Uwagi

1: Wstęp napiszemy na końcu.

5: słowo zagajenia – jakieś zdanie wprowadzenia zanim pojawi się spis użytkowników

6: słowo zagajenia

7: W jaki sposób zweryfikujemy, czy to wymaganie zostało spełnione?

9: W jaki sposób zweryfikujemy, czy to wymaganie zostało spełnione?

9: cytowanie dokumentacji

Politechnika
Śląska

PROJEKT INŻYNIERSKI

Tytuł pracy dyplomowej inżynierskiej

Imię NAZWISKO
Nr albumu: ⟨wpisać właściwy⟩

Kierunek: ⟨wpisać właściwy⟩
Specjalność: ⟨wpisać właściwą⟩

PROWADZĄCY PRACĘ
⟨tytuł lub stopień naukowy oraz imię i nazwisko⟩
KATEDRA ALGORYTMIKI I OPROGRAMOWANIA
Wydział Automatyki, Elektroniki i Informatyki

OPIEKUN, PROMOTOR POMOCNICZY
⟨stopień naukowy imię i nazwisko⟩

Gliwice 2024

Tytuł pracy

Tytuł pracy dyplomowej inżynierskiej

Streszczenie

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

Słowa kluczowe

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

Thesis title

Thesis title in English

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

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.

Wzory

$$y = \frac{\partial x}{\partial t} \tag{2.1}$$

jak i pojedyncze symbole x i y składa się w trybie matematycznym.

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

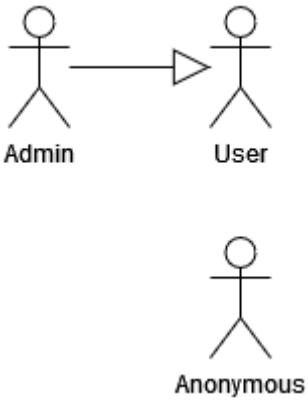
[słowo zagajenia – jakieś zdanie wprowadzenia zanim pojawi się spis użytkowników]
Znaczna większość funkcjonalności aplikacji wymaga, aby znany był aktualnie zalogowany użytkownik – od edycji profilu po przeglądanie plików. Ponadto istnieć powinien użytkownik specjalny (administrator), który zarządzać może pozostałymi użytkownikami. Wyróżnić można więc 3 rodzaje użytkowników:

U.1 Użytkownik zwykły: edytuje profil, tworzy, przegląda, modyfikuje, usuwa, udostępnia elementy.

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

U.3 Użytkownik niezalogowany: loguje lub rejestruje się.

Zależności między użytkownikami przedstawiono na rys. 3.1.



Rysunek 3.1: Użytkownicy aplikacji.

3.1.2 Funkcjonalności

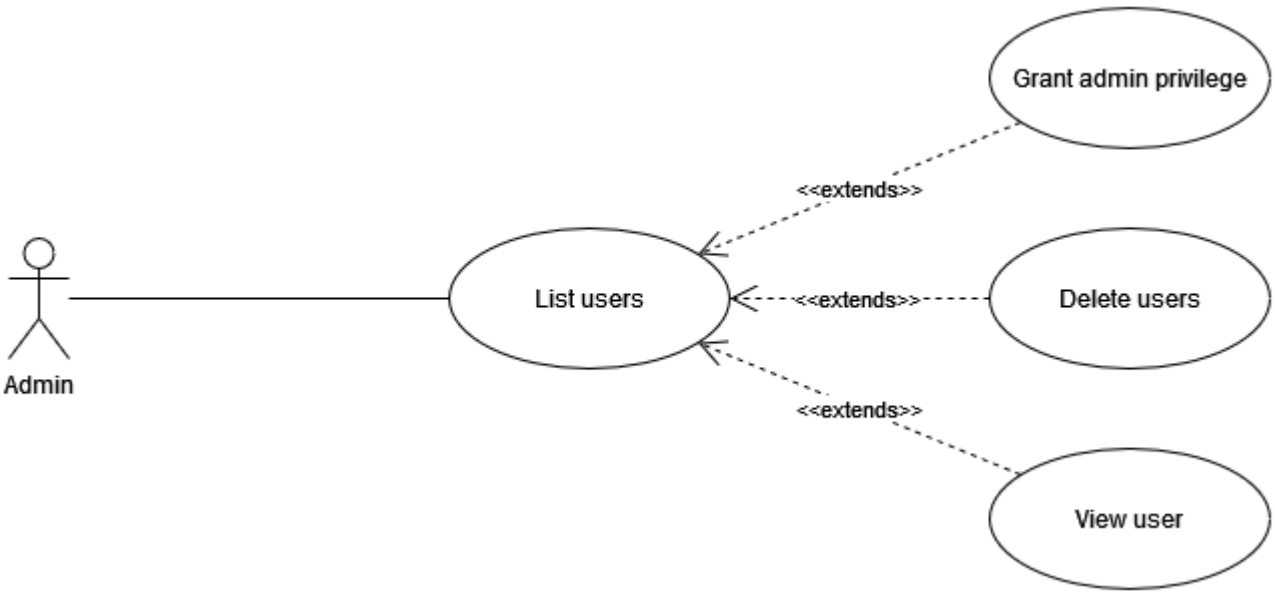
[słowo zagajenia] Dostępne użytkownikowi funkcje aplikacji można ogólnie podzielić na operacje związane z kontem – autoryzacją i edycją – oraz związane z plikami – przeglądanie, edycja, tworzenie nowych plików.

