‘.
tags	Relacja wiele do wielu, która jest dynamicznie poszerzana od nowe wpisy w momencie dodawania tagów, które jeszcze nie istnieją. Tagi pozwalają na określenie głównego nurtu dla materiałów
video_reactions	Przechowywana w tej relacji jest historia reakcji użytkowników, która wykorzystywana jest przy określaniu, czy użytkownik pozostawił już reakcję przy odpowiednim materiale.
comments	Przechowywane są tutaj komentarze użytkowników oraz informacje o reakcjach, które pozostawili inni użytkownicy. Komentarze dotyczą materiałów oraz przynależą do konkretnego użytkownika.
comments_reactions	W tej relacji przechowywana jest historia reakcji użytkowników, która odwołuje się do komentarzy z tabeli ‘comments‘.

Table 1: Informacja o typie danych przechowywanych w bazie danych

8.3 Przechowywanie konfiguracji dla aplikacji

W celu konfiguracji poszczególnych usług, przechowywania kluczy dostępowych do zewnętrznych API oraz konfiguracji ścieżek dla kontenerów - zastosowany został plik *.env*, który odczytywany jest przez aplikację backendową. W celu odczytania tego pliku zastosowany został moduł GoLang, który wykorzystuje bibliotekę viper. Przykładowa zawartość pliku *.env* prezentuje się następująco:

```
...

POSTGRES_USER=postgres
POSTGRES_PASSWORD=postgres
POSTGRES_DB=postgres

PGADMIN_DISABLE_POSTFIX=1
PGADMIN_DEFAULT_EMAIL=admin@admin.com
PGADMIN_DEFAULT_PASSWORD=123

...
```

Kod 1: Przykładowy plik .env

Konfiguracja odczytywana jest przed moduł, który udostępnia następującą funkcję:

```
func GetString(targetEnvVar string, defaultValue ...string) string {
  if viper.IsSet(targetEnvVar) {
    return viper.GetString(targetEnvVar)
  } else {
    for _, value := range defaultValue {
      return value
    }
    log.WithField("field", targetEnvVar).Panic("Environment variable is not set")
    return ""
  }
}
```

Kod 2: 'Przykładowa funkcja pobierająca string z konfiguracji'

W wypadku nie istniejącej wartości w pliku konfiguracyjnym, można zastosować domyślną wartość lub zwrócić krytyczny błąd uniemożliwiający uruchomienie aplikacji.

8.4 System połączenia z bazą danych - sqlc

Dzięki zastosowanemu w projekcie - SQLC, można definiować swoje zapytania SQL w oddzielnych plikach .sql, używając adnotacji do wskazania parametrów i typów zwracanych. Narzędzie następnie automatycznie generuje kod Go w oparciu o te definicje zapytań, zapewniając silnie typowane funkcje i struktury danych. Poniżej przykład metody wygenerowanej w oparciu o kod SQL.

```
CREATE TYPE ROLE AS ENUM ('free', 'premium', 'administrator');
CREATE TABLE users_packages (
  id BIGSERIAL PRIMARY KEY,
  user_id BIGINT,
  tier ROLE NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  valid_from DATE NOT NULL DEFAULT NOW(),
  valid_until DATE NOT NULL DEFAULT NOW()
);
```

Kod 3: Przykładowy plik - 'schema.sql'

```
-- name: GiveUserPackage :exec
INSERT INTO users_packages (
```

```
  user_id,
  tier,
  valid_from,
  valid_until
) VALUES (
  $1, $2, $3, $4
);
```

Kod 4: Przykładowy plik - ‘query.sql‘

```
    // Create database block
  ctx := context.Background()
  db, err := sql.Open("postgres", config.GetString("PG_CONNECTION_STRING"))
  if err != nil {
    log.WithField("err", err).Error("Could not create database connection")
    return c.SendStatus(fiber.StatusInternalServerError)
  }
  qtx := sqlc.New(db)
  defer db.Close()

    // Add package to database
  if err := qtx.GiveUserPackage(ctx, sqlc.GiveUserPackageParams{
    UserID:     sql.NullInt64{Int64: int64(data.UserID), Valid: true},
    Tier:       sqlc.Role(data.Tier),
    ValidFrom:  data.ValidFrom,
    ValidUntil: data.ValidUntil,
  }); err != nil {
    log.WithField("err", err.Error()).Error("Could not give user the package")
    return c.SendStatus(fiber.StatusInternalServerError)
  }
```

Kod 5: Implementacja funkcji w języku Go

8.5 Zastosowanie narzędzi kontrolujących jakość kodu - ESLint, Prettier, Husky

Wspólne korzystanie z ESLint, Prettier i Husky umożliwia zwiększenie jakości wysyłanego kodu. ESLint zapewnia statyczną analizę kodu, identyfikując potencjalne problemy i egzekwując standardy kodowania, zapewniając spójność i redukując liczbę błędów. Prettier dba o formatowanie kodu, utrzymując spójny styl w całej bazie kodu. Husky umożliwia zautomatyzowane uruchamiania ESLint i Prettier przed wysłaniem commita, zapobiegając zatwierdzeniu kodu z błędami. Ta kombinacja zapewnia czysty, dobrze sformatowany i wolny od błędów kod. Dodatkową zaletę jest uniemożliwienie wysłania wadliwego kodu, który może przeszkadzać we współpracy pomiędzy deweloperami.

Przykład wyników uruchomienia narzędzia ESLint:

8.6 Generowanie wiadomości z szablonów

Do wygenerowania szablonu musi zostać wykorzystany taki fragment kodu, który odpowiednio pobiera lokalizację szablonów na dysku. W przykładzie[6] można zaobserwować pobieranie

```
  11:10  warning  'data' is assigned a value but never used       @typescript-eslint/no-unused-vars
  11:16  warning  'setData' is assigned a value but never used    @typescript-eslint/no-unused-vars
  16:19  warning  'setIsValid' is assigned a value but never used @typescript-eslint/no-unused-vars
  18:9   warning  'navigate' is assigned a value but never used   @typescript-eslint/no-unused-vars

/home/user/Documents/college/dianomi/src/pages/SidePanel/SidePanel.tsx
  11:9  warning  'navigate' is assigned a value but never used  @typescript-eslint/no-unused-vars

/home/user/Documents/college/dianomi/src/pages/UserDashboard/UserDashboard.tsx
   4:10  warning  'Notify' is defined but never used       @typescript-eslint/no-unused-vars
 106:28  warning  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any

/home/user/Documents/college/dianomi/src/pages/VideoPlayerPage/VideoPlayer.tsx
   1:40  warning  'useRef' is defined but never used             @typescript-eslint/no-unused-vars
   5:10  warning  'Notify' is defined but never used             @typescript-eslint/no-unused-vars
  28:10  warning  'videoName' is assigned a value but never used @typescript-eslint/no-unused-vars
 101:17  warning  'request' is assigned a value but never used   @typescript-eslint/no-unused-vars

/home/user/Documents/college/dianomi/src/services/profile.service.tsx
  2:10  warning  'Package' is defined but never used  @typescript-eslint/no-unused-vars

✖ 44 problems (0 errors, 44 warnings)
```

Grafika 2: Lista problemów znaleziona przy pomocy narzędzia ESLint

wymaganych wartości, obsługę błędów w wypadku problemów na którymkolwiek z etapów generowania ciągu znakowego oraz samo umieszczenie gotowej wiadomości w zmiennej buf, która jest buforem na bajty odzwierciedlające znaki.

```
storagePath := config.GetString("APP_STORAGE_PATH", "./storage")
appUrl := config.GetString("APP_URL")
// Create verification code and url
code, err := queries.CreateVerificationCode(ctx, sql.NullInt64{
  Int64: userId,
  Valid: true,
})
if err != nil {
  log.WithField("error", err.Error()).Error("Could not create verification code")
}
verificationUrl := fmt.Sprintf("%s/auth/verify?validate=%s", appUrl, code.Code)
// Parse template
var buf bytes.Buffer
templatePath := fmt.Sprintf("%s/%s", storagePath, "templates/verification-email.
    html")
tmplt, err := template.ParseFiles(templatePath)
if err != nil {
  log.WithField("error", err.Error()).Error("Template could not be parsed")
  return
}
if err := tmplt.Execute(&buf, struct{ Url string }{Url: verificationUrl}); err !=
    nil {
  log.WithField("error", err.Error()).Error("Template could not be recreated")
  return
}
```

