

# LMS-API DOC

---

```
=====
/_____|  /\  |_____|  |_____|  /\  |_____|  |_____|  |_____|
| (_____|  /\  |_____|  |_____|  |_____|  |_____|  |_____|
\_____|  /\  |_____|  |_____|  |_____|  |_____|  |_____|
_____) |  /\  |_____|  |_____|  |_____|  |_____|  |_____|
|_____|  /\  |_____|  |_____|  |_____|  |_____|  |_____|
=====
:: Spring Boot      v3.2.0 ::                               :: 2024 ::
:: api version      v0.2.2 ::
```

Dokumentacja do aplikacji rest api projektu LSM.

## Spis treści

1. [Plik konfiguracyjny](#)
2. [Zadeklarowane stałe](#)
  - [Statusy](#)
  - [Role](#)
  - [Flagi](#)
3. [Autentykacja użytkownika](#)
  - [Rejestracja nowego użytkownika](#)
  - [Logowanie użytkownika](#)
4. [Użytkownicy](#)
  - [Pobieranie listy wszystkich użytkowników](#)
  - [Pobieranie pojedynczego użytkownika](#)
  - [Pobieranie pojedynczego użytkownika po loginie](#)
  - [Usuwanie użytkownika](#)
  - [Dodawanie nowego użytkownika](#)
  - [Aktualizacja danych użytkownika](#)
5. [Przedmioty](#)
  - [Pobieranie listy wszystkich przedmiotów](#)
  - [Pobieranie pojedynczego przedmiotu](#)
  - [Pobieranie pojedynczego przedmiotu po kodzie](#)
  - [Usuwanie przedmiotu](#)
  - [Dodawanie nowego przedmiotu](#)
  - [Aktualizacja danych przedmiotu](#)
6. [Okresy](#)
  - [Pobieranie listy wszystkich okresów](#)
  - [Pobieranie pojedynczego okresu](#)
  - [Usuwanie okresu](#)
  - [Dodawanie nowego okresu](#)
  - [Aktualizacja danych okresu](#)

## 7. Rejestracja

- Rejestracja ucznia na przedmiot
- Pobieranie listy lub konkretnego powiązania
- Pobieranie powiązania o konkretnym numerze ID
- Wystawienie oceny uczniowi
- Wyrejestrowanie ucznia z przedmiotu

## 8. Zadania

- Rodzaje zadań
- Pobieranie listy wszystkich zadań
- Pobieranie pojedynczego zadania
- Pobieranie aktywnych zadań
- Usuwanie zadania
- Dodawanie nowego zadania
- Aktualizacja danych zadania

## 9. Odpowiedzi

- Rodzaje odpowiedzi
- Pobieranie listy wszystkich odpowiedzi
- Pobieranie pojedynczej odpowiedzi
- Dodawanie nowej odpowiedzi
- Aktualizacja danych odpowiedzi
- Usunięcie odpowiedzi
- Wystawienie oceny dla zadania

## 10. Materiały

- Pobieranie listy wszystkich materiałów
- Pobieranie pojedynczego materiału
- Usuwanie materiału
- Dodawanie nowego materiału
- Aktualizacja danych materiału

## 11. Powiadomienia

- Pobieranie listy wszystkich powiadomień dla użytkownika
- Pobieranie pojedynczego powiadomienia
- Usuwanie powiadomienia
- Aktualizacja flagi powiadomienia

---

## Plik konfiguracyjny

**Nazwa pliku:** `application.properties`

```
### KONFIGURACJA SERWERA ###
# PORT SERWERA - Port, na którym aplikacja nasłuchuje
server.port=8080
# ADRES SERWERA - Adres IP, na którym działa serwer
server.address=127.0.0.1

### KONFIGURACJA BAZY DANYCH ###
# URL BAZY DANYCH
```

```
spring.datasource.url=jdbc:mysql://<data-base-url>
# NAZWA UŻYTKOWNIKA BAZODANOWEGO
spring.datasource.username=<user-name>
# HASŁO UŻYTKOWNIKA BAZODANOWEGO
spring.datasource.password=<password>

### KONFIGURACJA BEZPIECZEŃSTWA ###
spring.security.filter.order=10
# SEKRET BEZPIECZEŃSTWA - klucz, na podstawie którego generowane są tokeny JWT
security.auth.secret=~a?%B^"}i[xu}~IhA+B0'nGS8G(o5x
```

## Zadeklarowane stałe

### Statusy

#### Statusy przedmiotów:

1. **DO\_ZATWIERDZENIA (2AP)** - Oczekuje na zatwierdzenie.
2. **ZATWIERDZONY (APR)** - Został zatwierdzony.
3. **ODRZUCONY (REJ)** - Został odrzucony.
4. **TRWAJĄCY (ACT)** - Jest aktualnie w trakcie.
5. **ZAKOŃCZONY (END)** - Zakończony.

#### Statusy użytkowników:

1. **AKTYWNY (1)** - Aktywny.
2. **NIEAKTYWNY (0)** - Nieaktywny.

### Role

1. **ADMIN (1)**
2. **NAUCZYCIEL (3)**
3. **UCZEN (2)**

### Flagi

1. **NIEPRZECZYTANA** - Oznacza, że obiekt nie został jeszcze przeczytany.
2. **PRZECZYTANA** - Oznacza, że obiekt został przeczytany.
3. **ZARCHIWIZOWANA** - Oznacza, że obiekt został zarchiwizowany.
4. **USUNIĘTA** - Oznacza, że obiekt został usunięty.
5. **NOWA** - Oznacza, że obiekt jest nowy.
6. **ROBOCZA** - Oznacza, że obiekt jest wersją roboczą lub tymczasową.

