Uwagi

1: Wstęp napiszemy na końcu.

Politechnika
Śląska

PROJEKT INŻYNIERSKI

Webowy organizer notatek, wydarzeń i zadań

Zofia LORENC
Nr albumu: 300306

Kierunek: Informatyka
Specjalność: Bazy Danych i Inżynieria Systemów

PROWADZĄCY PRACĘ
Dr hab. inż. Krzysztof Simiński
KATEDRA ALGORYTMIKI I OPROGRAMOWANIA
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2024

Tytuł pracy

Webowy organizer notatek, wydarzeń i zadań

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

(2-5 slow (fraz) kluczowych, oddzielonych przecinkami)

Thesis title

Web organizer of notes, events and tasks

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Key words

(2-5 keywords, separated by commas)

Spis treści

Rozdział 1

Wstęp

[Wstęp napiszemy na końcu.]

- wprowadzenie w problem/zagadnienie
- osadzenie problemu w dziedzinie
- cel pracy
- zakres pracy
- zwięzła charakterystyka rozdziałów
- jednoznaczne określenie wkładu autora, w przypadku prac wieloosobowych – tabela z autorstwem poszczególnych elementów pracy

Rozdział 2

[Analiza tematu]

- sformułowanie problemu
- osadzenie tematu w kontekście aktualnego stanu wiedzy (*state of the art*) o poruszanym problemie
- studia literaturowe [11, 12, 10, 9] - opis znanych rozwiązań (także opisanych naukowo, jeżeli problem jest poruszany w publikacjach naukowych), algorytmów,

2.1 Opis problemu

Aplikacja w dużym stopniu oparta jest o interakcję z użytkownikiem, który w każdym momencie użytkowania ma wiele dostępnych mu działań. Wyzwaniem jest więc nie zaprojektowanie złożonej logiki programu, a zapewnienie poprawnego działania dla wszelkich danych wejściowych pobranych od użytkownika. Zabezpieczenia takie muszą pojawić się od strony interfejsu, który powinien maksymalnie ograniczać możliwość wprowadzenia niepoprawnych danych, ale i od strony backendu, gdzie weryfikować należy przychodzące zapytania HTTP. Zawartość odpowiedzi HTTP jest łatwo dostępna w inspektorze przeglądarki, dlatego już na jej poziomie nie mogą być dostępne dane wrażliwe (hasła), a nie dopiero na poziomie interfejsu.

Jedną z zalet aplikacji przeglądarkowych jest możliwość zapisywania odnośników do stron, na przykład do konkretnego pliku. Oznacza to jednak, że nawet dokładnie zaprojektowany interfejs nie pozwoli na pełną kontrolę ścieżki, którą użytkownik poruszać się będzie po programie. Zmieniając adres w przeglądarce, może on w dowolnym momencie usiłować dostać się do nieistniejących bądź nieudostępnionych mu zasobów.

Drugim problemem związanym z wykorzystaniem odnośników jest fakt, że użycie ich wiąże się z przeładowywaniem aplikacji, a co za tym idzie – z utratą przechowywanych w niej danych. Aby zachować ciągłość działania, każdorazowo konieczne są odwołania do bazy danych, a także pamięci przeglądarki: ciasteczek, pamięci lokalnej i pamięci sesji.

2.2 Istniejące rozwiązania

2.2.1 Google Calendar

Popularną aplikacją służącą do zarządzania wydarzeniami i zadaniami jest Google Calendar. Zadania można dodać do kalendarza, ustawić ich termin oraz opis. Wydarzenia pozwalają ponadto między innymi na dodanie osób uczestniczących, miejsca wydarzenia i powiadomień.

2.2.2 Google Docs, Google Drive

Aplikacja Google Docs pozwala na tworzenie różnych rodzajów plików – dokumentów tekstowych, prezentacji multimedialnych, arkuszy kalkulacyjnych – i zapisywanie ich na dysku Google Drive. Pliki i foldery na dysku można udostępniać innym użytkownikom, dodając ich pojedynczo bądź grupowo, jak i udostępniając link. Udostępnienie ma trzy możliwe poziomy dostępu: tylko przeglądanie, komentowanie oraz edytowanie. Udostępnienie folderu przyznaje dostęp do wszystkich plików i podfolderów w nim się znajdujących.

2.2.3 Produkty Microsoft Office

Analogiczne funkcjonalności zapewniają aplikacje pakietu Microsoft Office. Programy typu OneNote, Word, PowerPoint, Excel pozwalają na tworzenie i udostępnianie plików różnych typów. Kalendarz pakietu Office pozwala na tworzenie wydarzeń, dodawanie do niego innych użytkowników i planowanie na ich podstawie spotkań w aplikacji Microsoft Teams.

Rozdział 3

Wymagania i narzędzia

- wymagania funkcjonalne i niefunkcjonalne
- przypadki użycia (diagramy UML) – dla prac, w których mają zastosowanie
- opis narzędzi, metod eksperymentalnych, metod modelowania itp.
- metodyka pracy nad projektowaniem i implementacją – dla prac, w których ma to zastosowanie

3.1 Wymagania funkcjonalne

3.1.1 Użytkownicy

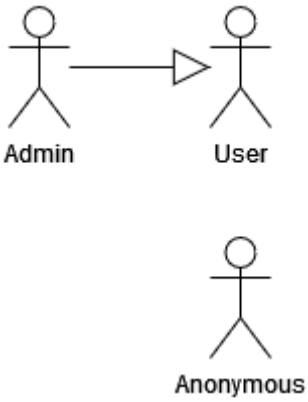
Znaczna większość funkcjonalności aplikacji wymaga, aby znany był aktualnie zalogowany użytkownik – od edycji profilu po przeglądanie plików. Ponadto istnieć powinien użytkownik specjalny (administrator), który zarządzać może pozostałymi użytkownikami. Wyróżnić można więc 3 rodzaje użytkowników:

U.1 User – użytkownik zwykły: edytuje profil, tworzy, przegląda, modyfikuje, usuwa, udostępnia elementy.

U.2 Admin – administrator: posiada dostęp do wszystkich funkcjonalności użytkownika zwykłego, zarządza użytkownikami – nadaje i odbiera uprawnienia administratorskie, usuwa użytkowników.

U.3 Anonymous – użytkownik niezalogowany: loguje lub rejestruje się.

Zależności między użytkownikami przedstawiono na rys. 3.1.



Rysunek 3.1: Użytkownicy aplikacji.

3.1.2 Funkcjonalności

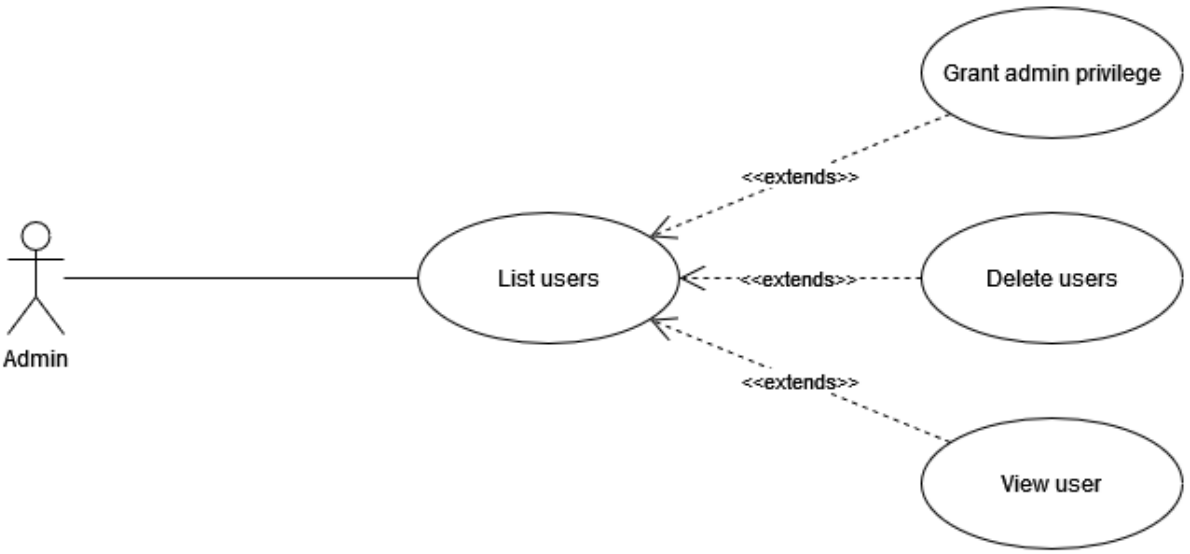
Dostępne użytkownikowi funkcje aplikacji można ogólnie podzielić na operacje związane z kontem – autoryzacją i edycją – oraz związane z plikami – przeglądanie, edycja, tworzenie nowych plików.