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.

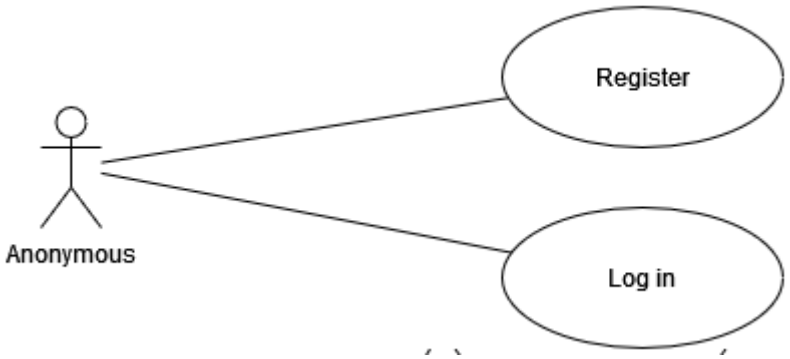
F.5. Odporność na nieprawidłowe dane wejściowe.

Wymagania funkcjonalne dla poszczególnych użytkowników przedstawiono na rys. **??**.

3.2 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. [W jaki sposób zweryfikujemy, czy to wymaganie zostało spełnione?]

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. [W jaki sposób zweryfikujemy, czy to wymaganie zostało spełnione?]

3.3 Narzędzia

Do implementacji back-endu wybrano framework Spring [8], 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 [4] przemawiała również posiadana już wiedza o REST API oraz chęć pogłębienia znajomości samego Springa. Wspieraną przez Spring bazę danych MySQL [cytowanie dokumentacji] oraz sam back-end postanowiono uruchomić w kontenerze Dockera [6].

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 [7], 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 [**material**]. Wykorzystanie gotowych komponentów pozwala na skupienie się na samej logice programu. Do operacji matematycznych na datach użyto biblioteki Luxon [5].

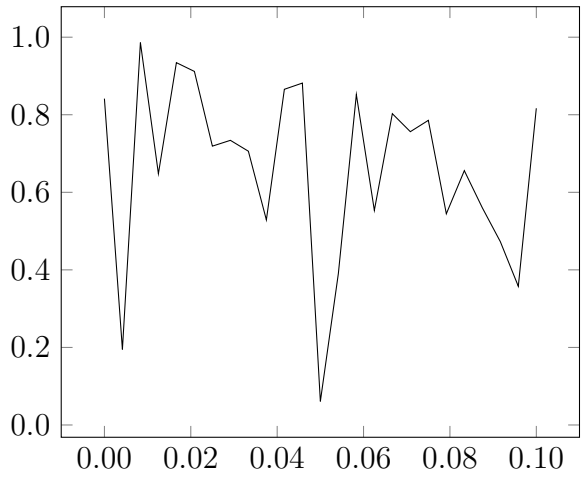
[1] `https://trends.stackoverflow.co/?tags=reactjs,vue.js,angular,svelte,angularjs,vuejs3`, dostęp 08.11.2024

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)



Rysunek 4.1: Podpis rysunku po rysunkiem.