Kod 6: Generowanie szablonu e-mail z linkiem aktywacyjnym

8.7 Wysyłanie wiadomości e-mail

W celu wysłania wiadomości, musi ona najpierw zostać odpowiednio spreparowana. W dużym skrócie, tak jak to widać w tej funkcji[7], wiadomość składa się z nagłówka informującego jakiej wartości użytkownik powinien się spodziewać, linii **From** z autorem, linii **To** z docelowymi adresatami, linii **Subject** z tematem wiadomośći oraz samym środkiem wiadomości.

```
func SendEmail(mail Mail) error {
    // Skip template generation if smtp is disabled
    smtpEnabled := config.GetBool("APP_SMTP_ENABLED", false)
    if !smtpEnabled {
      return errors.New("smtp is disabled")
    }
    smtpHost := config.GetString("APP_SMTP_HOST", "")
    smtpPort := config.GetInt("APP_SMTP_PORT", -1)
    smtpUser := config.GetString("APP_SMTP_USER", "")
    smtpPassword := config.GetString("APP_SMTP_PASSWORD", "")
    smtpNoreplay := config.GetString("APP_SMTP_NOREPLAY", "")
    smtpHostAddr := fmt.Sprintf("%s:%d", smtpHost, smtpPort)

    // smtpTLS := config.GetString("APP_SMTP_TLS", "")
    msg := "MIME-version: 1.0;\nContent-Type: text/html; charset=\"UTF-8\";\r\n"
    msg += fmt.Sprintf("From: %s\r\n", smtpNoreplay)
    msg += fmt.Sprintf("To: %s\r\n", strings.Join(mail.To, ";"))
    msg += fmt.Sprintf("Subject: %s\r\n", mail.Subject)
    msg += fmt.Sprintf("\r\n%s\r\n", mail.Body)

    auth := smtp.PlainAuth("", smtpUser, smtpPassword, smtpHost)
    if err := smtp.SendMail(smtpHostAddr, auth, smtpNoreplay, mail.To, []byte(msg)
    ); err != nil {
      log.WithField("error", err.Error()).Error("")
      return errors.New("email could not be sent")
    }
    return nil
}
```

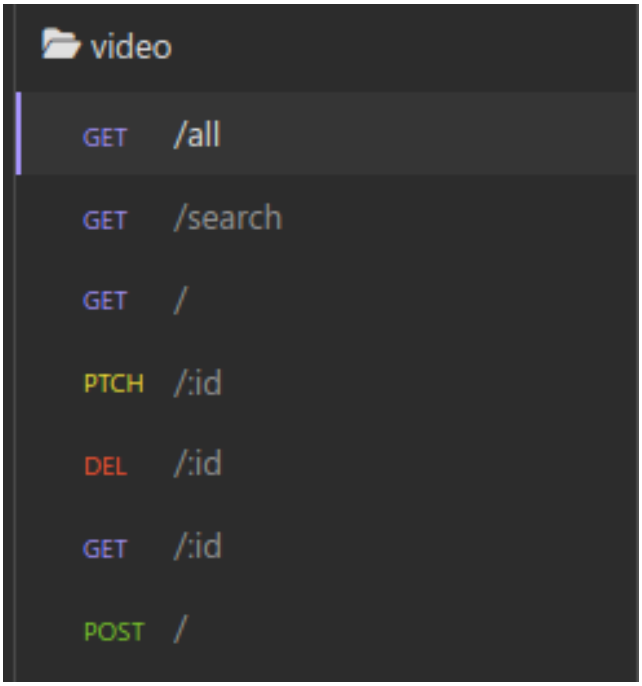
Kod 7: Funkcja odpowiedzialna za połączenie fragmentów wiadomości e-mail oraz jej wysłanie

8.8 Implementacja Rest API po stronie serwera

Serwer został zaprojektowany w konwencji REST API, która zakłada bezstanowe udostępnianie usług, które w związku z wymogiem posiadania systemu uwierzytelniania użytkowników zakłada użycie tokenów JWT, które pozwalają na autoryzację bez potrzeby tworzenia sesji dla użytkownika. Wykonane API udostępnia punkty HTTP, które mogą posiada różne metody. Przykład punktów końcowych dla usługi wideo można zauważyć na tej grafice[3].

Za przykład implementacji punktu końcowego przy pomocy frameworku Fiber może posłużyć ten odpowiedzialny za sprawdzanie zdrowia serwera:

```
// Some informations about service usage
func InternalHealthcheck(c *fiber.Ctx) error {
```



Grafika 3: Punkty końcowe dla usługi wideo

```
  if !slices.Contains(whitelist, c.IP()) {
    return c.SendStatus(fiber.StatusNotFound)
  }
  var m runtime.MemStats
  runtime.ReadMemStats(&m)
  return c.Status(fiber.StatusOK).JSON(fiber.Map{
    "healthy": true,
    "resources": fiber.Map{
      "allocMb":      bToMb(m.Alloc),
      "totalAllocMb": bToMb(m.TotalAlloc),
      "sysMb":        bToMb(m.Sys),
      "numGCMb":      bToMb(uint64(m.NumGC)),
    },
  })
}
```

Kod 8: API Helthcheck endpoint

Następnie tak zadeklarowany punkt końcowy może zostać udostępniony przy pomocy odpowiedniej deklaracji w głównym module aplikacji[9]

```
app := fiber.New()
app.Use(logger.New())
api := app.Group("/api/v1")

api.Get("/", mid.AuthMiddleware, func(c *fiber.Ctx) error { return c.SendStatus(
    fiber.StatusOK) })

// * Healthcheck
api.Get("/healthcheck", ctrl.Healthcheck)
```

```
api.Get("/_healthcheck", ctrl.InternalHealthcheck)
```

Kod 9: API Helthcheck endpoint

8.9 Wysyłanie zapytań do serwera API

W celu wysłania zapytań do serwera API, zastosowane zostają odpowiednie klasy, które odpowiadają poszczególnym usługom. Są one zadeklarowane w folderze *src/services* i pozwalają na zaimportowanie wewnątrz modułów. Zwracane są przez nie dwie wartości - **request**, który jest asynchroniczną obietnicą oraz **cancel**, który pozwala na przerwanie zapytania w dowolnym momencie (co jest istotne w wypadku debugowania przy pomocy ReactStrict, ponieważ operacje wykonywane są podwójnie).

Definicję takiej usługi można zaobserwować tutaj[10].

```
(...)
class TestService {
  GetOldPassword(data: userResData) {
    const controller = new AbortController();
    const request = http.post('/test/', data, { signal: controller.signal });
    return { request, cancel: () => controller.abort() };
  }
  (...)
}
```

Kod 10: Klasa zawierająca funkcje do komunikacji z serwerem API

Klasa możliwa jest potem do zaimportowania jako moduł co daje dostęp do jej funkcjonalności z poziomu innych komponentów. Zastosowanie tej metodyki jest swojego rodzaju fasadą dla skomplikowanego API, do którego zapytania można wykonywać za pomocą funkcji przyjmujących tyle unikatowych parametrów ile potrzeba.

Przykład wykorzystania tej klasy został zaprezentowany tutaj[11]

```
import adminService from '../../../services/admin.service';

    (...)

    useEffect(() => {
      if (userId === 0) return;
      const { request, cancel } = adminService.getUser(Number(userId));
      request
        .then(({ data }) => {
          setEmail(data?.email);
          setVerify(data?.verified);
          setReset(false);
        })
        .catch((err) => {
          if (err instanceof CanceledError) return;
          Notify.failure('Could not fetch information about user');
          onRequestClose();
        });
      return () => cancel();
```

```
20      }, [userId]);
```

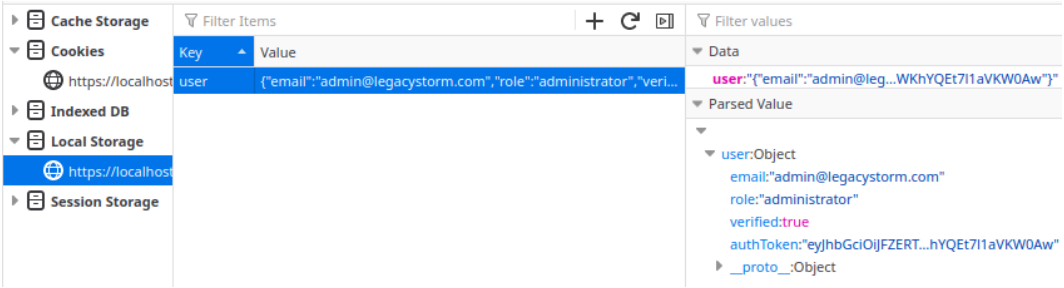
Kod 11: Praktyczne wykorzystanie usługi wewnątrz komponentu

8.10 System autoryzacji użytkowników

8.10.1 Frontend

Połączenie React, Axios i JWT (JSON Web Tokens) zostało tutaj zastosowane w celu zapewnienia dostępu do Rest API oraz usług dostępnych po stronie serwerowej. React służy jako framework frontendowy, zapewniając przyjazny dla użytkownika interfejs, podczas gdy Axios działa jako biblioteka klienta HTTP do wykonywania żądań API. JWT to bezpieczna i wydajna metoda przesyłania informacji uwierzytelniających między klientem a serwerem. Umożliwia ona serwerowi wygenerowanie tokena po pomyślnym zalogowaniu, który jest następnie wysyłany do klienta i przechowywany lokalnie.

Token jest przechowywany wewnątrz obiektu *User*, który przetrzymywany jest w **localStorage** przeglądarki użytkownika. Przykład został zaprezentowany na grafice 4. Zastosowanie tej metody pozwala dodatkowo na umieszczenie informacji, które mogą być istotne dla aplikacji klienta, np. czy użytkownik jest zalogowany, stan weryfikacji konta email, albo jakie posiada uprawnienia.