F.1. Autentykacja użytkowników

- F.1.1. Logowanie użytkownika U.3. na istniejące konto przy pomocy nazwy użytkownika oraz hasła.
- F.1.2. Wylogowanie zalogowanego użytkownika (U.1., U.2.).
- F.1.3. Rejestracja nowego użytkownika w systemie, umożliwiająca następnie zalogowanie (por. F.1.1.).

F.2. Zarządzanie elementami.

- F.2.1. Tworzenie nowego elementu.
- F.2.2. Wyświetlenie elementu. Dla należącego do innego użytkownika - zależne od przydzielonego dostępu (por. F.2.5.).
- F.2.3. Edycja elementu.
  - F.2.3.1. Przeniesienie do innego katalogu.
  - F.2.3.2. Zmiana nazwy.
  - F.2.3.3. Mofyfikacja zawartości. Dla należącego do innego użytkownika - zależne od przydzielonego dostępu (por. F.2.5.).
  - F.2.3.4. Modyfikacja opcji powiadomień.
  - F.2.3.5. Oddawanie głosu na termin wydarzenia.
- F.2.4. Usunięcie elementu.
- F.2.5. Przydział dostępu innym użytkownikom do elementów.



Rysunek 3.2: Diagram przypadków użycia aplikacji dla administratora.

F.3. Zarządzanie kontem użytkownika.

F.3.1. Edycja danych.

F.3.2. Usunięcie konta.

F.4. Zarządzanie użytkownikami przez administratora (U.2.).

F.4.1. Wyświetlenie wszystkich użytkowników systemu.

F.4.2. Wyświetlenie jednego użytkownika.

F.4.3. Nadanie bądź odebranie użytkownikowi uprawnień administratora.

F.4.4. Usunięcie użytkownika.

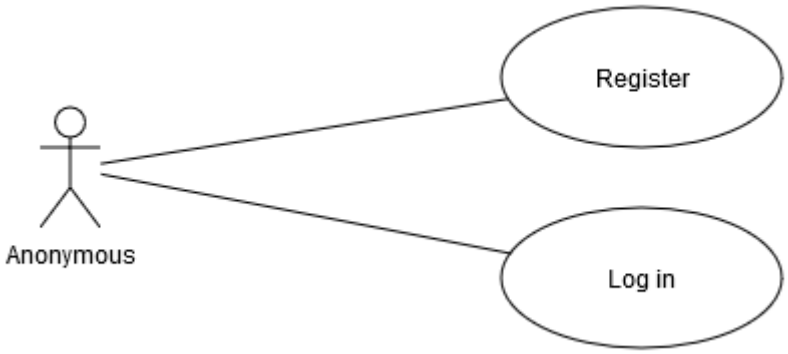
F.5. Odporność na nieprawidłowe dane wejściowe.

Wymagania funkcjonalne dla poszczególnych użytkowników przedstawiono na rys. 3.2, 3.3, 3.4.

3.2 Wymagania niefunkcjonalne

Dla aplikacji wymagającej zalogowania przez użytkownika, służącej do przechowywania prywatnych informacji niezbędne jest zapewnienie bezpieczeństwa danych przed potencjalnym przechwyceniem. Hasła użytkowników muszą być przechowywane w formie zakodowanej, do minimum należy też ograniczyć przesyłanie ich do front-endu – potrzebne są tylko do logowania, rejestracji oraz zmiany hasła.

Rysunek 3.3: Diagram przypadków użycia aplikacji dla zwykłego, zalogowanego użytkownika.



Rysunek 3.4: Diagram przypadków użycia aplikacji dla użytkownika niezalogowanego.

Istotna również jest intuicyjność interfejsu użytkownika. Elementy interfejsu i ich zachowania powinny być podobne do takich, jakie można znaleźć w innych podobnych aplikacjach. Ponadto przyciski i formularze powinny zawierać informację, czego dotyczą, jakie operacje wykonują. Pomocne może być też zastosowanie ikon wraz z opisem tekstowym.

3.3 Narzędzia

Do implementacji back-endu wybrano framework Spring [15], pozwalający m.in. na pisanie aplikacji z wykorzystaniem Java Persistence API (Spring Data JPA) oraz proste zaimplementowanie autentykacji i autoryzacji użytkowników (Spring Security). Za wybraniem tego frameworku zamiast np. Entity Framework [5] przemawiała również posiadana już wiedza o REST API oraz chęć pogłębienia znajomości samego Springa. Wspieraną przez Spring bazę danych MySQL [8] oraz sam back-end postanowiono uruchomić w kontenerze Dockera [4].

Do testowania back-endu niezależnie od front-endu wykorzystano aplikację Postman [13]. Pozwala ona na bezpośrednie wysyłanie zapytań HTTP m.in. metodami GET, POST, PUT, DELETE z ciałem w wybranym formacie, a także na przykład na przeglądanie ciasteczek.

Do wykonania front-endu posłużył framework Angular [1, 3]. Podobnie jak w przypadku Springa, za tym wyborem przemawiało pragnienie pogłębienia wstępnej wiedzy o frameworku. Subiektywną zaletą jest też, w przeciwieństwie do popularniejszego [1] Reacta [14], wyraźny rozdział na pliki HTML, CSS (SCSS) oraz TypeScript w ramach danego komponentu. Wykorzystano również komponenty Angular Material [2], dzięki którym w prosty sposób uzyskać można estetycznie i spójnie wyglądający interfejs graficzny, zgodny z wytycznymi Material Design firmy Google [7]. Wykorzystanie gotowych komponentów pozwala na skupienie się na samej logice programu. Do operacji matematycznych na datach użyto biblioteki Luxon [6].

[1] `https://trends.stackoverflow.co/?tags=reactjs,vue.js,angular,svelte,angularjs,vuejs3`, dostęp 08.11.2024

Rozdział 4

Specyfikacja zewnętrzna

Jeśli „Specyfikacja zewnętrzna”:

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)

4.1 Uruchamianie

Do uruchomienia programu potrzebne są: JDK wersja 17, Docker (w systemie Linux zainstalować można sam silnik Docker, w systemie Windows potrzebna jest aplikacja Docker Desktop) oraz Node.js, wraz z NPM (wykorzystano wersję 20). Program wykorzystywać będzie porty: 3306 (baza danych), 8080 (back-end) i 4200 (front-end) – należy więc zadbać, aby żadna inna aplikacja nie korzystała z nich podczas uruchamiania.

Jeżeli spełnione zostały powyższe wymagania, można przejść do zbudowania i uruchomienia projektu. Po rozpakowaniu plików źródłowych, w wybranym terminalu należy ustawić bieżący katalog na `organizer`. Znajdować się w nim powinny pliki `mvnw` oraz `mvnw.cmd`. Najpierw należy uruchomić kontener zawierający bazę danych poleceniem

`docker compose up db`. Aby zbudować projekt należy użyć polecenia `.\mvnw.cmd install` w systemie Windows, bądź `./mvnw install` w systemie Linux. Polecenie automatycznie zainstaluje również narzędzie Maven w odpowiedniej wersji, jeśli nie znajdzie go na komputerze. Po poprawnym wykonaniu polecenia, wygenerowany zostanie katalog `target`, zawierający plik z rozszerzeniem .jar.

Następnym krokiem jest uruchomienie back-endu w kontenerze Dockera. W tym celu można najpierw zatrzymać kontener z bazą danych, a wykonać polecenia `docker compose build` i `docker compose up -d`. Zalecane jest użycie przełącznika `-d`, aby uruchomić kontener w tle.

Niektóre środowiska zintegrowane, takie jak IntelliJ IDEA, pozwalają na wykonanie powyższych poleceń (z wyjątkiem `docker compose build`) w z poziomu interfejsu graficznego.

W celu uruchomienia front-endu należy ustawić bieżący katalog na `organizer-front` i wywołać polecenie `npm install -g @angular/cli` i `npm install` aby zainstalować potrzebne pakiety. Następnie należy wykonać polecenie `ng serve --proxy-config profxy.config.json` W terminalu wypisany zostanie adres http://localhost:4200/, na który można nacisnąć lewym przyciskiem myszy, wciskając jednocześnie klawisz Ctrl, aby otworzyć go w domyślnej przeglądarce, lub też wpisać w jej polu adresu.