### TypyZadan

1. **ZADANIE (1)**
2. **TEST (2)**
3. **EGZAMIN (3)**

## Autentykacja użytkownika

## Opis

Klasa **AuthController** odpowiada za obsługę end-pointów związanych z rejestracją i logowaniem użytkowników. W ramach dokumentacji przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

## End-pointy

### 1. Rejestracja nowego użytkownika

#### WAŻNE!

Login użytkownika nadawany jest automatycznie w konwencji <pierwsza\_litera\_imienia>.[?nr\_porządkowy]

W bazie danych przechowywane jest zaszyfrowane hasło przy użyciu algorytmu **BCrypt**

Gdy nie zostanie przekazane zdjęcie, ustawione zostanie na domyślne (id 1).

- **Ścieżka:** `/api/v1/auth/register`
- **Metoda:** `POST`
- **Parametry:**
  - **uzytkownik** (ciało zapytania, wymagane): Obiekt reprezentujący dane użytkownika.
- **Odpowiedź:**
  - `201 Created` - sukces, konto zostało utworzone.
  - `500 Internal Server Error` - błąd utworzenia konta.

```
POST /api/v1/auth/register
Content-Type: application/json
```

```
{
  "imie": "Jan",
  "nazwisko": "Kowalski",
  "tytNauk": null,
  "email": "john.doe@example.com",
  "haslo": "haslo",
  "telefon": 123456789,
  "dataUrodz": "1990-01-01",
  "status": "AKTYWNY",
  "rola": "UCZEN",
  "zdjecie": {
    "plik": "<plik_binarny_base_64>",
    "nazwa": "zdjecie",
    "ext": "png",
    "alt": "zdjecie-uzytkownika"
  }
}
```

## 2. Logowanie użytkownika

- **Ścieżka:** `/api/v1/auth/login`
- **Metoda:** `POST`
- **Parametry:**
  - `request` (ciało zapytania, wymagane): Obiekt zawierający dane logowania (login, hasło).
- **Odpowiedzi:**
  - `200 OK` - sukces, logowanie udane, zwraca token JWT.
  - `400 Bad Request` - błędny login lub hasło.
  - `400 Bad Request` - inny błąd podczas logowania.

```
POST /api/v1/auth/login
Content-Type: application/json
```

```
{
  "login": "j.doe",
  "haslo": "haslo"
}
```

### Odpowiedź:

```
{
  "login": "j.doe",
  "token": "<jwt-token>"
}
```

Pozostałe zapytania wymagają uwierzytelnienia, dlatego należy dołączać do zapytania:

```
Authorization: Bearer <token>
```

## Użytkownicy

### Opis

Klasa `UzytkownicyController` odpowiada za obsługę end-pointów związanych z zarządzaniem użytkownikami w systemie. W ramach dokumentacji przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

### End-pointy

#### 1. Pobieranie listy wszystkich użytkowników

- **Ścieżka:** `/api/uzytkownik/all`

- **Metoda:** GET
- **Parametry:**
  - **size** (opcjonalny): liczba elementów na stronie
  - **page** (opcjonalny): numer strony (liczony od 0)
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca listę użytkowników w formacie JSON
  - **404 Not Found** - brak użytkowników

```
GET /api/uzytkownik/all
Authorization: Bearer <token>
```

## 2. Pobieranie pojedynczego użytkownika

- **Ścieżka:** /api/uzytkownik/{id}
- **Metoda:** GET
- **Parametry:**
  - **id** (ścieżka): identyfikator użytkownika
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca dane użytkownika w formacie JSON
  - **404 Not Found** - użytkownik o podanym identyfikatorze nie istnieje

```
GET /api/uzytkownik/1
Authorization: Bearer <token>
```

## 3. Pobieranie pojedynczego użytkownika po loginie

- **Ścieżka:** /api/uzytkownik
- **Metoda:** GET
- **Parametry:**
  - **login** (parametr zapytania): login użytkownika (zakodowany w Base64)
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca dane użytkownika w formacie JSON
  - **404 Not Found** - użytkownik o podanym loginie nie istnieje

```
GET /api/uzytkownik/?login=<login_base_64>
Authorization: Bearer <token>
```

## 4. Usuwanie użytkownika

- **Ścieżka:** /api/uzytkownik/{id}
- **Metoda:** DELETE
- **Parametry:**
  - **id** (ścieżka): identyfikator użytkownika

- **Odpowiedź:**
  - **204 No Content** - sukces, użytkownik został usunięty
  - **404 Not Found** - użytkownik o podanym identyfikatorze nie istnieje

```
DELETE /api/uzytkownik/<nr_id>
Authorization: Bearer <token>
```

## 5. Dodawanie nowego użytkownika

- **Ścieżka:** `/api/uzytkownik`
- **Metoda:** `POST`
- **Parametry:**
  - Ciało żądania zawiera dane nowego użytkownika w formacie JSON
- **Odpowiedź:**
  - **201 Created** - sukces, użytkownik został dodany, zwraca link do nowo utworzonego użytkownika
  - **400 Bad Request** - błąd w danych wejściowych

### WAŻNE!

Login użytkownika nadawany jest automatycznie w konwencji `<pierwsza_litera_imienia>.[?nr_porządkowy]`