Grafika 4: Obiekt użytkownika przechowywany w localStorage

8.10.2 Backend

Po stronie serwera autoryzacja przebiega poprzez sprawdzanie odpowiednich informacji w bazie danych. Podczas rejestracji oraz logowania, istnienie użytkownika jest sprawdzane w tabeli *users* o definicji w kodzie 12.

```
1  CREATE TABLE users (
2    id BIGSERIAL PRIMARY KEY,
3    email VARCHAR(255) NOT NULL UNIQUE,
4    password VARCHAR(255) NOT NULL,
5    verified_at TIMESTAMP WITH TIME ZONE DEFAULT NULL,
6    created_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW(),
7    updated_at TIMESTAMP WITH TIME ZONE NOT NULL DEFAULT NOW()
8  );
```

Kod 12: Skrypt SQL tworzący tabelę ‘users‘

Weryfikacja poprawności wprowadzonych danych przebiega na zasadzie przedstawionej w sekcji 8.11, a użytkownik dodawany jest poprzez publiczną funkcję udostępnioną z modułu sqlc, co przedstawione jest w sekcji 8.4.

Aplikacja po poprawnej weryfikacji danych dostępowych, generuje odpowiedni token JWT, który przechowuje wszystkie informacje o użytkowniku, o posiadanych przez niego uprawnieniach oraz czas ważności, po którym będzie musiał ten token odświeżyć (zalogować się jeszcze raz). Przykład token JWT można zaobserwować na tym fragmencie kodu[13], a przykładową odpowiedź od serwera można zaobserwować tutaj[14]

```
{
  "aud": "dianomiTV",
  "exp": 1686418841,
  "iat": 1686332441,
  "iss": "dianomiTV",
  "jti": "ci1m86djtb6s7s5ha99g",
  "nbf": 1686332441,
  "role": "administrator",
  "sub": 2,
  "verified": true
}
```

Kod 13: Przykładowy token JWT

```
{
  "status": "success",
  "data": {
    "email": "admin@test.com",
    "role": "administrator",
    "token": "eyJhbGciOiJFZERTQSIsInR5cCI6IkpXVCJ9...",
    "verified": true
  }
}
```

Kod 14: Odpowiedź po udanym zalogowaniu

Informacje uzyskane podczas logowania zostają przechowane po stronie klienta, o czym więcej w sekcji 8.10.1.

8.10.3 S3 (Minio)

Autoryzacja użytkowników w usłudze Minio S3 składa się z 4 etapów:

a) Wysłanie przez klienta zapytania o tymczasowy token sesji do serwera S3 (z tokenem sesji wygenerowanym przez backend)

b) Minio następnie przekierowuje token otrzymany od użytkownika i przesyła zapytanie z prośbą o weryfikację do serwera aplikacji

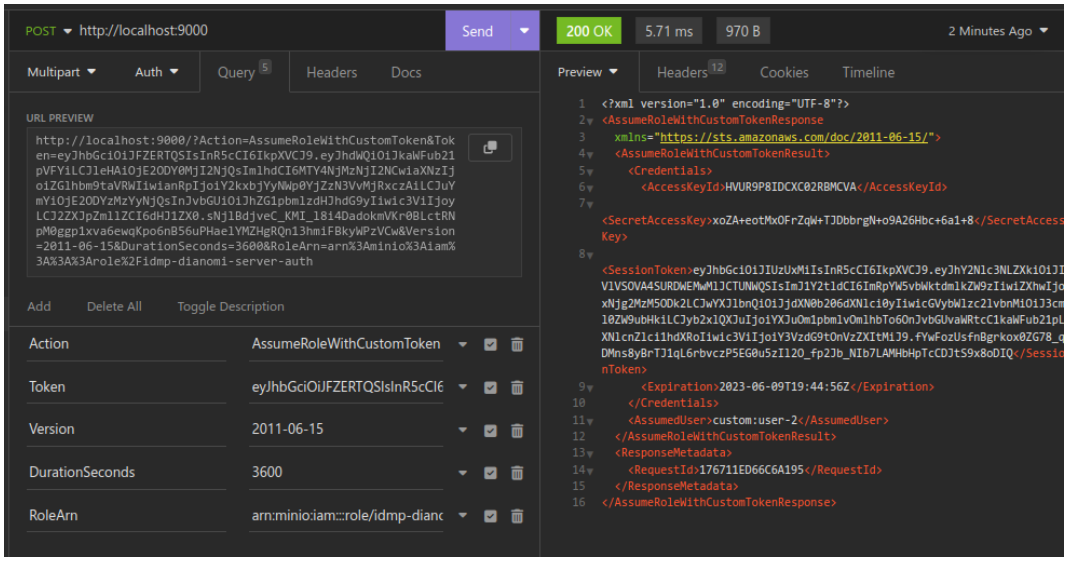
c) Serwer aplikacji potwierdza tożsamość użytkownika, albo jej zaprzecza i zwraca odpowiednią informację do serwera Minio

d) Serwer następnie generuje odpowiedź dla użytkownika z odpowiednimi tokenami, albo informuje o braku uprawnień

Wskazana wyżej sekwencja działań pozwala uniknąć potrzebę tworzenia dodatkowego konta w celu uzyskania dostępów do serwera plików. Tym samym jedno konto z uprawnieniami administratora otrzymuje możliwość wysyłania plików.

```
{
  "claims": {
    "bucket": "dianomi-videos",
    "permissions": "writeonly"
  },
  "maxValiditySeconds": 3600,
  "user": "user-2"
}
```

Kod 15: Potwierdzenie tożsamości użytkownika przez aplikację



Grafika 5: Wysłanie tokenu do serwera Minio w celu autoryzacji

8.10.4 Weryfikacja tokenu

W aplikacji zastosowany został algorytm wykorzystujący asymetryczne podpisywanie tokenów JWT przy pomocy klucza ed25519. Klucz jest automatycznie wygenerowany, jeśli administrator nie wprowadzi swojej wartości. Wykorzystanie tej metody pozwala podpięcie zewnętrznych usług, które będą korzystać z uwierzytelniania udostępnianego na serwerze aplikacji.

Token weryfikowany jest przez middleware, który podpięty jest w punktach końcowych, które wymagają jego obecności. Dostępne Middleware-y to **AuthMiddleware** i **AdminMiddleware**, gdzie ten drugi jest zależny od sprawdzenia przez pierwszy.

Weryfikacja zakłada zdekodowanie tokenu, sprawdzenie jego poprawności, sprawdzenie czy nadal jest aktualny i czy nie został wpisany na listę zablokowanych tokenów.

```
func AuthMiddleware(c *fiber.Ctx) error {
  // Extract header with token
```

```
  splitToken := strings.Split(c.Get("Authorization"), "Bearer ")
  // Check if token was provided
  if len(splitToken) < 2 {
    log.WithField("token", splitToken).Debug("User tried connection without token"
    )
    return c.Status(fiber.StatusUnauthorized).JSON(mod.Response{
      Status: "error",
      Data:   "Missing authorization token",
    })
  }
  // Extract claims
  token := splitToken[1]
  claims, err := jwt.ExtractClaims(token)
  if err != nil {
    log.WithFields(log.Fields{
      "token": splitToken,
      "err":   err.Error(),
    }).Debug("Invalid authorization token")

    return c.Status(fiber.StatusUnauthorized).JSON(mod.Response{
      Status: "error",
      Data:   "Invalid authorization token",
    })
  }
  // Check if token is valid
  now := time.Now().Unix()
  tokenSub := uint64((*claims)["sub"].(float64))
  tokenExp := time.Unix(int64((*claims)["exp"].(float64)), 0)
  tokenNbf := time.Unix(int64((*claims)["nbf"].(float64)), 0)
  tokenJti := (*claims)["jti"].(string)
  verified := (*claims)["verified"].(bool)
  // Is user verified
  if !verified {
    log.WithFields(log.Fields{
      "verified": verified,
      "sub":      tokenSub,
    }).Debug("User is not verified")
    return c.SendStatus(fiber.StatusUnauthorized)
  }
  // Is token valid
  if now > tokenExp.Unix() ||
    now < tokenNbf.Unix() ||
    jwt.IsTokenRevoked(tokenJti) {
    log.WithField("jti", tokenJti).Debug("Usage of unauthorized/revoked token")
    return c.SendStatus(fiber.StatusUnauthorized)
  }
  // Extract additional data
  role := (*claims)["role"].(string)
  c.Locals("sub", tokenSub)
  c.Locals("role", role)
  c.Locals("jti", tokenJti)
  c.Locals("exp", tokenExp)
  c.Locals("verified", verified)
  return c.Next()
```

```
}
```

