**Dokumentacja techniczna projektu**

1. **Autorzy**:

* Krystian Frąckowiak 140476
* Piotr Raczkowiak 140587

1. **Tytuł projektu**:

Serwis video poświęcony tematyce sera „**CheeseHub**”.

1. **Krótki opis działania projektu**:

Naszym projektem jest serwis video poświęcony tematyce sera żartobliwie nazwany przez nas „CheeseHub”, stylistycznie nawiązujący do satyrycznego serwisu video „CornHub”. Serwis posiadać będzie funkcjonalny interfejs logowania oraz możliwość rejestracji. W skład funkcjonalności naszego serwisu będą wchodziły:

* Przeglądanie i odtwarzanie umieszczonych materiałów wideo podzielonych na różne kategorie.
* Komentowanie oraz ocenia umieszczonych materiałów.
* Zamieszczanie własnych materiałów wideo.

1. **Specyfikacja wykorzystanych technologii:**

Aplikacja składa się z dwóch modułów – backendu napisanego w technologii .NET 8 oraz frontendu napisanego w Angularze 19. Ze względu na wymagania co do sposobu oddania pracy, frontend został skompilowany i umieszczony w projekcie backendowym.

Backend – Web Api w .NET wykorzystuje architekturę MVC – do operacjach na danych wykorzystywane są modele, Działają na nich kontrolery, które zwracają dane.

Ponadto wykorzystany został w pewnym zakresie Respository Pattern, który pomógł ograniczyć liczbę zduplikowanego kodu. Wykorzystane zostało również dependency injection, co ułatwi przygotowanie testów w przyszłości. Powstał również middleware, który weryfikuje ważność JWT

1. **Instrukcje pierwszego uruchomienia projektu:**

Aby uruchomić aplikacje wystarczy pobrać pliki, rozpakować a następnie uruchomić plik CheeseHub.sln. Po uruchomieniu, dla poprawnego działania należy upewnić się że strona połączona jest za pomocą odpowiedniego portu – 44351.  
  
Uwaga: uruchamiać przy pomocy IIS

Dane dla administratora:

**Login: admin@example.pl**

**Hasło: zaq1@WSX**

1. **Opis struktury projektu:**

Projekt zawiera skompilowany kod frontendowy (Angular) w folderze wwwroot, backendu

1. **Modele:**
   * Category - odpowiada za możliwość kategoryzowania plików video, zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| Name | string | Nazwa kategorii. Pole nie może być puste i musi być unikatowe. |
| ImagePath | string | Ścieżka do obrazu repezentującego kategorie. |
| Videos | ICollection<Video> | Kolekcja obiektów typu Video przypisanych do danej kategorii. |

* + Comment - odpowiada za komentarze zamieszczone pod video. Na komentarz można zareagować oraz odpowiedzieć komentarzem, zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| VideoId | Guid | Id video pod którym komentarz został opublikowany. |
| Video | Video | video pod którym komentarz został opublikowany. |
| UserId | Guid | Id użytkownika który opublikował komentarz. |
| User | User | Użytkownik który opublikował komentarz. |
| Content | string | Zawartość komentarza. Nie może być pusty, maksymalna długość 500 znaków. |
| CreatedAt | DateTime | Data utworzenia komentarza |
| ParentCommentId | Guid | Id komentarza dla którego ten komentarz jest odpowiedzią. Może być null-em |
| ParentComment | Comment | komentarz dla którego ten komentarz jest odpowiedzią. |
| Replies | ICollection<Comment> | Odpowiedzi na komentarz |
| Reactions | ICollection<CommentReaction> | Lista reakcji na komentarz |

* + CommentReaction – reakcje na komentarze(lubię lub nie lubię), zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| UserId | Guid | Id użytkownika który dał reakcje. |
| User | User | Użytkownik który dał reakcje. |
| TargetId | Guid | Id komentarza który otrzymał reakcje. |
| Comment | Comment | Komentarz który otrzymał reakcje. |
| IsLike | bool | Typ reakcji, true = lubię, false = nie lubię. |

* + RefreshToken

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| UserId | Guid | Id użytkownika przypisanego do tokenu. |
| User | User | Użytkownika przypisany do tokenu. |
| Token | string | Wygenerowany token. |
| ExpiryDate | DateTime | Data wygaśnięcia tokenu. |
| CreatedAt | DateTime | Data utworzenia tokenu. |

* + Role – rola jaką można przypisać do użytkownika, zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid |  |
| Name | string | Nazwa roli, musi być unikatowa, maksymalna długość 150 znaków. |
| Users | ICollection<User> | Lista użytkowników przypisanych do danej roli. |