W bazie danych przechowywane jest zaszyfrowane hasło przy użyciu algorytmu `BCrypt`

Gdy nie zostanie przekazane zdjęcie, ustawione zostanie na domyślne (id 1).

```
POST /api/uzytkownik
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "imie": "Jan",
  "nazwisko": "Kowalski",
  "tytNauk": null,
  "email": "john.doe@example.com",
  "haslo": "haslo",
  "telefon": 123456789,
  "dataUrodz": "1990-01-01",
  "status": "AKTYWNY",
  "rola": "UCZEN",
  "zdjecie": {
    "plik": "<plik_binarny_base_64>",
    "nazwa": "zdjecie",
    "ext": "png",
    "alt": "zdjecie-uzytkownika"
```

```
}  
}
```

## 6. Aktualizacja danych użytkownika

- **Ścieżka:** `/api/uzytkownik/{id}`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): identyfikator użytkownika
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON -**Odpowiedź:**
  - `200 OK` - sukces, użytkownik został zaktualizowany, zwraca link do zaktualizowanego użytkownika
  - `400 Bad Request` - błąd w danych wejściowych
  - `404 Not Found` - użytkownik o podanym identyfikatorze nie istnieje

Przykład:

```
PATCH /api/uzytkownik/1  
Content-Type: application/json  
Authorization: Bearer <token>
```

```
{  
  "nazwisko": "Smith",  
  "email": "john.smith@example.com"  
}
```

## Przedmioty

### Opis

Klasa `PrzedmiotyController` odpowiada za obsługę end-pointów związanych z zarządzaniem przedmiotami w systemie. W ramach dokumentacji przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

### End-pointy

#### 1. Pobieranie listy wszystkich przedmiotów

- **Ścieżka:** `/api/przedmiot/all`
- **Metoda:** `GET`
- **Parametry:**
  - `size` (opcjonalny): liczba elementów na stronie
  - `page` (opcjonalny): numer strony (liczony od 0) (liczony od 0)
- **Odpowiedź:**



- 200 OK - sukces, zwraca listę przedmiotów w formacie JSON
- 404 Not Found - brak przedmiotów

```
GET /api/przedmiot/all
Authorization: Bearer <token>
```

## 2. Pobieranie pojedynczego przedmiotu

- **Ścieżka:** /api/przedmiot/{id}
- **Metoda:** GET
- **Parametry:**
  - id (ścieżka): identyfikator przedmiotu
- **Odpowiedź:**
  - 200 OK - sukces, zwraca dane przedmiotu w formacie JSON
  - 404 Not Found - przedmiot o podanym identyfikatorze nie istnieje

```
GET /api/przedmiot/1
Authorization: Bearer <token>
```

## 3. Pobieranie pojedynczego przedmiotu po kodzie

- **Ścieżka:** /api/przedmiot
- **Metoda:** GET
- **Parametry:**
  - kod (parametr zapytania): kod przedmiotu (zakodowany w Base64)
- **Odpowiedź:**
  - 200 OK - sukces, zwraca dane przedmiotu w formacie JSON
  - 404 Not Found - przedmiot o podanym kodzie nie istnieje

```
GET /api/przedmiot/?kod=<kod_base_64>
Authorization: Bearer <token>
```

## 4. Usuwanie przedmiotu

- **Ścieżka:** /api/przedmiot/{id}
- **Metoda:** DELETE
- **Parametry:**
  - id (ścieżka): identyfikator przedmiotu
- **Odpowiedź:**
  - 204 No Content - sukces, przedmiot został usunięty
  - 404 Not Found - przedmiot o podanym identyfikatorze nie istnieje

```
DELETE /api/przedmiot/<nr_id>
Authorization: Bearer <token>
```

## 5. Dodawanie nowego przedmiotu

- **Ścieżka:** `/api/przedmiot`
- **Metoda:** `POST`
- **Parametry:**
  - Ciało żądania zawiera dane nowego przedmiotu w formacie JSON
- **Odpowiedź:**
  - `201 Created` - sukces, przedmiot został dodany, zwraca link do nowo utworzonego przedmiotu
  - `400 Bad Request` - błąd w danych wejściowych

### WAŻNE!

Kod przedmiotu nadawany jest automatycznie w konwencji:



znaki okresu>/<4 znaki nazwy przedmiotu>/<numer porządkowy>

```
POST /api/przedmiot
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "nazwa": "Matematyka",
  "idProwadzacego": 123,
  "limit": 30,
  "opis": "Przedmiot z zakresu matematyki",
  "warunkiZaliczenia": "Egzamin końcowy",
  "idOkresu": 1,
  "status": "DO_ZATWIERDZENIA",
  "czyRejestrUczn": true
}
```

## 6. Aktualizacja danych przedmiotu

- **Ścieżka:** `/api/przedmiot/{id}`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): identyfikator przedmiotu
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON
- **Odpowiedź:**
  - `200 OK` - sukces, przedmiot został zaktualizowany, zwraca link do zaktualizowanego przedmiotu
  - `400 Bad Request` - błąd w danych wejściowych

- **404 Not Found** - przedmiot o podanym identyfikatorze nie istnieje

```
PATCH /api/przedmiot/1
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "nazwa": "Fizyka",
  "opis": "Przedmiot z zakresu fizyki"
}
```

## Okresy

### Opis

Klasa **OkresyController** odpowiada za obsługę end-pointów związanych z zarządzaniem okresami przedmiotów w systemie. W ramach dokumentacji przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