Kod 16: Middleware odpowiedzialny za sprawdzanie tokenu JWT

Analogicznie middleware odpowiedzialny za sprawdzenie uprawnień administratorskich sprawdza tylko czy użytkownik posiada stosowną rolę. Rozwiązanie zastosowane w tym systemie niesie ze sobą pewne konsekwencje, tj. w związku z pomijaniem sprawdzania bazy danych, użytkownik, który posiada token, uzyskuje dostęp do wszystkich swoich uprawnień, nawet jeśli jego konto zostało usunięte. Dlatego też zaimplementowany musiał zostać system blokujący odpowiednie tokeny, jeśli nie chcemy, aby uzytkownik już z nich korzystał (np. został wylogowany).

```
  // * Video group
  video := api.Group("video", mid.AuthMiddleware)
  video.Get("/search", mid.AuthMiddleware, videoCtrl.VideoSearch)
  video.Post("/", mid.AdminMiddleware, videoCtrl.PostVideo)
  video.Get("/all", mid.AdminMiddleware, videoCtrl.GetAllVideos)
  video.Get("/:id", videoCtrl.GetVideo)
  video.Get("/", videoCtrl.GetRecommendedVideos)
  video.Delete("/:id", videoCtrl.DeleteVideo)
  video.Patch("/:id", videoCtrl.PathVideo)
```

Kod 17: Podłączenie pośrednika do punktu końcowego

```
// This is an interface to be replaced with solution of choice
package jwt
import (
  "time"
  "golang.org/x/exp/slices"
)
var (
  revokedTokens []string
)
func IsTokenRevoked(jti string) bool {
  return slices.Contains(revokedTokens, jti)
}
func RevokeToken(jti string, validUntil time.Time) {
  revokedTokens = append(revokedTokens, jti)
}
```

Kod 18: Funkcja odpowiedzalna za dodanie tokenu do czarnej listy

8.10.5 Weryfikacja konta e-mail

Ta operacja nie wymaga implementacji po stronie aplikacji klienckiej. Jedynym elementem może być dodanie komunikaty o niezweryfikowanym koncie. Jest to w pełni bezpieczne, ponieważ po stronie aplikacji serwerowej istnieje już zabezpieczenie, które nie pozwala niezweryfikowanym użytkownikom na wykonywanie operacji.

Proces weryfikacji użytkownika składa się z etapów:

a) Użytkownik najpierw musi się zarejestrować

b) Generowany jest specjalny UUID, który skojarzony jest z operacją uwierzytelniania konta użytkownika

c) Na podstawie szablonu generowana jest wiadomość z linkiem, który składa się z adresu do punktu, w którym odbywa się weryfikacja oraz kodu potwierdzającego tożsamość użytkownika.

d) Wiadomość zostaje wysłana na wskazany przez użytkownika adres e-mail

e) Użytkownik odwiedza stosowny link, a jego konto zostaje aktywowane

Zanim wiadomość może zostać wysłana, odpowiedni szablon musi zostać wykonany, więcej w sekcji 8.6. Zanim wiadomość e-mail zostanie wysłana, musi ona najpierw zostać utworzona z ciągów znakowych. W związku ze standaryzacją struktury e-mail, utworzenie wiadomości HTML sprowadza się do odpowiedniej konkatenacji zawartości.

```
if err := SendEmail(Mail{
  To:      []string{userEmail},
  Subject: "DianomiTV - Account verification",
  Body:    buf.Bytes(),
}); err != nil {
  log.WithFields(log.Fields{
    "err": err.Error(),
    "url": verificationUrl,
  }).Error("Verification email could not be sent.")
  return
}
```

Kod 19: Wywołanie funkcji wysyłającej e-mail oraz obsłużenie błędów

Więcej informacji na temat funkcji wysyłającej wiadomość w sekcji 8.7

8.10.6 System resetowania hasła

W celu zresetowania hasła, wykorzystane muszą zostać obie aplikacje - kliencka i serwerowa. Resetowanie hasła zakłada wygenerowanie unikatowego kodu i umieszczenie go w tabeli ‘verification‘. Od użytkownika wymagane jest podanie typu kodu oraz użytkownika,

```
// Create reset code
code, err := queries.CreateResetCode(ctx, sql.NullInt64{
  Int64: user.ID,
  Valid: true,
})
if err != nil {
  log.WithField("error", err.Error()).Error("Could not create reset code")
}
frontUrl := config.GetString("APP_FRONT_URL")
resetUrl := fmt.Sprintf("%s/reset-password?validate=%s", frontUrl, code.Code)
```

Kod 20: Funkcja dodającą odpowiednią wartość do tabeli ‘verification‘

Do wygenerowania unikatowego adresu UUID, wykorzystywana jest wbudowana w silnik bazodanowy funkcja (która używana jest jako wartość domyślna):

```
CREATE TYPE VERIFY_EMAIL_TYPE AS ENUM ('emailVerification', 'emailChange', '
    passwordReset');
CREATE TABLE verification (
  id BIGSERIAL PRIMARY KEY,
  user_id BIGINT,
  task_type VERIFY_EMAIL_TYPE NOT NULL,
  code UUID NOT NULL DEFAULT gen_random_uuid(),
  used BOOLEAN NOT NULL DEFAULT FALSE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  valid_until TIMESTAMP WITH TIME ZONE DEFAULT (NOW() + interval '15 minutes'),
  CONSTRAINT fk_user_verification
    FOREIGN KEY(user_id)
      REFERENCES users(id)
      ON UPDATE CASCADE
      ON DELETE SET NULL
);
```

Kod 21: Definicja tabeli automatycznie wprowadzającej wartość UUID

8.11 Walidacja wprowadzanych danych

Walidacja jest bardzo istotnym aspektem z punktu widzenia aplikacji klienckiej, jak i serwerowej. W wypadku aplikacji graficznej, istotne jest, aby użytkownik miał świadomość problemów z wprowadzonymi przez siebie danymi. W wypadku aplikacji serwerowej, walidacja jest etapem, który pozwoli na uniknięcie problemów integralności danych, czy zapobiegnie potencjalnym atakom, które mogą zakłócić działanie aplikacji.

8.11.1 Frontend

Walidacja na tym etapie aplikacji zakłada zastosowanie dwóch elementów, lokalnego sprawdzania przed wysłaniem formularza oraz sprawdzenia odpowiedzi od serwera i rozpoczęcia odpowiedniej akcji bazującej na odpowiedzi. Sprawdzanie po stronie klienckiej wykonywane jest za pomocą regexów i instrukcji warunkowych [22].

```
  if (!/^\S+@\S+\.\S+$/.test(regEmail)) {
    setIsValid(false);
  }

  ...
  {!isValid && <p className="alert alert-danger">Please enter a valid email.</p>}
  {regPassword != regPasswordRepeat && regPasswordRepeat != '' && (
    <p className="alert alert-danger" style={{ marginTop: '5px' }}>
      {' '}
      The passwords are not identical <br />{' '}
    </p>
  )}
```

Kod 22: Weryfikowanie wprowadzonych danych wewnątrz komponentu

Istotnym etapem jest również przechwytywanie informacji o problemach i wyświetlanie stosownego komunikatu, które zaimplementowane jest w sposób przedstawiony tutaj [23]

```
const { request } = videoService.takeVideo();
request
  .then((res) => {
    (...)
  })
  .catch((err) => {
    Report.failure(
      'Problem with fetching videos',
      `Video could not be fetched from the server. Message: ${err.message}`,
      'Okay',
    );
  });
```

Kod 23: 'Reagowanie na informacje o błędach'

8.11.2 Backend

Po stronie serwera istotne jest dokładne sprawdzenie danych i zwrócenie stosownego komunikatu w momencie wystąpienia problemu. Dane przyjmowane są w postaci obiektów JSON, które następnie parsowane są do odpowiednich obiektów, zadeklarowanych w ten sposób[24].

```
type VideoPatchData struct {
  Name        string   `json:"name" validate:"required"`
  Description string   `json:"description" validate:"required"`
  CategoryId  int64    `json:"category_id" validate:"required"`
  Premium     bool     `json:"is_premium"`
  Tags        []string `json:"tags" validate:"required,tags"`
}
```

Kod 24: Deklaracja obiektu przechowującego dane

Tak zadeklarowany struct przyjmuje dane, które prezentują się w ten sposób:

```
{
  "name": "nowa nazwa",
  "description": "lorem ipsum dolorem",
  "category_id": 1,
  "tags": [
    "tagjeden",
    "tagdwa",
    "tagtrzy"
  ]
}
```

Proces walidacji w wypadku serwera zakłada trzy etapy:

a) Etap konwersji ciągu znakowego JSON do obiektu, do którym można następnie dowolnie operować.

b) Sprawdzenie poprawności danych przy pomocy biblioteki **go-playground/validator**

c) Sprawdzenie integralności bazy przed wykonaniem operacji (dane już istnieją/nie istnieją)

W wypadku wystąpienia błędów, na którymkolwiek z etapów zostanie zwrócona odpowiednia informacja do użytkownika, a sam incydent zostanie umieszczony w logach. (Pominięto tutaj 3 krok)

```
// * PARSE DATA
var userData *mod.FormRegisterUser
if err := c.BodyParser(&userData); err != nil {
  return c.SendStatus(fiber.StatusBadRequest)
}
 // * VALIDATE DATA
err = mod.Validate.Struct(userData)
if err != nil {
  log.WithField("err", err).Debug("Could not validate user data")
  return c.Status(fiber.StatusBadRequest).JSON(mod.Response{
    Status: "error",
    Data:   "Incorrect registration information provided",
  })
}
```