Rozdział 5

Specyfikacja wewnętrzna

Jeśli „Specyfikacja wewnętrzna”:

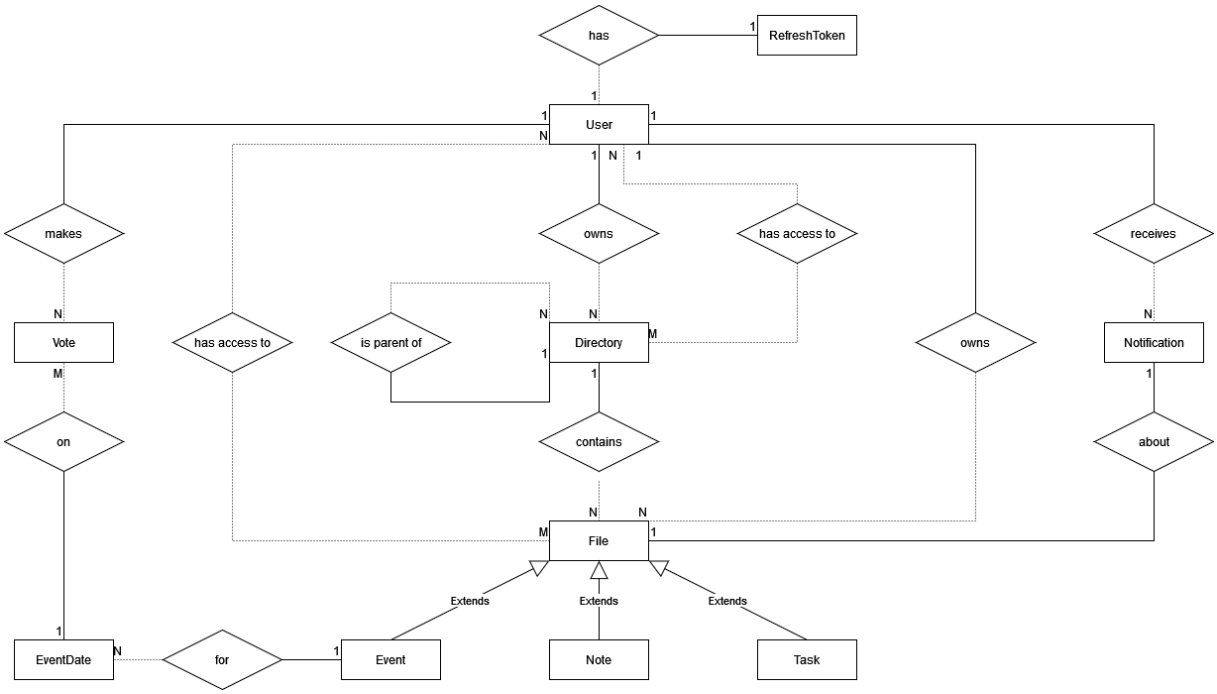
- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótka wstawka kodu w linii tekstu jest możliwa, np. `int a;` (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys **??**, a naprawdę długie fragmenty – w załączniku.

5.1 Lista klas

Zależności między klasami przedstawiono na rys. 5.1.
Tabele w bazie danych posiadają następujące atrybuty:

- `AccessDirectory(directory_id (PK, FK), user_id (PK, FK), access_privilege)`
- `AccessFile(file_id (PK, FK), user_id (PK, FK), access_privilege)`
- `Directory(id (PK), name, owner_id (FK), parent_id (FK))`
- `EventDate(id (PK), start, end, total_score, event_id (FK))`
- `File(id (PK), file_type, creation_date, name, text_content, start_date, end_date,`



Rysunek 5.1: Schemat ERD klas w bazie danych.

- `Notification(id (PK), message, send_time_setting, is_sent, is_read, file_id (FK),`
- `RefreshToken(id (PK), token, expiry_date, user_id (FK))`
- `User(id (PK), username, password, name, email, role)`
- `Vote(id (PK), score, user_id (FK), event_date_id (FK))`

Do zaimplementowania dziedziczenia klas `Event`, `Note` i `Task` po klasie bazowej wybrano strategię Singe Table Inheritance. W bazie danych wszystkie klasy reprezentowane są przez tę samą tabelę `File`, która zawiera kolumny atrybutów wszystkich dziedziczących klas, a także ukrytą kolumnę `file_type`, pozwalającą frameworkowi na rozróżnienie typów podczas mapowania obiektowo-relacyjnego. Rozwiązanie takie pozwala również na odczytanie jednocześnie wszystkich typów plików, co jest przydatne podczas wyświetlania ich w eksploratorze.

Rozwiązanie MappedSuperclass zostało odrzucone, ponieważ nie tworzy ono tabeli klasy bazowej, przez co nie może ona być w relacji z żadną inną klasą. Rozwiązania Joined Table oraz Table Per Class mogłyby wykorzystane, jednak są mniej optymalne z perspektywy założonego działania.

5.2 Lista endpointów

W tym podrozdziale przedstawiono listę wszystkich endpointów wystawianych przez backend. Dla każdego opisano URL, ogólną zasadę działania, dane wejściowe oraz możliwe

odpowiedzi.

5.2.1 Authorization

Log In `POST /api/auth/login`

Logowanie użytkownika.

Wymagane w ciele zapytania: nazwa użytkownika (username), hasło (password)

```
{
    "username": "newUser",
    "password": "password"
}
```

Odpowiedzi:

- Poprawne zalogowanie: obiekt użytkownika z pustym hasłem, kod 200 (OK)
- Niepoprawne hasło/login: „Incorrect credentials”, kod 403 (Forbidden)
- Brak hasła/loginu: „Empty username or password”, kod 400 (Bad Request)

Log Out `POST /api/auth/logout`

Wylogowanie użytkownika.

Brak wymaganego ciała zapytania.

```
{}
```

Odpowiedź: „Success”, puste ciasteczka w nagłówku, kod 200 (OK)

Register `POST /api/auth/register`

Rejestracja nowego użytkownika i założenie jego katalogu bazowego Base Directory.

Wymagana nazwa użytkownika i hasło, opcjonalny adres email oraz imię (name). Pozostałe parametry są ignorowane.

```
{
    "username": "someUser",
    "password": "password",
    "email": "some@email.com",
    "name": "Some User"
}
```

Odpowiedzi:

- Poprawne zarejestrowanie: obiekt utworzonego użytkownika z pustym hasłem i rolą ROLE_USER, kod 200 (OK)
- Próba zarejestrowania użytkownika o istniejącej już nazwie: kod 403 (Forbidden)
- Brak nazwy użytkownika lub hasła: kod 400 (Bad Request)

Change Password `PUT /api/auth/password`

Pozwala na zmianę hasła użytkownika.

Wymaga podania starego hasła, służącego do zatwierdzenia zmiany, oraz nowego hasła.

```
{
  "oldPassword": "pwdOld",
  "newPassword": "pwdNew"
}
```

Odpowiedzi:

- Poprawna zmiana hasła: „Success”, kod 200 (OK)
- Brak podanego starego lub nowego hasła: kod 400 (Bad Request)
- Użytkownik nie istnieje: kod 404 (Not Found)
- Stare hasło jest niepoprawne: kod 403 (Forbidden)

Grant/Revoke Admin Privilege `PUT /api/auth/grant` `PUT /api/auth/revoke`

Nadanie/odebranie roli Administratora użytkownikowi.

Wymagana nazwa użytkownika. Endpoint dostępny jest tylko administratorowi.

```
{
    "username": "newUser"
}
```

Odpowiedzi:

- Poprawne nadanie/odebranie roli: obiekt użytkownika, kod 200 (OK)
- Nieistniejąca nazwa użytkownika: kod 404 (Not Found)
- Próba nadania/odebrania sobie roli: kod 400 (Bad Request)
- Użytkownik nie jest administratorem: kod 403 (Forbidden)

Refresh Token `POST /api/refreshtoken`

Odświeża JWT odpowiadający za autoryzację użytkownika, jeżeli refreshToken zapisany w ciasteczkach nie stracił ważności.

Ciało zapytania jest ignorowane.

Odpowiedzi:

- Poprawne odświeżenie JWT: nowe ciasteczko JWT w nagłówku, kod 200 (OK)
- refreshToken stracił ważność: kod 403 (Forbidden)
- Nie znaleziono refreshToken w bazie danych: kod 404 (Not Found)
- refreshToken w ciasteczku jest pusty lub null: kod 400 (Bad Request)

5.2.2 Users

Get All Users `GET /api/users`

Zwraca wszystkich użytkowników z pustymi hasłami. Endpoint dostępny tylko administratorowi.

Odpowiedzi:

- Użytkownik jest administratorem: lista obiektów użytkowników, kod 200 (OK)
- Użytkownik nie jest administratorem: kod 403 (Forbidden)

Get All Users Safe `GET /api/users/safe`

Zwraca ID, nazwy użytkownika i nazwy wszystkich użytkowników.

Odpowiedź: lista obiektów użytkowników, kod 200 (OK)

Get User By ID `GET /api/users/{id}`

Zwraca użytkownika o podanym ID z pustym hasłem. Endpoint dostępny tylko administratorowi.