Rozdział 5

Specyfikacja wewnętrzna

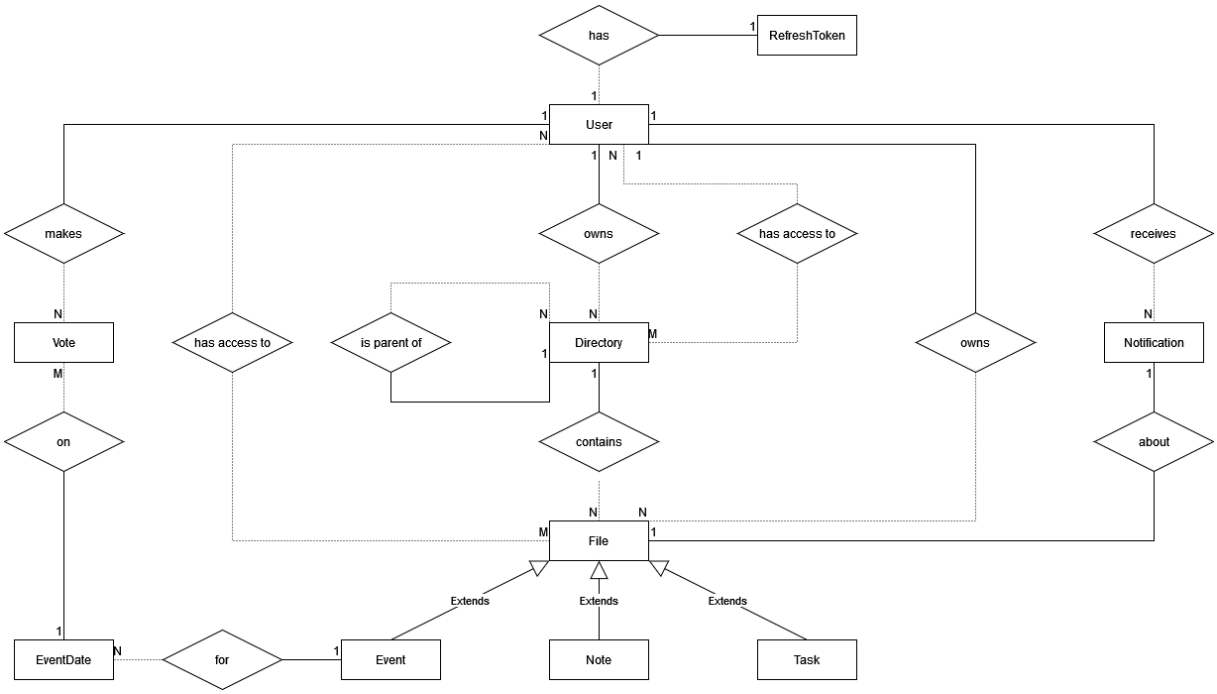
Jeśli „Specyfikacja wewnętrzna”:

- 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 5.1, a naprawdę długie fragmenty – w załączniku.

```
1  class test : public basic
2  {
3      public:
4        test (int a);
5        friend std::ostream operator<<(std::ostream & s,
6                                       const test & t);
7      protected:
8        int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w `listings`.



Rysunek 5.2: Schemat ERD klas w bazie danych.

5.0.1 Lista klas

Zależności między klasami przedstawiono na rys. 5.2.
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, location, deadline, is_finished, owner_id (FK), parent_id (FK))
- Notification(id (PK), message, send_time_setting, is_sent, is_read, file_id (FK), user_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.

5.0.2 Lista endpointów

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

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/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)

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)

Update User `PUT /api/users`

Aktualizuje użytkownika.

Wymagane podanie ID użytkownika w ciele zapytania.

```
{
    "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)
- Próba zmiany nazwy użytkownika na już istniejącą/nie podano ID: kod 400 (Bad Request)

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)

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)

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: 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)

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)

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: 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)

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: 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)

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: 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)

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)

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)

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.

```
{
    "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)

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

Usuwa obiekt EventDate o podanym ID, wraz z jego głosami.

Odpowiedzi:

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

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)

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

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

	metoda						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

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] Imię Nazwisko i Imię Nazwisko. *Angular*. 2024. URL: https://angular.dev/ (term. wiz. 20. 11. 2024).

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

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

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

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

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

[7] Imię Nazwisko i Imię Nazwisko. *React*. 2024. URL: https://react.dev/ (term. wiz. 20. 11. 2024).

[8] Imię Nazwisko i Imię Nazwisko. *Spring Framework*. 2024. URL: https://spring.io/ (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.

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