Kod 25: Weryfikacja poprawności wprowadzonych danych

8.12 Zastosowany system paginacji

W celu zabezpieczenia serwera przed przeciążeniem i użytkownika przed potrzebą odczytywania możliwych tysięcy rekordów zastosowany został system paginacji, który przyjmuje parametry określające przesunięcie, które użytkownik sobie życzy:

```
https://.../api/v1/video/recommended?offset=2
```

Kod 26: Przykładowe zapytanie do serwera API o materiały

Zapytanie SQL, które odpowiada za wyciągnięcie materiałów znajduje się tutaj:

```
-- name: GetAllVideos :many
SELECT
  v.id id,
  v.name name,
  v.description description,
  c.name category,
  v.upvotes upvotes,
  v.downvotes downvotes,
  v.views views,
  v.is_premium is_premium,
  th.file_name as thumbnail
FROM
  video v LEFT JOIN categories c ON v.category_id = c.id
  LEFT JOIN video_thumbnails th ON th.video_id = v.id
LIMIT $1
OFFSET $2;
```

Kod 27: Zapytanie z ograniczeniem wyników

Po stronie programowej operacja odczytywania materiałów odbywa się w ten sposób:

```
func GetVideo(c *fiber.Ctx) error {
  var offset int32 = 0
```

```
  offsetArray := c.Query("offset")

    ...

  res, err := qtx.GetRandomVideos(ctx, sqlc.GetRandomVideosParams{
    Limit:  25,
    Offset: offset,
  })

    ...
```

Kod 28: Zapytanie z ograniczeniem wyników

8.13 Wyszukiwanie treści na stronie głównej

W pierwszej kolejności sprawdzane jest czy w polu wyszukiwania występuje jakaś wartość i jeśli jej nie ma to wyświetlana jest lista polecanych materiałów z punktu końcowego */api/v1/video/recommended*. W przeciwnym wypadku wykorzystywany jest punkt */api/v1/video/search?phrase=[f* gdzie [fraza] jest szukanym tytułem materiału.

Po stronie SQL zapytanie to zostało zaimplementowane w ten sposób:

```
-- name: GetVideoIDByName :many
SELECT id FROM video WHERE LOWER(name) LIKE LOWER($1);
```

Tak otrzymana lista numerów ID, następnie może na poziomie serwera zostać przekształcona na materiały, które następnie zostaną wysłane użytkownikowi w postaci listy.

```
// Look for video with phrase
var results []int64
for _, key := range search {
  items, err := qtx.GetVideoIDByName(ctx, fmt.Sprintf("%%%s%%", key))
  if err != nil {
    log.WithField("err", err).Error("Could not get data from database")
    return c.SendStatus(fiber.StatusInternalServerError)
  }
  // Add missing items
  for _, item := range items {
    if !slices.Contains(results, item) {
      results = append(results, item)
    }
  }
}
```

Kod 29: Pobieranie listy ID z bazy na podstawie podzielonego ciągu znakowego

8.14 System odwarzacza wideo

W celu wyświetlenia wszystkich wymaganych elementów na stronie, aplikacja kliencka musi wykonać parę zapytań w kilka miejsc, wszystkie te zapytania znajdują się w funkcji useEffect(), która wykonywana jest po wyrenderowaniu elementów na stronie i zgodnie ze sztuką pozwala pytać zewnętrzne usługi o dane aplikacji.

Na stronie odtwarzacza znajdują się główne elementy:

- Materiał wideo z możliwością wyboru rozdzielczości, opisem oraz tagami
- Komentarze użytkowników oraz ocena (Ręka w górę, w dół)
- sekcja z rekomendowanymi treściami

Otwierając materiał o adresie wyglądającym w ten sposób **/VideoPlayer/:id**, gdzie *:id* jest unikatowym id dla materiału, zostają wysłane zapytania do enpointów:

- **/video?offset=**$offset$&$limit$ **=limit** - lista rekomendowanych materiałów (wraz z ich limitem)
- **/video/:id** - informacja o materiale wideo
- **/video/comment/:id** - lista komentarzy dla odpowiedniego wideo

8.14.1 Dane o materiale otrzymywane od serwera

Informacje o materiale wideo odbierane są w postaci pliku JSON, w którym można znaleźć wszystkie potrzebne właściwości.

```
{
  "id": 1,
  "name": "Vitae turpis massa sed...",
  "category": "yup",
  "category_id": 1,
  "upvotes": 0,
  "downvotes": 0,
  "views": 0,
  "IsPremium": false,
  "thumbnail_url": "chp3cb5jtb6shseg48e0crysis3.mp4",
  "videos": [
    {
      "resolution": "360p",
      "duration": 0,
      "file_path": "1/360p.mp4"
    },
    {
      "resolution": "480p",
      "duration": 0,
      "file_path": "1/480p.mp4"
    },
    {
      "resolution": "720p",
      "duration": 0,
      "file_path": "1/720p.mp4"
    }
  ],
  "tags": [
    "test",
    "tester"
  ]
},
```

Kod 30: Informacje o materiale wideo

Dane następnie przechowywane są w obiekcie o zadeklarowanym wcześniej interfejsie po stronie klienta:

```
interface VideoItemData {
  id: number;
  name: string;
  description: string;
  category: string;
  tags: string[];
  thumbnail_url: string;
  videos: VideoData[];
}
export const VideoPlayer = () => {
    ...

  useEffect(() => {
    const { request } = videoService.takeVideoId(VideoIdInt);
    request
      .then((res) => {
        if (res.data.IsPremium === true && user?.role === 'free') {
          navigate('/');
        }
        setVideoName(res.data.name);
        setDataVideo(res.data.videos);
        setVideoThumbnail(res.data.thumbnail_url);
        setVideoTags(res.data.tags);
        setVideoDescription(res.data.description);
      })
      .catch((err) => {
        console.log(err);
      });
  }, []);

    ...
}
```

Kod 31: Interfejs oraz funkcja odczytująca wideo z bazy

8.14.2 Zbieranie statystyk oglądania

Dodatkowo zbierane są informacje o czasie jaki użytkownik poświęcił na oglądanie. Po stronie klienta, wykonywana jest funkcja, która uruchamia się w momencie jak wideo zmieni swoją wartość na pasku. Następnie informacje przesyłane są z częstotliwością 2,5s na serwer.

```
  const handleProgress = (state: { playedSeconds: React.SetStateAction<number> })
    => {
    const playedSecondsString = state.playedSeconds.toString(); // Konwersja na
    string

    setPlayedSeconds(parseInt(playedSecondsString));
    console.log(playedSeconds);
    setIsRunning(true);
  };

```

```
  useEffect(() => {
    let interval: NodeJS.Timeout;

    if (isRunning) {
      interval = setInterval(() => {
        const Data = {
          email: user?.email,
          video_id: VideoIdInt,
          time_spent_watching: 2,
          stopped_at: playedSeconds,
        };
        const { request } = profileService.PostVideoMertics(Data);
      }, 2500);
    }

    return () => {
      clearInterval(interval);
    };
  }, [isRunning]);
}
```

Kod 32: Funkcjonalność zbierająca czas oglądania po stronie klienta

Po stronie serwera dane odbierane są na punkt końcowy */api/v1/metrics*, który umieszcza informacje w bazie.

8.15 Wyświetlanie materiałów na stronie głównej

W celu wyświetlenia materiałów na stronie głównej odpytywany jest punkt końcowy */api/v1/video/recommended*, który zwraca listę materiałów w tej samej konwencji co w kodzie 30. Zwrócona lista ograniczona jest do 25 elementów, a paginacja tej listy odbywa się w ten sposób po stronie klienta, przy pomocy komponentu **Paginate**, w ten sposób:

```
const Paginate = ({ postsPerPage, totalPosts, paginate }: props) => {
  const pageNumbers: number[] = [];
  for (let i = 1; i <= Math.ceil(totalPosts / postsPerPage); ++i) pageNumbers.push
    (i);
  return (...);
}
```

Kod 33: Komponent odpowiedzialny za paginację zawartości

Zwracanym elementem jest lista, która pozwala na wybranie numeru strony z materiałami wideo. Sam komponent przyjmuje wartości, które pozwalają określić zagęszczenie elementów na stronie.