Odpowiedzi:

- Użytkownik istnieje: obiekt użytkownika, kod 200 (OK)
- Użytkownik nie istnieje: kod 404 (Not Found)
- Brak ID: kod 400 (Bad Request)
- Użytkownik nie jest administratorem: kod 403 (Forbidden)

Get User By ID Safe `GET /api/users/safe/{id}`

Zwraca ID, nazwę użytkownika oraz nazwę dla użytkownika o podanym ID z pustym hasłem.

Odpowiedzi:

- Użytkownik istnieje: obiekt użytkownika, kod 200 (OK)
- Użytkownik nie istnieje: kod 404 (Not Found)
- Brak ID: kod 400 (Bad Request)

Update User `PUT /api/users`

Aktualizuje użytkownika.

Wymagane podanie ID użytkownika w ciele zapytania. Możliwa jest tylko aktualizacja własnego użytkownika.

```
{
    "id": 5,
    "name": "New User 2"
}
```

Parametry brane pod uwagę:

- username
- name
- email

Odpowiedzi:

- Pomyślna aktualizacja: obiekt użytkownika z pustym hasłem, kod 200 (OK)
- Nie znaleziono ID: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)
- Próba zmiany nazwy użytkownika na już istniejącą lub pustą/zmiany użytkownika innego niż aktualnie zalogowany: kod 403 (Forbidden)

Delete User `DELETE /api/users/{id}`

Usuwa użytkownika o podanym ID, wraz z jego katalogami, głosami i powiadomieniami. Endpoint dostępny tylko administratorowi.

Odpowiedzi:

- ID istnieje: kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Próba usunięcia własnego użytkownika/użytkownik nie jest administratorem: kod 403 (Forbidden)

Delete My User `DELETE /api/users/delete`

Wylogowuje i usuwa aktualnie zalogowanego użytkownika, wraz z jego katalogami, głosami i powiadomieniami.

Odpowiedzi:

- Poprawne usunięcie: puste ciasteczka w nagłówku, kod 200 (OK)
- Nie znaleziono użytkownika: kod 404 (Not Found)

5.2.3 Directories

Get My Base Directory `GET /api/directories/basedirs`

Zwraca katalog bazowy aktualnie zalogowanego użytkownika.

Odpowiedź: obiekt katalogu bazowego, kod 200 (OK)

Get Directories By Parent ID `GET /api/directories/subdirs/{id}`

Zwraca katalogi podrzędne katalogu o podanym ID.

Odpowiedzi:

- ID istnieje: lista obiektów katalogów, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)
- Brak dostępu: kod 403 (Forbidden)

Get Directory By ID `GET /api/directories/{id}`

Zwraca katalog o podanym ID.

Odpowiedzi:

- ID istnieje: obiekt katalogu, kod 200 (OK)

- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)
- Brak dostępu: kod 403 (Forbidden)

Check Directory Edit Access `GET /api/directories/check/{id}`

Sprawdza, czy aktualnie zalogowany użytkownik posiada prawa do edycji katalogu o danym ID.

Odpowiedzi:

- ID istnieje: true jeżeli użytkownik może edytować katalog, false jeżeli nie, kod 200 (OK)
- Nie podano ID: kod 400 (Bad Request)
- ID nie istnieje: kod 404 (Not Found)

Create Directory `POST /api/directories`

Tworzy nowy katalog.

Wymagane ID rodzica, można podać nazwę. W przypadku niepodania nazwy, domyślnie ustawiana jest ona na „Unnamed Directory”.

```
{
    "name": "Test Directory",
    "parent": 3
}
```

Odpowiedzi:

- Poprawne utworzenie katalogu: obiekt nowego katalogu, kod 200 (OK)
- Brak ID rodzica: kod 400 (Bad Request)
- Nie znaleziono ID rodzica: kod 404 (Not Found)
- Katalog rodzic należy do innego użytkownika: kod 403 (Forbidden)

Update Directory `PUT /api/directories`

Aktualizuje katalog.

Wymagane podanie ID katalogu w ciele zapytania.

```
{
    "id": 2,
    "name": "Some Directory"
}
```

Parametry brane pod uwagę:

- name

Odpowiedzi:

- Pomyślna aktualizacja: obiekt katalogu, kod 200 (OK)
- Brak obiektu/próba zmiany katalogu nadrzędnego na należący do innego użytkownika: kod 400 (Bad Request)
- Nie znaleziono ID: kod 404 (Not Found)
- Brak dostępu do edycji: kod 403 (Forbidden)

Delete Directory `DELETE /api/directories/\{id\}`

Usuwa katalog o podanym ID, wraz z jego plikami.
Odpowiedzi:

- ID istnieje: kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)

5.2.4 Files

Get Files In Directory `GET /api/files/dir/{id}`

Zwraca wszystkie pliki w katalogu o podanym ID.
Odpowiedzi:

- ID istnieje: lista obiektów katalogów, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Brak dostępu: kod 403 (Forbidden)

Get File By ID `GET /api/files/{id}`

Zwraca katalog o podanym ID.
Odpowiedzi:

- ID istnieje: obiekt katalogu, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)
- Brak dostępu: kod 403 (Forbidden)

Check File Edit Access `GET /api/files/check/{id}`

Sprawdza, czy aktualnie zalogowany użytkownik posiada prawa do edycji pliku o danym ID.

Odpowiedzi:

- ID istnieje: true jeżeli użytkownik może edytować plik, false jeżeli nie, kod 200 (OK)
- Nie podano ID: kod 400 (Bad Request)
- ID nie istnieje: kod 404 (Not Found)

Create Event `POST /api/files/event`

Tworzy nowe wydarzenie.

Wymagane jest podanie ID katalogu rodzica. Domyślną nazwą jest „Unnamed Event”.

```
{
    "name": "My First Event",
    "parent": 3,
    "textContent": "Hello World! This is my first event",
    "location": "Katowice"
}
```

Odpowiedzi:

- Pomyślne utworzenie: obiekt nowego wydarzenia, kod 200 (OK)
- Nie podano ID rodzica: kod 400 (Bad Request)
- Nie znaleziono ID rodzica: kod 404 (Not Found)
- Katalog rodzic należy do innego użytkownika: kod 403 (Forbidden)

Update Event `PUT /api/files/event`

Aktualizuje wydarzenie.

Wymagane podanie ID wydarzenia w ciele zapytania.

```
{
    "id": 2,
    "location": "Gliwice"
}
```

Parametry brane pod uwagę:

- name

- textContent
- startDate
- endDate
- location

Odpowiedzi:

- Pomyślna aktualizacja: obiekt wydarzenia, kod 200 (OK)
- Brak obiektu/próba zmiany katalogu nadrzędnego na należący do innego użytkownika: kod 400 (Bad Request)
- Nie znaleziono ID: kod 404 (Not Found)
- Brak dostępu do edycji: kod 403 (Forbidden)

Create Note `POST /api/files/note`

Tworzy nową notatkę.

Wymagane jest podanie ID katalogu rodzica. Domyślną nazwą jest „Unnamed Note”.

```
{
  "name": "My First Note",
  "parent": 3,
  "textContent": "Hello World! This is my first note"
}
```

Odpowiedzi:

- Pomyślne utworzenie: obiekt nowej notatki, kod 200 (OK)
- Nie podano ID rodzica: kod 400 (Bad Request)
- Nie znaleziono ID rodzica: kod 404 (Not Found)
- Katalog rodzic należy do innego użytkownika: kod 403 (Forbidden)

Update Note `PUT /api/files/note`

Aktualizuje notatkę.

Wymagane podanie ID notatki w ciele zapytania.

```
{
  "id": 1,
  "name": "My Note"
}
```

Parametry brane pod uwagę:

- name
- textContent

Odpowiedzi:

- Pomyślna aktualizacja: obiekt notatki, kod 200 (OK)
- Brak obiektu/próba zmiany katalogu nadrzędnego na należący do innego użytkownika: kod 400 (Bad Request)
- Nie znaleziono ID: kod 404 (Not Found)
- Brak dostępu do edycji: kod 403 (Forbidden)

Create Task `POST /api/files/task`

Tworzy nowe zadanie.
Wymagane jest podanie ID katalogu rodzica. Domyślną nazwą jest „Unnamed Task”.

```
{
  "name": "My First Task",
  "parent": 2,
  "textContent": "Hello World! This is my first task",
  "isFinished": false
}
```

Odpowiedzi:

- Pomyślne utworzenie: obiekt nowego zadania, kod 200 (OK)
- Nie podano ID rodzica: kod 400 (Bad Request)
- Nie znaleziono ID rodzica: kod 404 (Not Found)
- Katalog rodzic należy do innego użytkownika: kod 403 (Forbidden)

Update Task `PUT /api/files/task`

Aktualizuje zadanie.
Wymagane podanie ID zadania w ciele zapytania.

```
{
  "id": 3,
  "isFinished": true
}
```

Parametry brane pod uwagę:

- name
- textContent
- isFinished
- deadline

Odpowiedzi:

- Pomyślna aktualizacja: obiekt zadania, kod 200 (OK)
- Brak obiektu/próba zmiany katalogu nadrzędnego na należący do innego użytkownika: kod 400 (Bad Request)
- Nie znaleziono ID: kod 404 (Not Found)
- Brak dostępu do edycji: kod 403 (Forbidden)

Delete File `DELETE /api/files/{id}`

Usuwa plik o podanym ID, w przypadku wydarzenia - wraz z jego obiektami EventDate.