### End-pointy

#### 1. Pobieranie listy wszystkich okresów

- **Ścieżka:** `/api/przedmiot/okres/all`
- **Metoda:** `GET`
- **Parametry:**
  - **size** (opcjonalny): liczba elementów na stronie
  - **page** (opcjonalny): numer strony (liczony od 0)
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca listę okresów w formacie JSON
  - **404 Not Found** - brak okresów

```
GET /api/przedmiot/okres/all
Authorization: Bearer <token>
```

#### 2. Pobieranie pojedynczego okresu

- **Ścieżka:** `/api/przedmiot/okres/{id}`
- **Metoda:** `GET`
- **Parametry:** **id** (ścieżka): identyfikator okresu
- **Odpowiedź:** **200 OK** - sukces, zwraca dane okresu w formacie JSON **404 Not Found** - okres o podanym identyfikatorze nie istnieje

```
GET /api/przedmiot/okres/1
Authorization: Bearer <token>
```

### 3. Usuwanie okresu

- **Ścieżka:** `/api/przedmiot/okres/{id}`
- **Metoda:** `DELETE`
- **Parametry:**
  - `id` (ścieżka): identyfikator okresu
- **Odpowiedź:**
  - `204 No Content` - sukces, okres został usunięty
  - `404 Not Found` - okres o podanym identyfikatorze nie istnieje

```
DELETE /api/przedmiot/okres/<nr_id>
Authorization: Bearer <token>
```

### 4. Dodawanie nowego okresu

- **Ścieżka:** `/api/przedmiot/okres`
- **Metoda:** `POST`
- **Parametry:**
  - Ciało żądania zawiera dane nowego okresu w formacie JSON
- **Odpowiedź:**
  - `201 Created` - sukces, okres został dodany, zwraca link do nowo utworzonego okresu
  - `400 Bad Request` - błąd w danych wejściowych

```
POST /api/przedmiot/okres
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "kod": "SEM01",
  "dataPoczatku": "2023-01-01T00:00:00",
  "dataKonca": "2023-02-28T00:00:00"
}
```

### 5. Aktualizacja danych okresu

- **Ścieżka:** `/api/przedmiot/okres/{id}`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): identyfikator okresu

- Ciało żądania zawiera dane do aktualizacji w formacie JSON
- **Odpowiedź:**
  - **200 OK** - sukces, okres został zaktualizowany, zwraca link do zaktualizowanego okresu
  - **400 Bad Request** - błąd w danych wejściowych
  - **404 Not Found** - okres o podanym identyfikatorze nie istnieje

```
PATCH /api/przedmiot/okres/1
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "dataPoczatku": "2023-02-01T00:00:00",
  "dataKonca": "2023-03-31T00:00:00"
}
```

## Rejestracja

### Opis

Klasa **UczenPrzedmiotController** odpowiada za obsługę end-pointów związanych z rejestracją ucznia na przedmiot w systemie i wystawianiem mu oceny. W ramach dokumentacji przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

### End-pointy

#### 1. Rejestracja ucznia na przedmiot

- **Ścieżka:** **/api/przedmiot/uczen/rejestruj**
- **Metoda:** **POST**
- **Parametry:**
  - **nick** (parametr zapytania, wymagany): nick (login) ucznia w formacie Base64
  - **kod** (parametr zapytania, wymagany): kod przedmiotu w formacie Base64
- **Odpowiedź:**
  - **201 Created** - sukces, uczniowi został przypisany przedmiot
  - **400 Bad Request** - błąd w danych wejściowych

```
POST /api/przedmiot/uczen/rejestruj?nick=<nick_base64>&kod=<kod_base64>
Authorization: Bearer <token>
```

#### 2. Pobieranie listy lub konkretnego powiązania:

Powiązanie można pobrać po:

- **nicku (loginie)** ucznia wtedy jest to lista wszystkich przedmiotów ucznia

- **kodzie przedmiotu**, wtedy jest to lista wszystkich uczniów w danym przedmiocie
- **nicku i kodzie**, wtedy pobiera konkretny powiązanie ucznia i przedmiotu
- **bez żadnych parametrów**, wtedy pobiera wszystko
- **Ścieżka:** `/api/przedmiot/uczen`
- **Metoda:** `GET`
- **Parametry:**
  - `nick` (parametr zapytania, opcjonalny): nick (login) ucznia w formacie Base64
  - `kod` (parametr zapytania, opcjonalny): kod przedmiotu w formacie Base64
  - `size` (parametr zapytania, opcjonalny): liczba wyników na stronie
  - `page` (parametr zapytania, opcjonalny): numer strony, liczony od 0
- **Odpowiedź:**
  - `200 OK` - sukces, zwraca kolekcję modeli powiązań w formacie JSON
  - `404 Not Found` - brak powiązań

```
GET /api/przedmiot/uczen?nick=<nick_base64>&kod=<kod_base64>&size=<size>&page=
<page>
```

```
Authorization: Bearer <token>
```

### 3. Pobieranie powiązania o konkretnym numerze ID

- **Ścieżka:** `/api/przedmiot/uczen/{id}`
- **Metoda:** `GET`
- **Parametry:**
  - `id` (ścieżka): numer ID powiązania
- **Odpowiedź:**
  - `200 OK` - sukces, zwraca model powiązania w formacie JSON
  - `404 Not Found` - brak powiązania o podanym ID

```
GET /api/przedmiot/uczen/{id}
```

```
Authorization: Bearer <token>
```