* + User -użytkownik, zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| Name | string | Nazwa pod którą użytkownik chce być widoczny. Pole wymagane, musi być unikatowe, maksymalna długość 150 znaków. |
| FirstName | string | Imię użytkownika. Pole wymagane, maksymalna długość 150 znaków. |
| Surname | string | Nazwisko użytkownika. Pole wymagane, maksymalna długość 150 znaków. |
| Email | string | Email użytkownika. Pole wymagane, musi być unikatowe, maksymalna długość 150 znaków. |
| PasswordHash | string | Hash hasła. Pole wymagane, maksymalna długość 150 znaków, minimalna 6 znaków. |
| IsTokenValid | bool | Flaga która wymusza wylogowanie. |
| Status | char | Obecnie niewykorzystywane, stworzone do przyszłego rozwoju. |
| CreatedAt | DateTime | Data utworzenia użytkownika. |
| RoleId | Guid | Id roli użytkownika. |
| Role | Role | Rola użytkownika. |
| RefreshTokens | Collection<RefreshToken> | Tokeny odświeżające przypisane do użytkowania. |
| Videos | ICollection<Video> | Lista video opublikowanych przez użytkownika. |
| VideoViews | ICollection<VideoView> | Lista video odtworzonych przez użytkownika. |
| Comments | ICollection< Comment> | Lista komentarzy opublikowanych przez użytkownika. |

* + Video – materiały zamieszone przez użytkowników. Mogą być odtwarzane, komentowane i otrzymywać reakcje.

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| Name | string | Nazwa video. Pole wymagane, maksymalna długość 150 znaków. |
| Description | string | Opis video. Maksymalna długość 4000 znaków. |
| Path | string | Ścieszka to pliku video. Maksymalna długość 150 znaków. Video może mieć maksymalny rozmiar 100MB i musi być w formacie: MKV, MP4, AVI. |
| ImagePath | string | Ścieszka do obrazu reprezentującego video. Maksymalna długość 150 znaków. Obraz może mieć maksymalny rozmiar 5MB i musi być w formacie: JPEG, PNG, WebP. |
| UserId | Guid | Id użytkownika który dodał video. |
| User | User | Użytkownik który dodał video. |
| CategoryId | Guid | Id przypisanej kategorii. |
| Category | Category | Przypisana kategoria. |
| CreatedAt | DateTime | Data dodania video. |
| Status | Status | Status video, może być zablokowane, ukryte lub dostępne. |
| Comments | ICollection<Comment> | Lista komentarzy pod video. |
| Reactions | ICollection<VideoReaction> | Lista reakcji na video. |
| Views | ICollection< VideoView> | Historia odtwarzania video. |

* + VideoReaction – reakcje na video(lubię lub nie lubię), zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| UserId | Guid | Id użytkownika który dał reakcje. |
| User | User | użytkownik który dał reakcje. |
| TargetId | Guid | Id video które otrzymało reakcje. |
| Comment | Video | Video które otrzymało reakcje. |
| IsLike | bool | Typ reakcji, true = lubię, false = nie lubię. |

* + VideoView - odpowiada za możliwość śledzenia kiedy, przez kogo i ile razy video było odtwarzane, zawiera następujące pola:

|  |  |  |
| --- | --- | --- |
| Nazwa | Typ | Opis |
| Id | Guid | Unikalny identyfikator |
| UserId | Guid | Id użytkownika który odtworzył video. |
| User | User | użytkownik który odtworzył video. |
| Video Id | Guid | Id odtworzonego video. |
| Video | Video | Odtworzone video. |
| ViewedAt | DateTime | Data odtworzenia video. |

1. **Kontrolery:**
   * CategoryController

|  |  |  |
| --- | --- | --- |
| Nazwa | Przyjmowane parametry | Opis |
| HttpGet - Get | Guid | Zwraca całą kategorie |
| HttpGet - GetCategories | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Zwraca listę kategorii |
| HttpGet - GetImage | Guid | Zwraca obrazek kategorii |
| HttpPost - Post | CreateCategoryDTO | Tworzy kategorię – zwraca błędy lub status 201. Funkcja dostępna tylko dla administratora. |

* + CommentController

|  |  |  |
| --- | --- | --- |
| Nazwa | Przyjmowane parametry | Opis |
| HttpGet - Get | Guid | Unikalny identyfikator |
| HttpPost - Post | CreateCommentDTO | Tworzy komentarz – zwraca błędy lub status 201. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpDelete - Delete | Guid | Usuwa komentarz, zwraca 200 OK. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpPost – ToggleComment  Reaction | ToggleReactionDTO | Służy do obsługi dawania like/dislike, zwraca OK 201. Funkcja dostępna tylko dla zalogowanego użytkownika. |