Odpowiedzi:

- ID istnieje: kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)

5.2.5 Access Directory

Get AccessDirectory By User `GET /api/ad/user/{user}`

Zwraca listę obiektów AccessDirectory dla podanego ID użytkownika.
Lista jest pusta dla nieistniejącego użytkownika.
Odpowiedzi:

- ID istnieje: lista obiektów AccessDirectory, kod 200 (OK)
- Brak ID: kod 400 (Bad Request)

Get AccessDirectory By Directory `GET /api/ad/dir/{dir}`

Zwraca listę obiektów AccessDirectory dla podanego ID katalogu.
Lista jest pusta dla nieistniejącego katalogu.
Odpowiedzi:

- ID istnieje: lista obiektów AccessDirectory, kod 200 (OK)
- Brak ID: kod 400 (Bad Request)

Modify AccessDirectory `POST /api/ad`

Tworzy lub aktualizuje obiekt AccessDirectory o podanych ID użytkownika i katalogu. Wymagane podanie obydwu ID w ciele zapytania.

```
{
    "id": {
        "userId": 4,
        "directoryId": 3
    },
    "accessPrivilege": 1
}
```

Odpowiedzi:

- Pomyślne utworzenie/aktualizacja: obiekt AccessDirectory, kod 200 (OK)
- Nie istnieje któreś z ID: kod 404 (Not Found)
- Brak ID: kod 400 (Bad Request)

Delete AccessDirectory `DELETE /api/ad/{user}/{dir}`

Usuwa obiekt AccessDirectory o podanych ID użytkownika i katalogu.
Odpowiedzi:

- ID istnieją: kod 200 (OK)
- ID nie istnieją: kod 404 (Not Found)

5.2.6 Access File

Get AccessFile By User `GET /api/af/user/{user}`

Zwraca listę obiektów AccessFile dla podanego ID użytkownika.
Lista jest pusta dla nieistniejącego użytkownika.
Odpowiedzi:

- ID istnieje: lista obiektów AccessFile, kod 200 (OK)
- Brak ID: kod 400 (Bad Request)

Get AccessFile By File `GET /api/af/file/{file}`

Zwraca listę obiektów AccessFile dla podanego ID pliku.
Lista jest pusta dla nieistniejącego pliku.
Odpowiedzi:

- ID istnieje: lista obiektów AccessFile, kod 200 (OK)
- Brak ID: kod 400 (Bad Request)

Modify AccessFile `POST /api/af`

Tworzy lub aktualizuje obiekt AccessFile o podanych ID użytkownika i pliku. Wymagane podanie obydwu ID w ciele zapytania.

```
{
    "id": {
        "userId": 4,
        "fileId": 3
    },
    "accessPrivilege": 1
}
```

Odpowiedzi:

- Pomyślne utworzenie/aktualizacja: obiekt AccessFile, kod 200 (OK)
- Nie istnieje któreś z ID: kod 404 (Not Found)
- Brak ID: kod 400 (Bad Request)

Delete AccessFile `DELETE /api/af/{user}/{file}`

Usuwa obiekt AccessFile o podanych ID użytkownika i pliku.
Odpowiedzi:

- ID istnieją: kod 200 (OK)
- ID nie istnieją: kod 404 (Not Found)

5.2.7 Event Dates

Get All EventDates `GET /api/ed`

Zwraca wszystkie obiekty EventDate.
Odpowiedź: lista obiektów EventDate, kod 200 (OK)

Get EventDates By Event ID `GET /api/ed?id={eventId}`

Zwraca obiekty EventDate dotyczące wydarzenia o podanym ID.
Odpowiedzi:

- ID istnieje: lista obiektów EventDate, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)

Get EventDate By ID `GET /api/ed/{id}`

Zwraca obiekt EventDate o podanym ID.

Odpowiedzi:

- ID istnieje: obiekt EventDate, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)

Create EventDate `POST /api/ed`

Tworzy obiekt EventDate.

Wymagane jest podanie ID wydarzenia, a także początku i końca terminu. Wynik całkowity ustawiany jest na 0. Wydarzenie musi istnieć i należeć do tworzącego obiekt użytkownika.

```
{
    "event": 2,
    "start" : "2024-10-18T12:00:00",
    "end": "2024-10-18T12:30:00"
}
```

Odpowiedzi:

- Pomyślne utworzenie: nowy obiekt EventDate, kod 200 (OK)
- Brak ID/terminu początkowego/końcowego: kod 400 (Bad Request)
- Wydarzenie nie istnieje: kod 404 (Not Found)
- Wydarzenie należy do innego użytkownika: kod 403 (Forbidden)

Delete EventDate `DELETE /api/ed/{id}`

Usuwa obiekt EventDate o podanym ID, wraz z jego głosami. Wydarzenie należeć musi do usuwającego obiekt użytkownika.

Odpowiedzi:

- ID istnieje: kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Wydarzenie należy do innego użytkownika: kod 403 (Forbidden)

5.2.8 Votes

Get Votes By EventDate ID `GET /api/votes/ed/{id}`

Zwraca głosy na termin EventDate o podanym ID.

Odpowiedzi:

- ID istnieje: lista obiektów głosów, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)

Get Current User's Vote By EventDate ID `GET /api/votes/myvote/{id}`

Zwraca listę głosów oddanych przez aktualnie zalogowanego użytkownika na termin EventDate o podanym ID.

Odpowiedzi:

- ID istnieje: lista obiektów głosów, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)

Cast Vote `POST /api/votes`

Sprawdza, czy zalogowany użytkownik oddał głos na dany termin, jeżeli nie, to tworzy nowy głos, jeżeli tak, to aktualizuje istniejący. Modyfikuje wynik całkowity dla terminu EventDate.

Wymagane podanie ID EventDate.

```
{
    "eventDate": 1,
    "score": 1
}
```

Odpowiedzi:

- Pomyślne oddanie/modyfikacja głosu: obiekt głosu, kod 200 (OK)
- Nie podano ID EventDate: kod 400 (Bad Request)
- Nie istnieje ID EventDate: kod 404 (Not Found)

Delete Vote `DELETE /api/vote/{id}`

Usuwa głos o podanym ID.

Odpowiedzi:

- ID istnieje: kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)

5.2.9 Notifications

Get All My Notifications `GET /api/notifs/mynotifs`

Zwraca wszystkie wysłane powiadomienia aktualnie zalogowanego użytkownika.

Odpowiedź: lista obiektów powiadomień, kod 200 (OK)

Get All My Read/Unread Notifications `GET /api/notifs/mynotifs?read={read}`

Zwraca wszystkie wysłane odczytane/nieodczytane powiadomienia aktualnie zalogowanego użytkownika, w zależności od parametru read (true - odczytane, false - nieodczytane).

Odpowiedź: lista obiektów powiadomień, kod 200 (OK)

Get Current User's Notification By File ID `GET /api/notifs/file/{id}`

Zwraca listę niewysłanych powiadomień aktualnie zalogowanego użytkownika powiązanych z plikiem o podanym ID.

Odpowiedzi:

- ID istnieje: lista obiektów powiadomień, kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)
- Nie podano ID: kod 400 (Bad Request)

Create Notification `POST /api/notifs`

Tworzy powiadomienie.

Wymagane jest podanie ID użytkownika odbiorcy oraz pliku. Czas wysłania domyślnie ustawiany jest na czas utworzenia, powiadomienie jest domyślnie nieodczytane. Wysłanie odbywa się na podstawie porównania aktualnego czasu z czasem wysłania.

```
{
  "user": 3,
  "file": 3,
  "message": "Test notif",
  "sendTimeSetting": "2024-10-28T15:30:00"
}
```

Odpowiedzi:

- Pomyślne utworzenie: nowy obiekt powiadomienia, kod 200 (OK)
- Brak ID użytkownika/pliku: kod 400 (Bad Request)
- Nie istnieje ID użytkownika/pliku: kod 404 (Not Found)

Send Current User's Notifications `PUT /api/notifs/send`

Wysyła powiadomienia aktualnie zalogowanego poprzez porównanie ich czasu wysłania z czasem aktualnym.

Odpowiedzi:

- Poprawna aktualizacja powiadomień: kod 200 (OK)
- Nie znaleziono użytkownika: kod 404 (Not Found)

Delete Notification `DELETE /api/notifs/{id}`

Usuwa powiadomienie o podanym ID.