8.16 Implementacja profilu użytkownika i historii oglądania

8.16.1 Aktualny pakiet

Użytkownik po wejściu na stronę profilu odpytuje serwer */api/v1/profile/package*, który zwraca aktualnie używany pakiet. Tak odebrany pakiet jest wykorzystywany do wyświetlenia informacji o akutalnej subskrybcji.

```
{
  "data": {
    "id": 20,
    "user_id": 4,
    "tier": "administrator",
    "valid_from": "2023-05-29T00:00:00Z",
    "valid_until": "2023-06-28T00:00:00Z"
  }
}
```

Kod 34: Odpowiedź z pakietem użytkownika z serwera

8.16.2 Kupowanie subskrybcji

Jeśli użytkownik aktualnie nie posiada pakietu - może dokonać subskrybcji, wybierając odpowiedni przycisk, wywołanie tej operacji wykonuje zapytanie POST do punktu serwera */api/v1/profile/package/pay*, który zwraca link do strony STRIPE pozwalający na wykonanie bezpiecznej płatności. Użytkownik po wykonaniu płatności wraca na stronę aplikacji, a serwer dostaje stosowną informację o poprawnej płatności.

```
{ "url": "https://checkout.stripe.com/c/pay/(...)" }
```

Kod 35: Dane zwracane przez serwer

Po stronie serwera wykorzystana została biblioteka udostepniona przez zespół deweloperski STRIPE-a. Funkcja przyjmuje ilość produktu, który ma zostać zakupiony. Sam produkt musi zostać zadeklarowany w panelu STRIPE, do którego trzeba się zalogować, aby uzyskać odpowiednie klucze API, pozwalające na wykonywanie transakcji.

```
(...)
params := &stripe.CheckoutSessionParams{
  LineItems: []*stripe.CheckoutSessionLineItemParams{
    {
      Price:    stripe.String(productID),
      Quantity: stripe.Int64(1),
    },
  },
  Mode:          stripe.String("payment"),
  SuccessURL:    stripe.String(frontURL + "/payment?status=success"),
  CancelURL:     stripe.String(frontURL + "/payment?status=canceled"),
  CustomerEmail: &user.Email,
}
(...)
return c.Status(fiber.StatusOK).JSON(fiber.Map{
  "url": s.URL,
})
```

Kod 36: Implementacja punktu generującego link do formularza płatności

Wartości SuccessURL i CancelURL określają strony, do których użytkownik ma zostać przkierowany w wypadku udanej i nieudanej płatności.

8.16.3 Historia oglądania

Historia oglądania odczytywana jest z punktu zbierającego metryki użytkownika */api/v1/metrics/user*, dane odbierane z tego punktu prezentują się w ten sposób:

```
[
   {
       "id": 1,
       "user_id": 1,
       "video_id": 1,
       "time_spent_watching": 870,
       "stopped_at": 14,
       "created_at": {
           "Time": "2023-06-11T15:20:09.563112+02:00",
           "Valid": true
       },
       "updated_at": {
           "Time": "2023-06-11T15:38:47.476409+02:00",
           "Valid": true
       },
       "name": "Java Fundamentals",
       "description": "[...]",
       "IsPremium": false,
       "thumbnail_url": [...]/thumbnails/ci2rondjtb6obc4j8mkgpobrany_plik.jpg"
   }
]
```

Z ważniejszych pól można wyróżnić:

- **time_spent_watching** - czas w sekundach, który użytkownik spędził na oglądanie
- **stopped_at** - czas, na którym użytkownik zatrzymał materiał, może go od tego momentu kontynuować
- **isPremium** - informacja czy materiał jest Premium

Materiały następnie umieszczane są w obiekcie stanowym Reacta z zaznaczeniem odpowiedniego interfejsu:

```
   interface MetricData {
     video_id: number;
     name: string;
     description: string;
     IsPremium: boolean;
     thumbnail_url: string;
     updated_at: string[];
   }
   ...
   const [metric, setMetric] = useState<MetricData[]>([]);
   ...
   const { request } = profileService.GetUserVideoMetric(data);
   request
     .then((res) => {
       const Data = ...;
       setMetric(Data);
```

```
      })
    ...
```

8.16.4 Resetowanie hasła

Resetowanie hasła na profilu odbywa się na podobnej zasadzie co w wypadku resetowania z poziomu ekranu logowania. Wykorzystywany jest jednak inny punkt końcowy, a sam użytkownik nie musi potwierdzać operacji kodem odbieranym na adres e-mail. Używany punkt końcowy to */api/v1/profile/new (POST)*. Dane odbierane przez serwer to:

```
{
    "email": "usermail@test.com",
    "NewPassword": "nowehaslo"
}
```

8.17 Administracja materiałami

Wszystkie zapytania w tej sekcji zabezpieczone są po stronie serwera, więc ich wykonanie wymaga posiadania roli **administrator**. Żeby uzyskać tą rolę, użytkownik musi posiadać odpowiedni pakiet, podobnie jak w wypadku użytkowników premium. Po stronie aplikacji klienckiej istnieje flaga sprawdzająca rolę użytkownika. Nawet jeśli użytkownik ręcznie zmieni wartość w localstorage, to w dalszym ciągu nie uzyska dostępu do punktu końcowego po stronie serwera.

8.17.1 Proces dodawania kategorii

Jest to pierwszy etap, który jest wymagany, aby dodać materiał. Dodanie kategorii sprowadza się do wybrania opcji w menu i wysłania pliku w formacie JSON na punkt końcowy **/api/v1/video/categories (POST)**

```
{
  "name": "Nowa kategoria"
}
```

Kod 37: Informacje przesyłane w celu dodania kategorii

Żeby odczytać dostępne w bazie kategorie można odpytać punkt **/api/v1/categories**, który zwróci plik JSON z kategoriami oraz ich id:

```
[
  {
    "ID": 1,
    "Name": "test"
  },
  {
    "ID": 2,
    "Name": "test2"
  }
]
```

Kod 38: Informacje przesyłane w celu dodania kategorii

8.17.2 Proces dodawania materiałów na stronie

Proces dodawania materiału na stronie jest dość skomplikowany bo zakłada wykorzystanie punktów na serwerze S3 oraz serwerze aplikacji oraz wzajemnego połączenia pomiędzy tymi serwerami.

Na początku użytkownik musi zostać zweryfikowany wysyłając swój token do serwera Minio S3. Wykorzystana do tego została bilbioteka AWS, która jest również kompatybilna z wiaderkiem aplikacji Minio. Z odpowiedzi w formie XML wyłuskane muszą zostać sekretny klucz, klucz sesji oraz token sesji, które potrzebne są w kolejnym kroku.

```
import { S3Client, S3 } from '@aws-sdk/client-s3';
...
const handleSubmit = async (event: any) => {
    try {
      const res = await axios({
        method: 'post',
        url: s3endpoint,
        params: {
          Action: 'AssumeRoleWithCustomToken',
          Token: user?.authToken,
          Version: '2011-06-15',
          DurationSeconds: '3600',
          RoleArn: 'arn:minio:iam:::role/idmp-dianomi-server-auth',
        },
      });
      const parser = new DOMParser();
      const xml = parser.parseFromString(res.data, 'application/xml');
      const accessKeyId = String(xml.querySelector('AccessKeyId')?.textContent);
      const secretAccessKey = String(xml.querySelector('SecretAccessKey')?.
    textContent);
      const sessionToken = String(xml.querySelector('SessionToken')?.textContent);

      if (!accessKeyId || !secretAccessKey || !sessionToken) {
        console.error('Access credentials empty');
        return;
      }
    ...
    }
}
```

Kod 39: Autoryzacja na serwerze S3

Następującym etapem jest wysłanie pliku na serwer przy pomocy biblioteki. W podobny sposób wysyłany jest również obrazek okraszający materiał.

```
      const s3config = {
        region: 'us-east-1',
        endpoint: 'http://localhost:9000',
        credentials: {
          accessKeyId: accessKeyId,
          secretAccessKey: secretAccessKey,
          sessionToken: sessionToken,
        },
        forcePathStyle: true,
```

```
    };

    const parallelUploads3 = new Upload({
      client: new S3(s3config) || new S3Client(s3config),
      queueSize: 4,
      leavePartsOnError: false,
      params: {
        ContentType: file?.type,
        Bucket: 'uploads',
        Key: file?.name,
        Body: file,
      },
    });

    parallelUploads3.on('httpUploadProgress', (progress) => {
      if (progress.total && progress.loaded) {
        const percentage = Math.round((progress.loaded / progress.total) * 100);
        setWidth(percentage);
      }
    });

    await parallelUploads3.done();
```

Kod 40: Wysłanie materiału na serwer bezpośrednio z przeglądarki

Następnie wysyłane jest zapytanie POST na adres **/api/v1/video**, które prezentuje się w ten sposób:

```
{
  "name": "My video name",
  "description": "Lorem ipsum dolorem test me",
  "file_name": "hisinsidemywalls.mp4",
  "thumbnail_name": "test.jpg",
  "file_bucket": "uploads",
  "category_id": 1,
  "tags": [
    "test",
    "anothertag",
    "killmeplease"
  ]
}
```

Kod 41: Payload dla zapytania dodającego materiał

Zawiera ono m.in. nazwę, opis, nazwę pliku na serwerze s3, wiaderko docelowe, id kategorii z sekcji wyżej oraz tagi, które są dynamiczne tworzone w bazie jeśli jeszcze nie istnieją.