### 4. Wystawienie oceny uczniowi

- **Ścieżka:** `/api/przedmiot/uczen/ocena`
- **Metoda:** `PATCH`
- **Parametry:**
  - `ocena` (parametr zapytania, wymagany): ocena do wystawienia
  - `nick` (parametr zapytania, wymagany): nick (login) ucznia w formacie Base64
  - `kod` (parametr zapytania, wymagany): kod przedmiotu w formacie Base64
- **Odpowiedź:**
  - `200 OK` - sukces, ocena została wystawiona
  - `404 Not Found` - brak powiązania o podanym nicku i kodzie

```
PATCH /api/przedmiot/uczen/ocena?ocena=<ocena>&nick=<nick_base64>&kod=<kod_base64>
Authorization: Bearer <token>
```

## 5. Wyrejestrowanie ucznia z przedmiotu

- **Ścieżka:** `/api/przedmiot/uczen/wyrejestruj`
- **Metoda:** `DELETE`
- **Parametry:**
  - `nick` (parametr zapytania, wymagany): nick (login) ucznia w formacie Base64
  - `kod` (parametr zapytania, wymagany): kod przedmiotu w formacie Base64
- **Odpowiedź:**
  - `204 No Content` - sukces, ucznia został wyrejestrowany z przedmiotu
  - `404 Not Found` - brak powiązania o podanym nicku i kodzie

```
DELETE /api/przedmiot/uczen/wyrejestruj?nick=<nick_base64>&kod=<kod_base64>
Authorization: Bearer <token>
```

## Zadania

### Rodzaje zadań

Zadania przekazywane w `tresc` mają format listy obiektów json. To pole jest typu `String`, więc wszystkie " muszą być poprzedzone znakiem `escape`: `\` . Poniżej typy zadań:

#### 1. Zadania otwarte:

```
{
  "typ": "OTWARTE",
  "pytanie": "Bardzo trudne pytanie wymagające rozbudowanej odpowiedzi",
  "punkty": 10
}
```

#### 2. Zadania zamknięte:

```
{
  "typ": "ZAMKNIETE",
  "pytanie": "Pytanie testowe z czterema odpowiedziami do wyboru, gdzie dwie są prawdziwe",
  "odpowiedz": ["odpowiedź1", "odpowiedź2", "odpowiedź3", "odpowiedź3"],
  "poprawneOdp": [1, 3],
  "punkty": 2
}
```

### 3. Zadania prawda-falsz:

```
{
  "typ": "PRAWDA_FALSZ",
  "pytanie": "Bardzo proste pytanie wymagające odpowiedzi prawda albo fałsz",
  "odpowiedz": "true",
  "punkty": 1
}
```

### 4. Zadania z plikiem:

```
{
  "typ": "PLIK",
  "pytanie": "Pytanie wymagające wgrania pliku lub jego edycję i ponowne wgranie",
  "plik": "<base64_plik>",
  "punkty": 40
}
```

### Przykład

```
[
  { "typ": "OTWARTE", "pytanie": "Bardzo trudne pytanie wymagające rozbudowanej odpowiedzi", "punkty": 2 },
  { "typ": "PRAWDA_FALSZ", "pytanie": "Bardzo proste pytanie wymagające odpowiedzi prawda albo fałsz", "odpowiedz": "true", "punkty": 30 },
  { "typ": "PLIK", "pytanie": "Pytanie wymagające wgrania pliku lub jego edycję i ponowne wgranie", "plik": "Ym5Wc2JBPT0=", "punkty": 20 },
  { "typ": "ZAMKNIETE", "pytanie": "Pytanie testowe z czterema odpowiedziami do wyboru, gdzie dwie są prawdziwe", "odpowiedz": [ "odpowiedź1", "odpowiedź2", "odpowiedź3", "odpowiedź3" ], "poprawneOdp": [ 1, 3 ], "punkty": 1 }
]
```

### Opis

Klasa **ZadaniaController** odpowiada za obsługę end-pointów związanych z zarządzaniem zadaniami przedmiotów w systemie. Poniżej przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

### End-pointy

#### 1. Pobieranie listy wszystkich zadań

- Ścieżka: **/api/przedmiot/zadanie/all**
- Metoda: **GET**



- Parametry:
  - **kod** (wymagany): Kod przedmiotu zakodowany w base64.
  - **size** (opcjonalny): Liczba elementów na stronie.
  - **page** (opcjonalny): Numer strony (liczony od 0).
- Odpowiedź:
  - **200 OK** - sukces, zwraca listę zadań w formacie JSON.
  - **404 Not Found** - brak zadań.

```
GET /api/przedmiot/zadanie/all?kod=aGFwcHk=  
Authorization: Bearer <token>
```

## 2. Pobieranie pojedynczego zadania

- **Ścieżka:** `/api/przedmiot/zadanie/{id}`
- **Metoda:** `GET`
- **Parametry:**
  - **id** (ścieżka): Identyfikator zadania.
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca dane zadania w formacie JSON.
  - **404 Not Found** - zadanie o podanym identyfikatorze nie istnieje.

```
GET /api/przedmiot/zadanie/1  
Authorization: Bearer <token>
```

## 3. Pobieranie aktywnych zadań

Poniższe zapytanie zwraca zdania o określonym typie, które spełniają warunek: `NOW()` `BETWEEN z.data_pocz` `AND z.data_konc`. Można pobrać zadania aktywne dla danego użytkownika, w danym przedmiocie lub wszystkie aktywne.