Odpowiedzi:

- ID istnieje: kod 200 (OK)
- ID nie istnieje: kod 404 (Not Found)

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Testowanie aplikacji wymaga sprawdzenia, czy jest ona odporna na nieprawidłowe dane wejściowe, oraz czy dla danych poprawnych zachowuje się zgodnie z oczekiwaniami. Przetestować pod tym kątem należy zarówno back-end, jak i front-end.

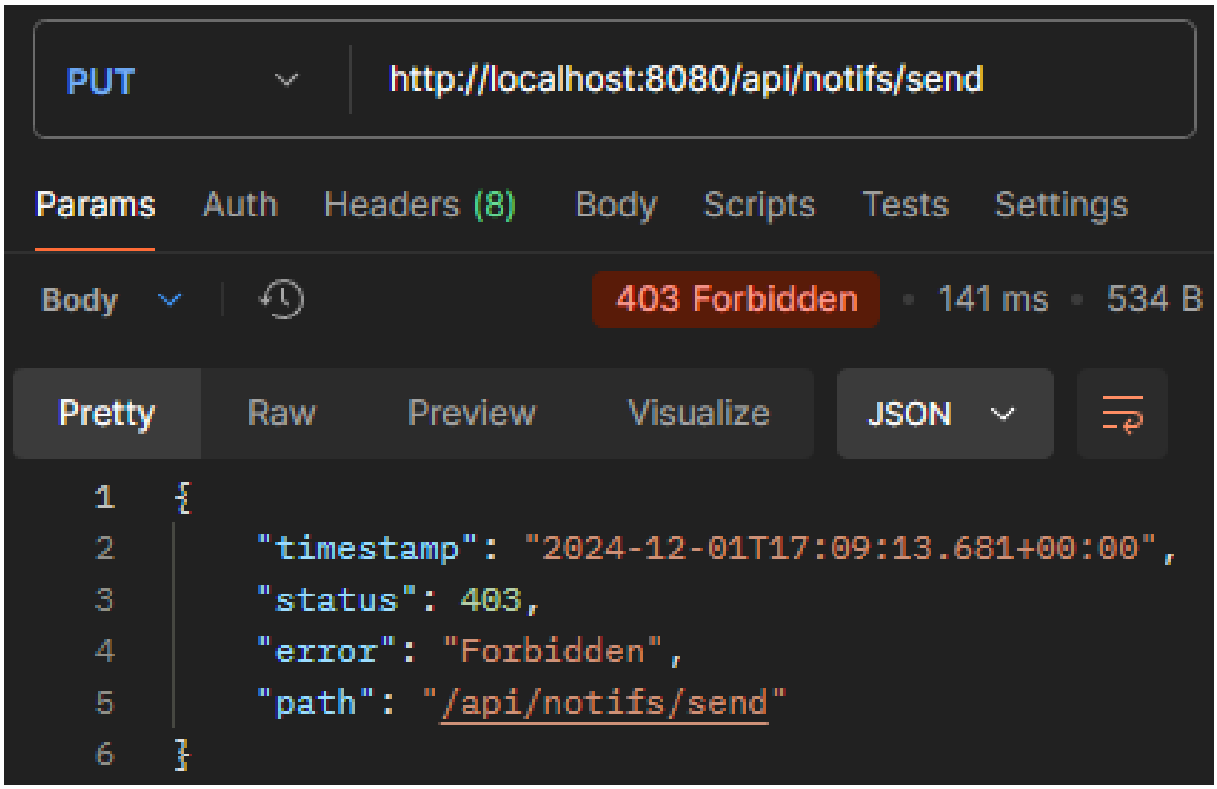
W przypadku back-endu należy zweryfikować, czy faktyczne odpowiedzi są zgodne z wymienionymi w podrozdziale „Lista Endpointów". Z kolei front-end powinien już na poziomie interfejsu w jak największym stopniu ograniczyć możliwość wprowadzenia błędnych danych.

6.0.1 Testowanie back-endu

Testowanie rozpoczęto od sprawdzenia przypadków użycia dla użytkownika niezalogowanego. Większość endpointów wymaga autentykacji, sprawdzono więc, że zwracają one bez niej kod 403 (Forbidden). Przykład takiego zapytania pokazano na rys. 6.1

Przy rejestracji, zgodnie z założeniami, otrzymano kod 400 gdy w ciele zapytania brakowało hasła lub nazwy użytkownika, oraz kod 403 gdy wpisano nazwę użytkownika występującą już w bazie danych. Dla poprawnych danych:

```
{
        "username": "test",
        "password": "password",
        "name": "Test User"
}
```



Rysunek 6.1: Zrzut ekranu z programu Postman przedstawiający próbę użycia endpointu, do którego użytkownik nie posiada dostępu.

Otrzymano odpowiedź:


```
{
        "id": 22,
        "username": "test",
        "name": "Test User",
        "email": null,
        "password": "",
        "role": "ROLE_USER",
        "directories": null,
        "votes": null,
        "notifications": null
}
```


Odpowiedź nie zawiera już hasła – zostało ono zakodowane i zapisane w bazie.

Podobnie przetestowano logowanie: dla brakujących danych otrzymano kod 400 i informację „Empty username or password”, dla nieistniejącego loginu i nieprawidłowego hasła – kod 403 i informację „Incorrect credentials”. Dla poprawnych danych logowania:


```
{
        "username": "test",
```

```
        "password": "password"
}
```

Otrzymano odpowiedź:

```
{
        "id": 22,
        "username": "test",
        "name": "Test User",
        "email": null,
        "password": "",
        "role": "ROLE_USER",
        "directories": [
                26
        ],
        "votes": [],
        "notifications": []
}
```

Obiekt użytkownika został pobrany z bazy danych. Automatycznie utworzony został katalog bazowy, który jest jedynym katalogiem użytkownika nieposiadającym katalogu nadrzędnego.

Ponadto wygenerowane zostały dwa ciasteczka:

```
organizerJwt=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZXN0IiwiaWF0IjoxNzMzMDc1MTYxLCJleHAiOjE
organizerRefreshJwt=1ee7c994-e296-4a12-a636-8cf39380a1e8; Path=/api/auth/refreshtoken
```

Następnie przetestowano przypadki użycia dla użytkownika zwykłego.

Wylogowanie przebiega zawsze pomyślnie, można je powtarzać nawet będąc już wylogowanym bez negatywnych skutków. W wyniku wylogowania obydwa ciasteczka stają się puste.

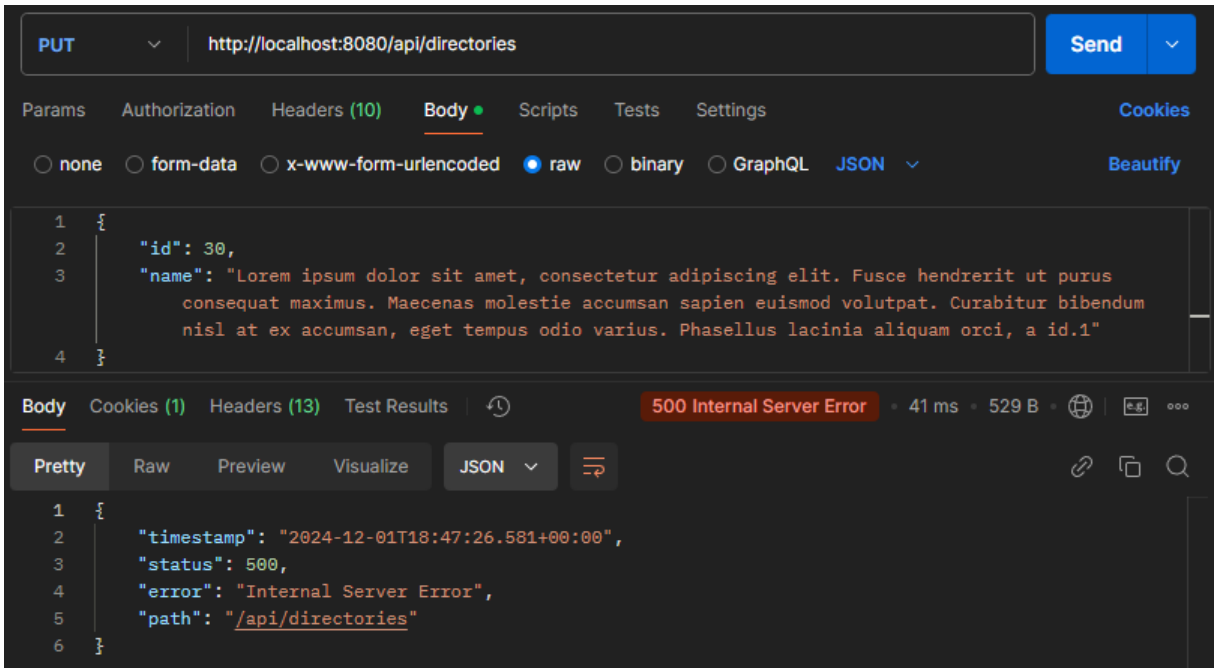
Przetestowano działanie odświeżania ciasteczka organizerJwt. Przed odświeżeniem ciasteczka miały następujące wartości:

```
organizerJwt=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJuZXdVc2VyIiwiaWF0IjoxNzMzMTY2NDA4LCJleHA
organizerRefreshJwt=8d9fab79-5a99-4bb0-b56b-ad0626fb2d61; Path=/api/auth/refreshtoken
```

Po wykonaniu odświeżenia:

```
organizerJwt=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJuZXdVc2VyIiwiaWF0IjoxNzMzMTY2NDg0LCJleHA
organizerRefreshJwt=8d9fab79-5a99-4bb0-b56b-ad0626fb2d61; Path=/api/auth/refreshtoken
```

Próba dostępu do endpointu dostępnego tylko administratorowi, na przykład nadania roli administratora, skutkuje kodem 403 odpowiedzi.