Pliki następnie są konwertowane przez FFMPEG do trzech rozdzielczości - 360p, 480p i 720p, żeby zostać ponownie umieszczone na serwerze S3, ale już w innym wiaderku.

```
  // Check if video exists
  uploads := minioClient.ListObjects(ctx, uploadBucket, minio.ListObjectsOptions
    {})
  found := false
  for vid := range uploads {
    if vid.Err != nil {
```

```
      log.WithField("err", vid.Err).Error("File error occured (minio s3)")
      return
    }
    if vid.Key == data.FileName {
      found = true
      break
    }
  }
  if !found {
    log.WithFields(log.Fields{
      "file":   data.FileName,
      "bucket": uploadBucket,
    }).Error("Specified file couldn't be found in the bucket")
    return
  }

  // Download video to fs
  downloadedFilePath := fmt.Sprintf("%s/tmp/%s", storagePath, data.FileName)
  if err := minioClient.FGetObject(
    ctx,
    uploadBucket,
    data.FileName,
    downloadedFilePath,
    minio.GetObjectOptions{}); err != nil {

    log.WithField("err", err.Error()).Error("Could not download file from minio s3
    ")
    return
  }
  filesToCleanup = append(filesToCleanup, downloadedFilePath)

  // Remove video from bucket
  if err := minioClient.RemoveObject(
    ctx,
    uploadBucket,
    data.FileName,
    minio.RemoveObjectOptions{}); err != nil {

    log.WithField("err", err.Error()).Error("Target file could not be removed from
     minio s3")
    return
  }
```

Kod 42: Proces pobierania materiału wideo

```
  // Mux files
  for res, ratio := range resolutions {
    localPath := fmt.Sprintf("%s/tmp/%s_%s.mp4", storagePath, fileId, res)
    if err := ffmpeg.Input(downloadedFilePath).
      Output(
        localPath,
        ffmpeg.KwArgs{
          "s":   ratio,
          "c:v": "libx265",
        }).
```

```
      OverWriteOutput().
      // ErrorToStdOut().
      Run(); err != nil {

      log.WithField("err", err.Error()).Error("FFMPEG returned an error")
      return
    }
    filesToCleanup = append(filesToCleanup, localPath)
  }
```

Kod 43: Proces konwertowania materiału do zadeklarowanych rozdzielczości

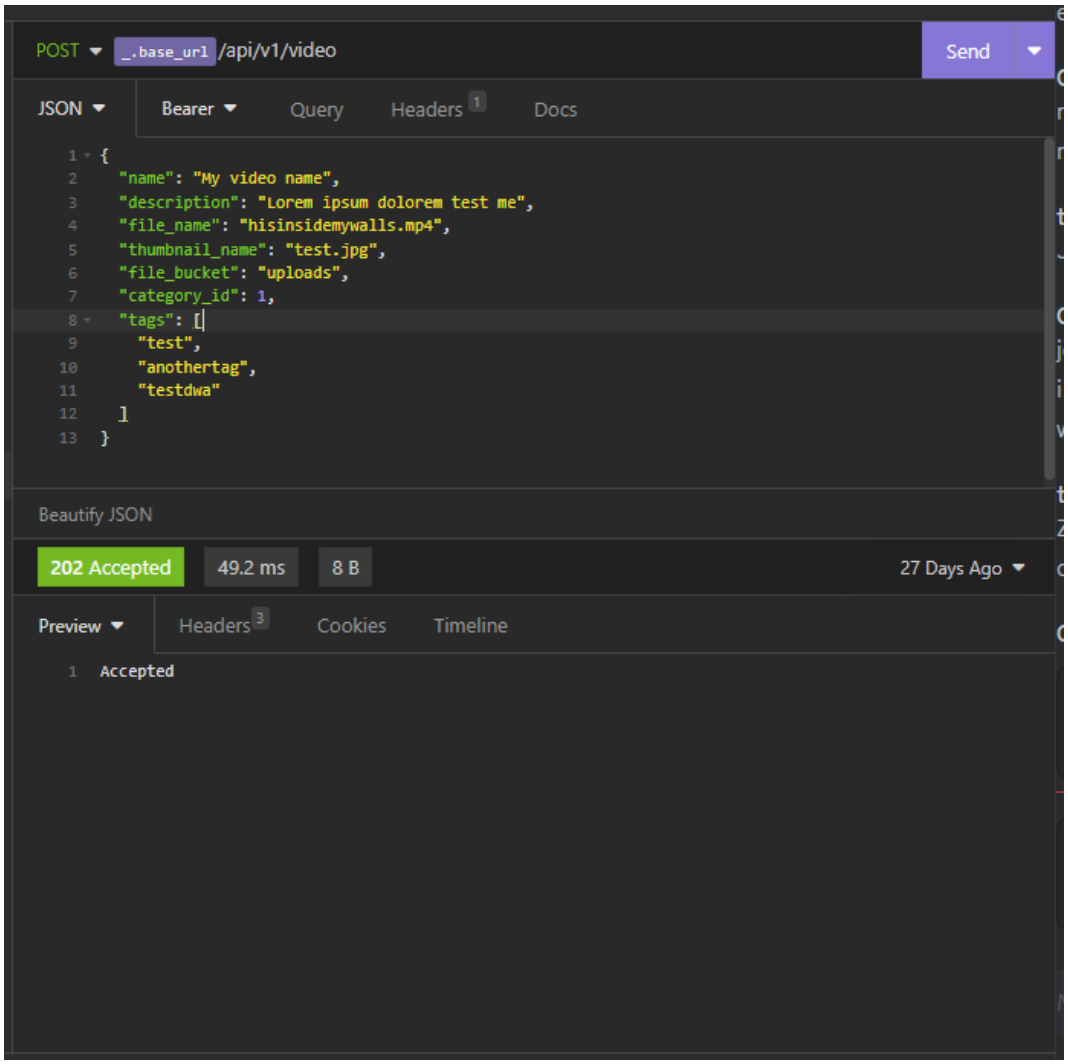
Ostatnim etapem jest dodanie materiału do bazy danych z wszystkimi potrzebnymi danymi:

```
      file, _ := os.Open(localPath)
    fileInfo, err := file.Stat()
    if err != nil {
      log.WithField("err", err.Error()).Error("Could not read file")
      return
    }

    // Get video information
    jsonData, err := ffmpeg.Probe(downloadedFilePath)
    if err != nil {
      log.WithField("err", err.Error()).Error("Could not probe video file")
      return
    }

    var videoInfo MediaInfo
    err = json.Unmarshal([]byte(jsonData), &videoInfo)
    if err != nil {
      log.WithField("err", err.Error()).Error("Could not get stream information")
      return
    }
    duration, err := strconv.ParseFloat(videoInfo.Info.Duration, 64)
    if err != nil {
      log.WithField("err", err.Error()).Error("Could not parse FFMPEG duration")
      return
    }

    // Add to database
    if err := qtx.AddVideoFile(ctx, sqlc.AddVideoFileParams{
      FilePath:   remotePath,
      VideoID:    vid.ID,
      FileSize:   fileInfo.Size(),
      Duration:   int64(duration),
      Resolution: sqlc.Resolution(res),
    }); err != nil {
      log.WithField("err", err.Error()).Error("Video file couldn't be added to
    database")
      return
    }
```

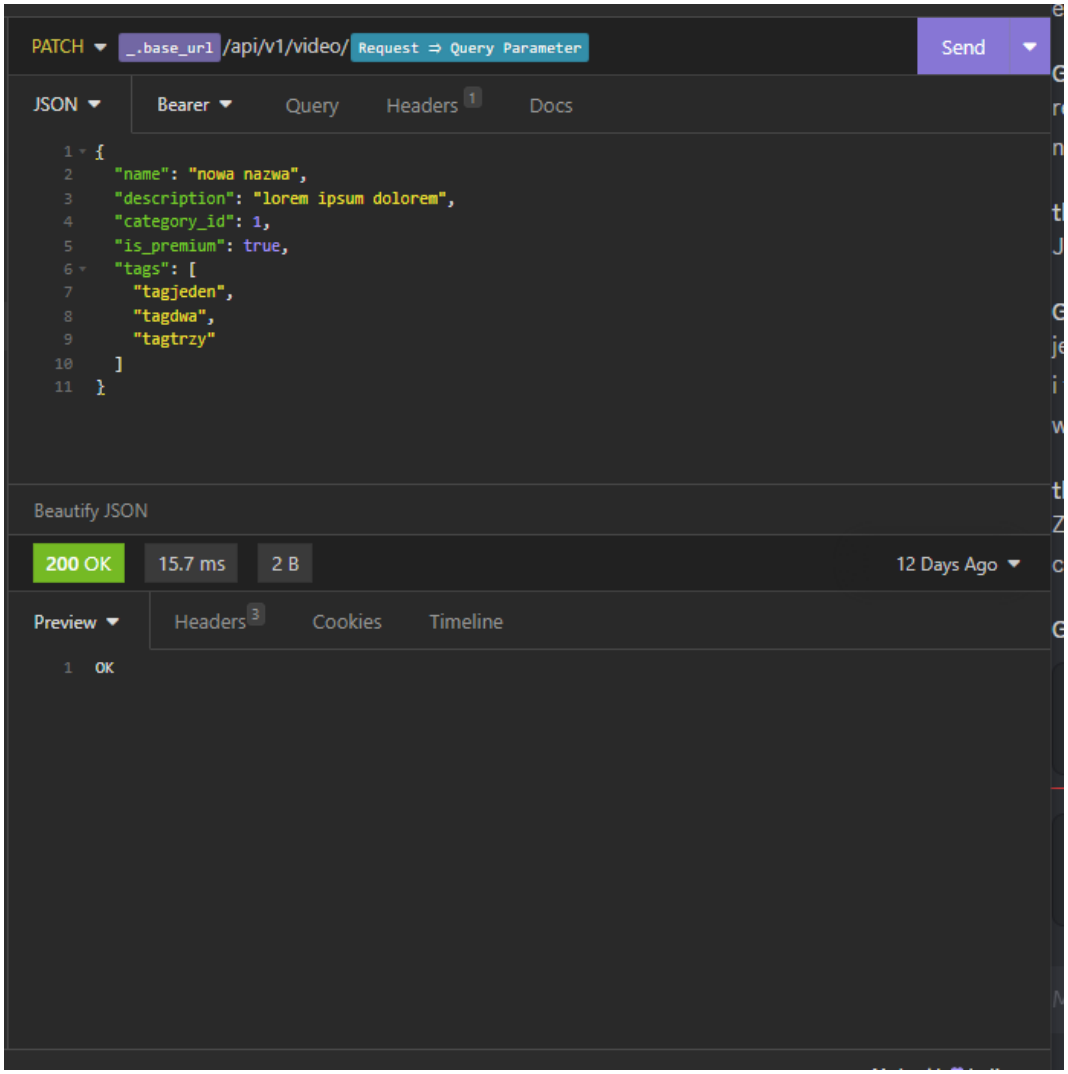
Kod 44: Proces wysyłania materiału na serwer S3