- **Ścieżka:** `/api/przedmiot/zadanie/aktywne`
- **Metoda:** `GET`
- **Parametry:**
  - **login** (opcjonalny): login użytkownika zakodowany w base64
  - **kod** (opcjonalny): Kod przedmiotu zakodowany w base64.
  - **typ** (wymagany): Typ zadania (zadanie, test, egzamin) zakodowany w base64.
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca listę zadań w formacie JSON.
  - **404 Not Found** - brak zadań.

## 4. Usuwanie zadania

- **Ścieżka:** `/api/przedmiot/zadanie/{id}`
- **Metoda:** `DELETE`

- **Parametry:**
  - `id` (ścieżka): Identyfikator zadania.
- **Odpowiedź:**
  - `204 No Content` - sukces, zadanie zostało usunięte.
  - `404 Not Found` - zadanie o podanym identyfikatorze nie istnieje.

```
DELETE /api/przedmiot/zadanie/<nr_id>  
Authorization: Bearer <token>
```

## 5. Dodawanie nowego zadania

- **Ścieżka:** `/api/przedmiot/zadanie`
- **Metoda:** `POST`
- **Parametry:**
  - Ciało żądania zawiera dane nowego zadania w formacie JSON.
- **Odpowiedź:**
  - `201 Created` - sukces, zadanie zostało dodane, zwraca link do nowo utworzonego zadania.
  - `400 Bad Request` - błąd w danych wejściowych.

```
POST /api/przedmiot/zadanie  
Content-Type: application/json  
Authorization: Bearer <token>
```

```
{  
  "idPrzedmiotu": 4,  
  "typZadania": "EGZAMIN",  
  "opis": "<opis_zadania>",  
  "dataWstawienia": "2023-12-28T19:11:04",  
  "dataPoczatku": "2024-02-28T00:00:00",  
  "dataKonca": "2024-02-28T00:00:00",  
  "tresc": "<tresc_json>"  
}
```

## 6. Aktualizacja danych zadania

- **Ścieżka:** `/api/przedmiot/zadanie/{id}`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): Identyfikator zadania.
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON.
- **Odpowiedź:**
  - `200 OK` - sukces, zadanie zostało zaktualizowane, zwraca link do zaktualizowanego zadania.
  - `400 Bad Request` - błąd w danych wejściowych.

- **404 Not Found** - zadanie o podanym identyfikatorze nie istnieje.

```
PATCH /api/przedmiot/zadanie/1
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "dataPoczatku": "2023-02-01T00:00:00",
  "dataKonca": "2023-03-31T00:00:00",
  "tresc": "<tresc_json>"
}
```

## Odpowiedzi

### Rodzaje Odpowiedzi

Rodzaje odpowiedzi są analogiczne dla typów zadań. Pola zawierają jedynie odpowiedź oraz punktację. Odpowiedzią dla zadań zamkniętych jest lista numerów odpowiedzi.

### Opis

Klasa **OdpowiedzRepository** odpowiada za interakcję z bazą danych w kontekście odpowiedzi na zadania przedmiotów w systemie. Poniżej przedstawione są wszystkie dostępne metody wraz z ich opisem, parametrami i możliwymi odpowiedziami.

Podczas dodawania nowej odpowiedzi można ustawiać wartości wszystkich pól (**komentarz**, **ocena**, **dataOcenienia**) jednak zakłada się, że wstawianie nowej odpowiedzi wykonywane jest przez ucznia, który nie ocenia.

Podczas wstawiania i uaktualniania odpowiedzi następuje jej sprawdzenie dla zadań typu zamkniętego oraz prawda-fałsz. Ręcznie muszą zostać ocenione zadania otwarte.

### End-pointy

#### 1. Pobieranie listy wszystkich odpowiedzi

- **Ścieżka:** `/api/odpowiedzi/all`
- **Metoda:** `GET`
- **Parametry:**
  - **login** (parametr zapytania, opcjonalny): login ucznia w formacie Base64
  - **kod** (parametr zapytania, obowiązkowy): kod przedmiotu w formacie Base64
  - **size** (parametr zapytania, opcjonalny): liczba wyników na stronie
  - **page** (parametr zapytania, opcjonalny): numer strony, liczony od 0
- **Odpowiedź:**
  - **200 OK** - sukces, zwraca listę odpowiedzi w formacie JSON
  - **404 Not Found** - brak odpowiedzi

```
GET /api/odpowiedzi/all?requestParams=size;page;kod;login
Authorization: Bearer <token>
```

## 2. Pobieranie pojedynczej odpowiedzi

**Ścieżka:** `/api/przedmioty/zadania/odpowiedz/{id}` **Metoda:** GET **Parametry:** - `id` (ścieżka): Identyfikator odpowiedzi. **Odpowiedź:**

- 200 OK - sukces, zwraca dane pojedynczej odpowiedzi w formacie JSON
- 404 Not Found - odpowiedź o podanym identyfikatorze nie istnieje

```
GET /api/przedmioty/zadania/odpowiedz/1
Authorization: Bearer <token>
```

## 3. Dodawanie nowej odpowiedzi

**Ścieżka:** `/api/przedmioty/zadania/odpowiedz` **Metoda:** POST **Parametry:** Ciało żądania zawiera dane nowej odpowiedzi w formacie JSON. **Odpowiedź:** 201 Created - sukces, odpowiedź została dodana, zwraca identyfikator nowo utworzonej odpowiedzi 400 Bad Request - błąd w danych wejściowych

```
POST /api/przedmioty/zadania/odpowiedz
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "idZadania": 1,
  "idUcznia": 2,
  "tresc": "<tresc_json>"
}
```