* + RoleController

|  |  |  |
| --- | --- | --- |
| Nazwa | Przyjmowane parametry | Opis |
| HttpGet - Get | Guid | Służy do pobrania roli |
| HttpPost - Post | CreateRoleDTO | Tworzy rolę – zwraca błędy lub status 201. Funkcja dostępna tylko dla administratora. |

* + UserController

|  |  |  |
| --- | --- | --- |
| Nazwa | Przyjmowane parametry | Opis |
| HttpPost - Register | RegisterUserDTO model | Endpoint do rejestracji, tworzy konto użytkownika |
| HttpPost - Login | LoginUserDto model | Endpoint do logowania, zwraca token lub komunikat błędu |
| HttpPost - RefreshToken | string userId, string token | Służy do odświeżenia tokenu, gdy zbliża się jego wygaśnięcie, zwraca token |
| HttpPost - Logout | string refreshToken | Unieważnia token |
| HttpGet - Get | Guid id | Służy do pobierania usera, akcja w przyszłości będzie używana przez admina. Funkcja dostępna tylko dla administratora. |
| HttpGet - GetMyData | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Służy do pobierania własnych danych na podstawie id w JWT. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpGet –  IsUserExistsByName | string name | Służy do sprawdzenia unikalności nazwy |
| HttpGet - IsTokenValid | string id | Służy do weryfikacji ważności JWT. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpGet - sUserExistsByEmail | string email | Służy do sprawdzenia unikalności e-maila |

* + VideoController

|  |  |  |
| --- | --- | --- |
| Nazwa | Przyjmowane parametry | Opis |
| HttpGet - Get | Guid Id | Służy do pobierania pojedynczego video |
| HttpGet - GetVideos | int page , int pageSize, string search = null, string? categoryId = null, bool? my = null, bool? liked = null | Służy do pobierania filmików - posiada mechanizm paginacji, oraz szereg filtrów |
| HttpGet - GetStream | Guid Id | Służy do streamowania filmiku |
| HttpGet - GetImage | Guid Id | Służy do pobierania obrazka przypisanego do filmika |
| HttpGet - GetWatch | Guid Id | Służy do pobierania danych o filmie potrzebnych w widoku oglądania |
| HttpPost - Post | CreateVideoDTO video | Służy do uploadowania filmiku. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpPost – ToggleVideo  Reaction | ToggleReactionDTO model | Analogiczna metoda do tej w komentarzach. Sterowanie reakcjami. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpPost - Ban | ModelId model | Banowanie filmiku. Funkcja dostępna tylko dla administratora. |
| HttpPost - Unban | ModelId model | Odbanowywanie filmiku. Funkcja dostępna tylko dla administratora. |
| HttpPost - Show | ModelId mode | Ukrywanie filmiku – mozna ukryc swoj filmik. Funkcja dostępna tylko dla zalogowanego użytkownika. |
| HttpPost - Hide | ModelId mode | Odkrywanie filmiku. Funkcja dostępna tylko dla zalogowanego użytkownika. |

1. **Opis systemu użytkowników:**

Aplikacja CheeseHub posiada system autentyfikacji i autoryzacji opierający się na JWT.  
JWT (JSON Web Token) to standardowy format tokenów używany do uwierzytelniania i autoryzacji, który składa się z trzech części: nagłówka (Header) zawierającego informacje o algorytmie i typie tokenu, ładunku (Payload) przechowującego dane użytkownika, takie jak identyfikator, rola czy czas wygaśnięcia, oraz podpisu (Signature) zapewniającego integralność i autentyczność tokenu dzięki użyciu klucza. Tokeny JWT są kodowane w Base64 i przesyłane w nagłówkach HTTP, umożliwiając bezstanowe uwierzytelnianie w aplikacjach.

1. **Krótka charakterystyka najciekawszych funkcjonalności:**

Jedną z ciekawszych funkcji w naszej aplikacji jest paginacja zastosowana przy wyświetlaniu plików video. Na początku na stronie ładowanych jest do 12 filmów, dopiero gdy użytkownik zjedzie na dół strony ładowane są następne pliki. Aplikacja zapewnia możliwość komentowania video jak i również odpowiadania na komentarze. Pod każdym filmem wyświetlana jest liczba odtworzeń oraz reakcji na video. Podsumowując nasza aplikacja jest dobrą bazą do stworzenia w pełni funkcjonalnego serwisu video.