8.17.3 Proces zarządzania użytkownikami i materiałami

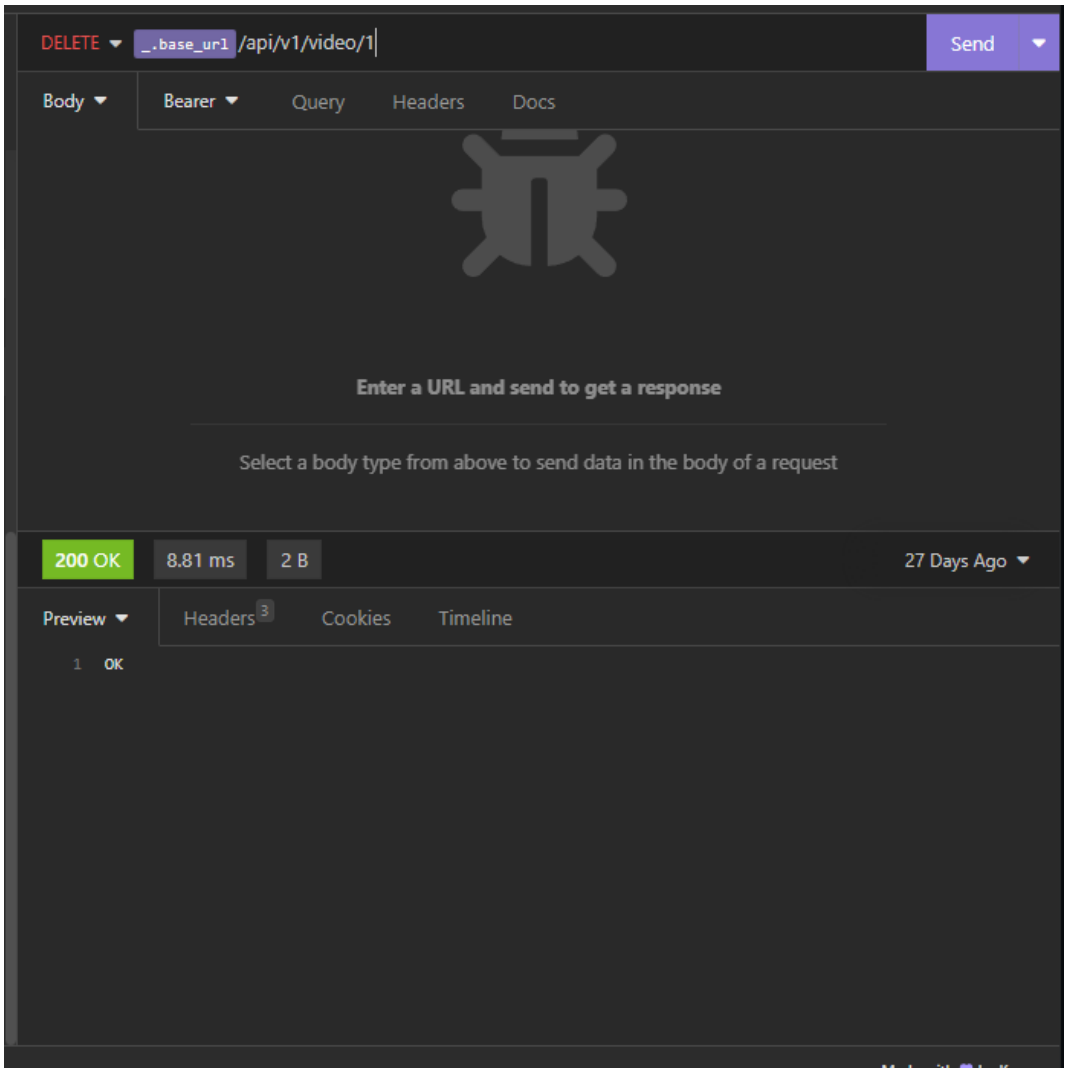
Po stronie klienta odpowiednie zapytania są zaimplementowanie w postaci usług, do których odwołują się poszczególne komponenty. Po stronie aplikacji serwerowej zostały przygotowane punkty końcowe dostępne pod adresami **/api/v1/users** i **/api/v1/video**. Dodawanie elementów sprowadza się do wysłania zapytania POST. W celu edycji zostaje wysłane zapytanie PATCH. A w celu usunięcia materiału używane jest zapytanie DELETE



Grafika 6: Dodanie materiału przy pomocy zapytania POST



Grafika 7: Edycja materiału przy pomocy zapytania PATCH



Grafika 8: Usunięcie materiału przy pomocy zapytania DELETE

9 Podsumowanie i bilans

Lista spełnionych warunków:
(✓) Działający formularz rejestracji.
(✓) Działający formularz logowania.
(✓) Możliwość resetowania hasła.
(✓) Potwierdzenie adresu e-mail.
(✓) Możliwość wyświetlania oraz odtwarzania materiałów na stronie.
(✓|X) Możliwość filtrowania materiałów na stronie.
(✓) Dodawanie komentarzy i ocen przez użytkowników pod materiałami.
(✓) Możliwość zakupu subskrybcji.
(X) Możliwość współdzielenia subskrybcji pomiędzy użytkownikami
(✓|X) Działający system rekomendacji materiałów konkretnemu użytkownikowi.
(✓) Wyświetlanie historii oglądanych materiałów przez użytkownika.
(✓|X) Przeglądanie historii płatności danego użytkownika przez Administratora.
(✓) Działający system administracji dostępnymi na witrynie elementami.
(✓) Podstawowa możliwość administrowanie kontami użytkowników przez administratora (Dodawanie, usuwanie, banowanie)
(✓) Podstawowa możliwość administrowania materiałami (Dodawanie, usuwanie, edycja materiału)
(✓|X) Podstawowa możliwość administrowania komentarzami (Dodawanie, usuwanie, zgłaszanie) oraz możliwość reagowania na te komentarze.

Bilans godzinowy projektu:

- 259/169h
- Witold Padula: 129h
- Marcel Kasprzycki: 130h

References

[1] Javascript - Wikipedia
https://pl.wikipedia.org/wiki/JavaScript

[2] PostgreSQL - Wikipedia
https://pl.wikipedia.org/wiki/PostgreSQL

[3] PostgreSQL - VAVATECH
https://vavatech.pl/technologie/bazy-danych/postgresql

[4] GoLang - Wikipedia
https://pl.wikipedia.org/wiki/Go_(j%C4%99zyk_programowania)

[5] GoLang - IT-Leaders Blog
https://blog.it-leaders.pl/golang-jezyk-programowania/

[6] Fiber
https://gofiber.io

[7] React.js
https://blog.hubspot.com/website/react-js

[8] React.js - Wikipedia
https://pl.wikipedia.org/wiki/React.js

[9] Traefik
https://doc.traefik.io/traefik/

[10] Stripe - Wikipedia
https://en.wikipedia.org/wiki/Stripe%2C_Inc.

[11] JWT - auth0
https://auth0.com/learn/json-web-tokens

[12] Docker
https://www.docker.com/

[13] Docker - Wikipedia
https://pl.wikipedia.org/wiki/Docker_(oprogramowanie)

[14] Minio S3 - Minio
https://min.io/

[15] Nginx - Wikipedia
https://pl.wikipedia.org/wiki/Nginx