## 4. Aktualizacja danych odpowiedzi

- **Ścieżka:** `/api/przedmioty/zadania/odpowiedz/{id}`
- **Metoda:** PATCH
- **Parametry:**
  - `id` (ścieżka): Identyfikator odpowiedzi.
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON.
- **Odpowiedź:**
  - 200 OK - sukces, odpowiedź została zaktualizowana
  - 400 Bad Request - błąd w danych wejściowych
  - 404 Not Found - odpowiedź o podanym identyfikatorze nie istnieje

```
PATCH /api/przedmioty/zadania/odpowiedz/1
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "tresc": "<tresc_json>",
  "komentarz": "Bardzo dobra praca!"
}
```

## 5. Usunięcie odpowiedzi

- **Ścieżka:** `/api/przedmiot/zadanie/odpowiedz/{id}`
- **Metoda:** `DELETE`
- **Parametry:**
  - `id` (ścieżka): Identyfikator zadania.
- **Odpowiedź:**
  - `204 No Content` - sukces, zadanie zostało usunięte.
  - `404 Not Found` - zadanie o podanym identyfikatorze nie istnieje.

```
DELETE /api/przedmiot/zadanie/odpowiedz/<nr_id>
Authorization: Bearer <token>
```

## 6. Wystawianie oceny

- **Ścieżka:** `/api/przedmioty/zadania/odpowiedz/{id}/ocen`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): Identyfikator odpowiedzi.
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON.
- **Odpowiedź:**
  - `200 OK` - sukces, odpowiedź została zaktualizowana
  - `400 Bad Request` - błąd w danych wejściowych

```
PATCH /api/przedmioty/zadania/odpowiedz/1/ocen
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "tresc": "<tresc_json>",
  "komentarz": "Bardzo dobra praca!",
}
```

```
"ocena": 5  
}
```

## Materialy

### Opis

Klasa `MaterialyController` obsługuje end-pointy związane z zarządzaniem materiałami przedmiotów w systemie. Poniżej przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

### End-pointy

#### 1. Pobieranie listy wszystkich materiałów

- **Ścieżka:** `/api/przedmiot/material/all`
- **Metoda:** `GET`
- **Parametry:**
  - `kod` (parametr zapytania, wymagany): Kod przedmiotu w base64.
  - `lp` (parametr zapytania, opcjonalny): Numer porządkowy materiału.
  - `size` (parametr zapytania, opcjonalny): Liczba wyników na stronie.
  - `page` (parametr zapytania, opcjonalny): Numer strony, liczony od 0.
- **Odpowiedź:**
  - `200 OK` - sukces, zwraca listę materiałów w formacie JSON
  - `404 Not Found` - brak materiałów

```
GET /api/przedmiot/material/all?kod=<kod>&lp=<lp>&size=<size>&page=<page>  
Authorization: Bearer <token>
```

#### 2. Pobieranie pojedynczego materiału

- **Ścieżka:** `/api/przedmiot/material/{id}`
- **Metoda:** `GET`
- **Parametry:**
  - `id` (ścieżka): Identyfikator materiału.
- **Odpowiedź:**
  - `200 OK` - sukces, zwraca dane pojedynczego materiału w formacie JSON
  - `404 Not Found` - materiał o podanym identyfikatorze nie istnieje

```
GET /api/przedmiot/material/1  
Authorization: Bearer <token>
```

#### 3. Usuwanie materiału

Po usunięciu materiału automatycznie zostaje poprawiona liczba porządkowa pozostałych materiałów,

- **Ścieżka:** `/api/przedmiot/materialy/{id}`
- **Metoda:** `DELETE`
- **Parametry:**
  - `id` (ścieżka): Identyfikator materiału.
- **Odpowiedź:**
  - `204 No Content` - sukces, materiał został usunięty
  - `404 Not Found` - materiał o podanym identyfikatorze nie istnieje

```
DELETE /api/przedmiot/material/1
Authorization: Bearer <token>
```

#### 4. Dodawanie nowego materiału

Automatycznie jest nadawany liczba porządkowa oraz ustawiana data wstawienia.

Można dodać zadanie do danego materiału przedmiotu. Wówczas należy najpierw utworzyć owe zadanie i przekazać jego `id` w tym zapytaniu w obiekcie `json`.

- **Ścieżka:** `/api/przedmiot/material`
- **Metoda:** `POST`
- **Parametry:**
  - Ciało żądania zawiera dane nowego materiału w formacie JSON
- **Odpowiedź:**
  - `201 Created` - sukces, materiał został dodany, zwraca link do nowo utworzonego materiału
  - `400 Bad Request` - błąd w danych wejściowych

```
POST /api/przedmiot/material
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "idPrzedmiotu": 1,
  "temat": "Nowy temat",
  "plik": "<Zawartość pliku zakodowana w base64>",
  "nazwaPliku": "nazwa_pliku.txt",
  "ext": "txt",
  "opis": "Opis materiału",
  "idZadania": 2,
  "widocznosc": 1
}
```

#### 5. Aktualizacja danych materiału

- **Ścieżka:** `/api/przedmiot/material/{id}`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): Identyfikator materiału.
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON
- **Odpowiedź:**
  - `200 OK` - sukces, materiał został zaktualizowany, zwraca link do zaktualizowanego materiału
  - `400 Bad Request` - błąd w danych wejściowych
  - `404 Not Found` - materiał o podanym identyfikatorze nie istnieje

```
PATCH /api/przedmiot/material/1
Content-Type: application/json
Authorization: Bearer <token>

json
{
  "temat": "Nowy temat"
}
```

## Powiadomienia

### Opis

Klasa `PowiadomieniaController` obsługuje end-pointy związane z zarządzaniem powiadomieniami użytkowników w systemie. Poniżej przedstawione są wszystkie dostępne end-pointy wraz z ich opisem, parametrami i możliwymi odpowiedziami.