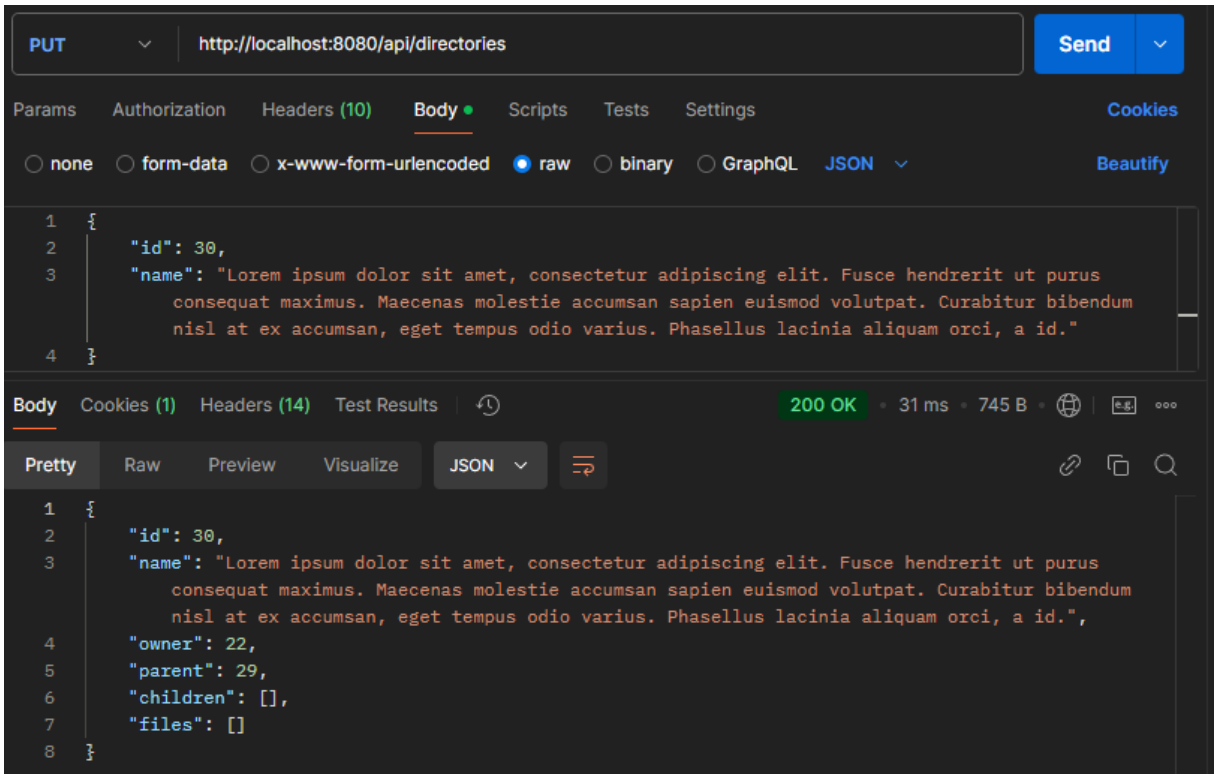
Rysunek 6.2: Zrzut ekranu z programu Postman przedstawiający nieudaną próbę zmiany nazwy katalogu na tekst o długości 256 znaków.

Przetestowano tworzenie nowych elementów. Kody błędów 400, 403 i 404 są zwracane zgodnie z oczekiwaniami. Elementy tworzone są poprawnie, można je tworzyć tylko we własnych katalogach – w przeciwnym razie zwracany jest kod 403. Kolumna textContent jest typu „TEXT” – jest to typ danych bazy MySQL o rozmiarze 64KB. Pozostałe kolumny tekstowe posiadają domyślny typ „TINYTEXT”, mogą więc mieć maksymalnie 255 znaków. Próba przypisania większej liczby znaków skutkuje wyjątkiem w bazie danych. Przykład takiej sytuacji pokazano na rys. 6.2 oraz 6.3 Zabezpieczenie przed nim występuje po stronie front-endu. Back-end nie weryfikuje też, czy termin początkowy wydarzenia jest wcześniejszy od końcowego.

Z takim samym wynikiem przeprowadzono testy edycji nazw i zawartości elementów. Notatki, wydarzenia oraz zadania posiadają osobne endpointy do ich edycji. Próba edycji elementu przy użyciu niewłaściwego endpointu, na przykład wydarzenia używając `PUT /api/files/note`, skutkuje kodem 404.

Element można przenieść tylko do istniejącego katalogu, należącego do tego samego użytkownika. Back-end nie blokuje sytuacji, w której użytkownik przemieszcza elementy innego użytkownika. Ignorowane są przypadki, gdy podejmowana jest próba przeniesienia katalogu bazowego, bądź ustawienia katalogu nadrzędnego na ten sam katalog:

```
{
        "id": 29,
        "parent": 29
}
```



Rysunek 6.3: Zrzut ekranu z programu Postman przedstawiający udaną próbę zmiany nazwy katalogu na tekst o długości 255 znaków.

Odpowiedź:


```
{
        "id": 29,
        "name": "Child Directory",
        "owner": 22,
        "parent": 26,
        "children": [
                30
        ],
        "files": []
}
```


Możliwe jest jednak przeniesienie katalogu do jego katalogu podrzędnego. Skutkować to może nieskończoną pętlą weryfikacji dostępu dla użytkowników niebędących właścicielami tego katalogu i błędem bazy danych podczas usuwania, a sam katalog nie będzie widoczny dla użytkownika w interfejsie graficznym. Zdecydowano się nie weryfikować tej sytuacji na poziomie back-endu ze względu na potencjalną dużą złożoność algorytmu sprawdzającego – konieczne byłoby sprawdzenie całej hierarchii katalogów poniżej edytowanego. Sytuację uniemożliwiono na poziomie front-endu. Zmianę można też łatwo cofnąć kolejną edycją, zmieniając ponownie katalog nadrzędny.

Przetestowano odczytywanie elementów, odpowiedzi były zgodne z wymienionymi na liście endpointów.

W katalogu o ID 29 umieszczono katalog podrzędny oraz plik:


```
{
        "id": 29,
        "name": "Child Directory",
        "owner": 22,
        "parent": 26,
        "children": [
                30
        ],
        "files": [
                15
        ]
}
```


Następnie usunięto katalog 29. Próba odczytu katalogu 29, 30 oraz pliku 15 skutkuje błędem 404 – zawartość katalogu została kaskadowo usunięta.

W przypadku katalogu bazowego, próba usunięcia skutkuje kodem błędu 403.

Sprawdzanie uprawnień do edycji zwraca kod 404 dla nieistniejącego elementu, „true” lub „false” odpowiednio kiedy użytkownik posiada lub nie posiada uprawnień do edycji elementu.

Przetestowano działanie algorytmu dostępu dla użytkownika o ID 4. Zdefiniowano dla niego dostęp do katalogu 3 z poziomem dostępu 1 (odczyt) oraz do plików 1 i 2 z poziomem dostępu 2 (edycja). Struktura wymienionych plików:

```
katalog 1
        katalog 2
                katalog 11
                        plik 2
        katalog 3
                plik 1
                plik 3
```

Próba odczytu plików 1, 2 i 3 oraz katalogu 3 przebiega pomyślnie. Wysłanie zapytania GET dla katalogów 1, 2 i 11 skutkuje kodem 403. Pomyślnie przebiega edycja plików 1 i 2, natomiast dla pliku 3 otrzymywany jest kod 403.