Powiadomienia są dodawane automatycznie po stronie bazy danych za pomocą odpowiednich *triggerów*:

- `tai_materialy` - Trigger dodaje powiadomienie wszystkim uczestnikom przedmiotu, gdy zostanie dodany materiał do przedmiotu.
  - Treść: `'<p>Dostępny nowy materiał z przedmiotu \'<nazwa_przedmiotu\'!</p>'`
- `tbu_materialy` - Trigger dodaje powiadomienie wszystkim uczestnikom przedmiotu, gdy zostanie uwidoczniiony materiał.
  - Treść: `'<p>Dostępny nowy materiał z przedmiotu \'<nazwa_przedmiotu\'!</p>'`
- `tai_odpowiedzi_zadania` - Trigger dodaje nowe powiadomienie nauczycielowi, gdy uczeń doda odpowiedź do zadania.
  - Treść: `'<p>Użytkownik <imie i nazwisko> dodał nową odpowiedź do zadania z przedmiotu \'<nazwa_przedmiotu\'.</p>'`
- `tbu_odpowiedzi_zadania` - Trigger dodaje nowe powiadomienie gdy wystawiono ocenę uczniowi za zadanie.
  - Treść: `'<p>Dostałeś nową ocenę za zadanie z przedmiotu \'<nazwa_przedmiotu\'!</p>'`
- `tai_zadania` - Trigger dodaje powiadomienie wszystkim uczestnikom przedmiotu, gdy zostanie dodane nowe zadanie, które jest aktywne danego dnia.
  - Treść: `'<p>Dostępne nowe zadanie dla przedmiotu \'<nazwa_przedmiotu\'!</p><p>Zadanie będzie dostępne do dnia: <data_konca>.<p>'`



- **tbu\_uczen\_przedmiot** - Trigger dodaje nowe powiadomienie, gdy wystawiono ocenę uczniowi z przedmiotu.
  - Treść: '`<p>Dostałeś nową ocenę z przedmiotu \'<nazwa_przedmiotu\'!\</p>`'

Dla zadań utworzony jest także *event* (`e_zadania_aktywne`), który wysyła powiadomienia użytkownikom, zarejestrowanym do danego przedmiotu, jeżeli danego dnia zadanie się otwiera.

## End-pointy

### 1. Pobieranie listy wszystkich powiadomień dla użytkownika

- **Ścieżka:** `/api/powiadomienie/all`
- **Metoda:** `GET`
- **Parametry:**
  - `login` (parametr zapytania, wymagany): Login użytkownika w base64.
  - `size` (parametr zapytania, opcjonalny): Liczba wyników na stronie.
  - `page` (parametr zapytania, opcjonalny): Numer strony, liczony od 0.
- **Odpowiedź:**
  - `200 OK` - sukces, zwraca listę powiadomień w formacie JSON
  - `404 Not Found` - brak powiadomień

```
GET /api/powiadomienie/all?login=<login>&size=<size>&page=<page>
Authorization: Bearer <token>
```

### 2. Pobieranie pojedynczego powiadomienia

- **Ścieżka:** `/api/powiadomienie/{id}`
- **Metoda:** `GET`
- **Parametry:**
  - `id` (ścieżka): Identyfikator powiadomienia.
- **Odpowiedź:**
  - `200 OK` - sukces, zwraca dane pojedynczego powiadomienia w formacie JSON
  - `404 Not Found` - powiadomienie o podanym identyfikatorze nie istnieje

```
GET /api/powiadomienie/1
Authorization: Bearer <token>
```

### 3. Usuwanie powiadomienia

- **Ścieżka:** `/api/powiadomienie/{id}`
- **Metoda:** `DELETE`
- **Parametry:**
  - `id` (ścieżka): Identyfikator powiadomienia.
- **Odpowiedź:**
  - `204 No Content` - sukces, powiadomienie zostało usunięte

- **404 Not Found** - powiadomienie o podanym identyfikatorze nie istnieje

```
DELETE /api/powiadomienie/1
Authorization: Bearer <token>
```

#### 4. Aktualizacja flagi powiadomienia

- **Ścieżka:** `/api/powiadomienie/{id}`
- **Metoda:** `PATCH`
- **Parametry:**
  - `id` (ścieżka): Identyfikator powiadomienia.
  - Ciało żądania zawiera dane do aktualizacji w formacie JSON
- **Odpowiedź:**
  - **200 OK** - sukces, flaga powiadomienia została zaktualizowana, zwraca link do zaktualizowanego powiadomienia
  - **404 Not Found** - powiadomienie o podanym identyfikatorze nie istnieje

```
PATCH /api/powiadomienie/1
Content-Type: application/json
Authorization: Bearer <token>
```

```
{
  "flaga": "PRZECZYTANA"
}
```