12 grudnia 2024 roku o godzinie 18:10 dla użytkownika 4 dodano powiadomienie do pliku 1.

```
{
        "id": 6,
        "user": 4,
        "file": 1,
        "message": "Test notif",
        "sendTimeSetting": "2024-12-02T18:15:00",
        "read": false,
        "sent": false
}
```


Wysłano od razu zapytanie Send Notifications. Zapytanie Get All Notifications dla użytkownika zwraca pustą listę, a zapytanie Get All Unsent Notifications dla pliku zwraca listę zawierającą to powiadomienie. Po godzinie 18:15 powtórzono wysłanie powiadomień i ponownie wykonano dwa zapytania Get. Tym razem powiadomienie zwrócone zostało przez endpoint `GET /api/notifs/mynotifs` zwróciło listę z powiadomieniem, którego wartość pola `sent` została zmieniona na `true`. Lista niewysłanych powiadomień jest pusta.

Utworzono powiadomienie z wcześniejszym terminem wysłania.


```
{
        "id": 7,
        "user": 4,
        "file": 1,
        "message": "Test notif 2",
        "sendTimeSetting": "2024-12-02T17:15:00",
        "read": false,
        "sent": true
}
```


Powiadomienie od początku jest wysłane.

Możliwe jest również dodanie powiadomienia do pliku, do którego dany użytkownik nie ma dostępu. Nie zablokowano takiej możliwości na poziomie back-endu – dodanie do pliku powiadomienia nie wpływa na dostęp do samego pliku. Podobnie możliwe jest też dodanie powiadomienia dla użytkownika innego niż zalogowany.

Próba utworzenia obiektu EventDate dla wydarzenia należącego do innego użytkownika skutkuje kodem 403, dla nieistniejącego wydarzenia – kodem 404. Podanie ID pliku innego niż wydarzenie również zwraca kod 404. Próba utworzenia wydarzenia z pustym terminem początkowym zwraca kod 400, można jednak nie podać terminu końcowego:


```
{
        "id": 7,
        "event": 16,
```

```
        "votes": null,
        "totalScore": 0,
        "start": "2024-11-18T12:00:00",
        "end": null
}
```


Oddawanie wielu głosów na ten sam termin EventDate modyfikuje wartość `score`, nie dodaje kolejnych głosów. Zalogowano się jako inny użytkownik i oddano głos: wartość `totalScore` dla terminu jest każdorazowo przeliczana, jest równa sumie punktów z każdego głosu. Nie jest sprawdzane, czy głosujący ma dostęp do wydarzenia.

Zwykły użytkownik nie może listować pełnych danych innych użytkowników (kod 403), może jednak skorzystać z bezpiecznego endpointu – dla wszystkich użytkowników oraz użytkownika o konkretnym ID. Fragment odpowiedzi dla bezpiecznego listowania wszystkich użytkowników:


```
{
        "id": 4,
        "username": "newUser",
        "name": "User",
        "email": "",
        "password": null,
        "role": null,
        "directories": null,
        "votes": null,
        "notifications": null
},
{
        "id": 5,
        "username": "newUser2",
        "name": "New User 2",
        "email": "",
        "password": null,
        "role": null,
        "directories": null,
        "votes": null,
        "notifications": null
},
```


Próba edycji danych innego użytkownika niż zalogowany skutkuje kodem błędu 403. Nie można też zmienić nazwy na inną występującą już w bazie. Wyjątkiem jest podanie własnej nazwy użytkownika. Podanie pustego ciągu znaków jako nowa nazwa użytkownika

zwraca kod 403. Zmiany hasła i roli są ignorowane. Na puste ciągi znaków można zmieniać pola `name` oraz `email`.

Przetestowano zmianę hasła. Jeżeli nie podane zostanie stare lub nowe hasło otrzymywany jest kod 400. Dla niepoprawnego starego hasła zwracany jest kod 403. Zmieniono hasło w poprawny sposób i spróbowano zalogować się przy pomocy starego i nowego hasła – operacja przebiegła pomyślnie dla hasła nowego.

Zwykły użytkownik może usunąć tylko swoje konto. Usunięto użytkownika:


```
{
        "id": 22,
        "username": "test",
        "name": "Test User",
        "email": null,
        "password": "",
        "role": "ROLE_USER",
        "directories": [
                26
        ],
        "votes": [
                13
        ],
        "notifications": []
}
```


Automatycznie nastąpiło wylogowanie – endpointy dostępne każdemu zalogowanemu użytkownikowi zwracają kod 403, ciasteczka organizerJwt i organizerRefreshJwt są puste. Zalogowano się na konto innego użytkownika. Próby odczytu katalogu o ID 26 i głosów dla terminów wydarzeń użytkownika zwracają kod 404.

Zalogowano się jako administrator. Możliwe było wylistowanie pełnych danych użytkowników (nie licząc haseł). Fragment odpowiedzi:


```
{
        "id": 4,
        "username": "newUser",
        "name": "User",
        "email": "newuser@email.com",
        "password": "",
        "role": "ROLE_ADMIN",
        "directories": [
                4,
                12,
```

```
                13
        ],
        "votes": [
                1,
                7,
                9,
                11
        ],
        "notifications": [
                6,
                7,
                8
        ]
},
{
        "id": 5,
        "username": "newUser2",
        "name": "New User 2",
        "email": "newuser2@email.com",
        "password": "",
        "role": "ROLE_USER",
        "directories": [
                5
        ],
        "votes": [],
        "notifications": []
},
```


Nadano i odebrano innym użytkownikom rolę administratora. Próba zmiany własnej roli zwróciła kod 400.

Próba usunięcia własnego użytkownika poprzez endpoint administratora zwraca kod 403. Usunięcie innego użytkownika przebiegło pomyślnie.

6.0.2 Wykryte i usunięte błędy

Autentykacja: podczas logowania generowane są dwa ciasteczka: organizerJwt oraz organizerRefreshJwt. Drugie z ciasteczek służy do odświeżania pierwszego, powinno więc mieć dłuższy czas ważności. Przez przypadek do obydwu ciasteczek przypisywano tę samą wartość maxAge.

Odświeżanie ciasteczek: odświeżane było jedynie ciasteczko organizerJwt zamiast obydwu.

Tworzenie elementu: dodano obsługę sytuacji, gdzie katalog nadrzędny o podanym ID nie istnieje lub należy do innego użytkownika.

Usuwanie elementu: element powinien móc usunąć jedynie jego właściciel. Przypadkowo zrealizowano sytuację odwrotną: wszyscy użytkownicy niebędący właścicielem mogli usuwać elementy.

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna . . .)
- problemy napotkane w trakcie pracy

Bibliografia

[1] *Angular*. 2024. URL: https://angular.dev/ (term. wiz. 20. 11. 2024).

[2] *Angular Material UI component library*. 2024. URL: https://material.angular.io/ (term. wiz. 20. 11. 2024).

[3] *Angular v17 documentation*. 2024. URL: https://v17.angular.io/docs (term. wiz. 20. 11. 2024).

[4] *Docker Docs*. 2024. URL: https://docs.docker.com/ (term. wiz. 27. 11. 2024).

[5] *Entity Framework documentation hub*. 2024. URL: https://learn.microsoft.com/en-us/ef/ (term. wiz. 20. 11. 2024).

[6] *Luxon*. 2024. URL: https://moment.github.io/luxon/#/ (term. wiz. 21. 11. 2024).

[7] *Material Design*. 2024. URL: https://m2.material.io/ (term. wiz. 20. 11. 2024).

[8] *MySQL documentation*. 2024. URL: https://dev.mysql.com/doc/ (term. wiz. 20. 11. 2024).

[9] Imię Nazwisko i Imię Nazwisko. *Tytuł strony internetowej*. 2021. URL: http://gdzies/w/internecie/internet.html (term. wiz. 30. 09. 2021).

[10] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu konferencyjnego”. W: *Nazwa konferecji*. 2006, s. 5346–5349.

[11] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu w czasopiśmie”. W: *Tytuł czasopisma* 157.8 (2016), s. 1092–1113.

[12] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. *Tytuł książki*. Warszawa: Wydawnictwo, 2017. ISBN: 83-204-3229-9-434.

[13] *Postman*. 2024. URL: https://www.postman.com/ (term. wiz. 01. 12. 2024).

[14] *React*. 2024. URL: https://react.dev/ (term. wiz. 20. 11. 2024).

[15] *Spring Framework*. 2024. URL: https://spring.io/ (term. wiz. 20. 11. 2024).

Dodatki

Spis skrótów i symboli

DNA kwas deoksyrybonukleinowy (ang. *deoxyribonucleic acid*)

MVC model – widok – kontroler (ang. *model–view–controller*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

$\mathbb{E}$ zbiór krawędzi grafu

$\mathcal{L}$ transformata Laplace'a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
          iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
          epsilon set — you should set either number of iterations
          or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel