Testy API Biblioteki - Lista 4 WWW

Michał Kallas

1 Wstęp

Testy zostały zaimplementowane przy użyciu frameworka testowego **Jest**, który jest popularnym narzędziem do testowania aplikacji napisanych w JavaScript. **Jest** zapewnia prostą konfigurację, wsparcie dla asynchronicznych testów oraz wbudowane mechanizmy do mockowania i asercji. Do zarządzania danymi testowymi wykorzystano **MongoDB Memory Server**, czyli lekką wersję bazy MongoDB działającą w pamięci RAM, co pozwala na szybkie i izolowane testy bez konieczności uruchamiania zewnętrznej instancji bazy danych. Przed każdym testem baza danych jest czyszczona, aby zapewnić niezależność przypadków testowych. Testy są w pełni zautomatyzowane i można je uruchomić za pomocą pojedynczej komendy w środowisku Node.js, np. `npm test`, co zapewnia łatwość użycia i potencjalnej integracji z procesami CI/CD.

2 Przypadki testowe dla zasobu Users

- **POST /api/users - Rejestracja użytkownika:**
  - Sprawdzenie, czy nowy użytkownik może się zarejestrować z poprawnymi danymi (status 201).
  - Sprawdzenie odrzucenia duplikatu nazwy użytkownika (status 409).
  - Sprawdzenie odrzucenia duplikatu adresu e-mail (status 409).
  - Sprawdzenie błędu dla nieprawidłowego formatu e-mail (status 400).
  - Sprawdzenie odrzucenia zbyt krótkiej nazwy użytkownika (status 400).
  - Sprawdzenie odrzucenia zbyt słabego hasła (status 400).
  - Sprawdzenie odrzucenia żądania z brakującymi polami (status 400).
- **GET /api/users - Lista użytkowników (tylko admin):**
  - Sprawdzenie, czy administrator otrzymuje listę użytkowników (status 200, brak haseł w odpowiedzi).
  - Sprawdzenie, że użytkownik bez uprawnień otrzymuje błąd 403.
  - Sprawdzenie odrzucenia nieautoryzowanego dostępu (status 401).

- **GET /api/users/:id - Pobieranie profilu użytkownika:**
  - Sprawdzenie, czy użytkownik może zobaczyć własny profil (status 200, brak hasła).
  - Sprawdzenie, że administrator może zobaczyć dowolny profil (status 200).
  - Sprawdzenie odrzucenia dostępu do cudzego profilu przez zwykłego użytkownika (status 403).
  - Sprawdzenie błędu 404 dla nieistniejącego użytkownika.
- **PUT /api/users/:id - Aktualizacja użytkownika:**
  - Sprawdzenie, czy użytkownik może zaktualizować własny profil (status 200).
  - Sprawdzenie, że administrator może zaktualizować dowolny profil (status 200).
  - Sprawdzenie odrzucenia próby aktualizacji cudzego profilu przez zwykłego użytkownika (status 403).
- **DELETE /api/users/:id - Usuwanie użytkownika (tylko admin):**
  - Sprawdzenie, czy administrator może usunąć użytkownika (status 204, użytkownik nie istnieje).
  - Sprawdzenie, że zwykły użytkownik nie może usunąć profilu (status 403).
  - Sprawdzenie błędu 404 dla nieistniejącego użytkownika.

3 Przypadki testowe dla zasobu Auth

- **POST /api/auth/login - Logowanie:**
  - Sprawdzenie logowania z poprawnymi danymi (status 200).
  - Sprawdzenie odrzucenia nieprawidłowego e-maila (status 401).
  - Sprawdzenie odrzucenia nieprawidłowego hasła (status 401).
  - Sprawdzenie odrzucenia brakujących danych logowania (status 401).
  - Sprawdzenie odrzucenia pustego ciała żądania (status 401).
- **Walidacja tokenu:**
  - Sprawdzenie poprawności struktury tokenu JWT (status 200).
  - Sprawdzenie odrzucenia nieprawidłowego tokenu (status 401).
  - Sprawdzenie odrzucenia żądania bez tokenu (status 401).

4 Przypadki testowe dla zasobu Books

- **POST /api/books - Dodawanie książki (tylko admin):**
  - Sprawdzenie, czy administrator może dodać książkę z poprawnymi danymi (status 201, dane zgodne z wysłanymi).
  - Sprawdzenie, że użytkownik bez uprawnień administratora otrzymuje błąd 403 (Brak uprawnień administratora).
  - Sprawdzenie odrzucenia żądania z brakującymi wymaganymi polami (tytuł, autor, ISBN) – status 400.
  - Sprawdzenie, że próba dodania książki z istniejącym ISBN zwraca błąd 409 (Konflikt).
  - Sprawdzenie odrzucenia nieautoryzowanego żądania – status 401.
- **GET /api/books - Lista książek:**
  - Sprawdzenie, czy lista książek jest zwracana bez autoryzacji (status 200, poprawna liczba książek i paginacja).
  - Sprawdzenie wsparcia dla paginacji (strona 1, limit 2, poprawne dane paginacji).
  - Sprawdzenie filtrowania książek po autorze (zwraca tylko książki danego autora).
  - Sprawdzenie filtrowania książek po roku publikacji (zwraca książki z danego roku).
  - Sprawdzenie sortowania książek po roku publikacji w porządku malejącym.
  - Sprawdzenie obsługi pustych wyników dla nieistniejącego autora (status 200, pusta lista).
- **GET /api/books/:id - Pobieranie pojedynczej książki:**
  - Sprawdzenie, czy pojedyncza książka jest zwracana poprawnie (status 200, zgodne dane).
  - Sprawdzenie błędu 404 dla nieistniejącej książki.
  - Sprawdzenie obsługi nieprawidłowego ObjectId (status 500).
- **PUT /api/books/:id - Aktualizacja książki (tylko admin):**
  - Sprawdzenie, czy administrator może zaktualizować książkę (status 200, zgodne dane).
  - Sprawdzenie, że użytkownik bez uprawnień administratora otrzymuje błąd 403.
  - Sprawdzenie błędu 404 dla nieistniejącej książki.
- **DELETE /api/books/:id - Usuwanie książki (tylko admin):**

- Sprawdzenie, czy administrator może usunąć książkę (status 204, książka nie istnieje w bazie).
- Sprawdzenie, że użytkownik bez uprawnień otrzymuje błąd 403.
- Sprawdzenie błędu 404 dla nieistniejącej książki.

5 Przypadki testowe dla zasobu Reviews

- **POST /api/reviews - Dodawanie recenzji:**
  - Sprawdzenie, czy uwierzytelniony użytkownik może dodać recenzję (status 201, zgodne dane).
  - Sprawdzenie odrzucenia nieautoryzowanej recenzji (status 401).
  - Sprawdzenie odrzucenia recenzji z brakującymi polami (status 400).
  - Sprawdzenie odrzucenia nieprawidłowych wartości oceny (poza zakresem 1–5, status 400).
  - Sprawdzenie błędu 404 dla nieistniejącej książki.
  - Sprawdzenie odrzucenia duplikatu recenzji tego samego użytkownika dla tej samej książki (status 409).
  - Sprawdzenie, że różni użytkownicy mogą oceniać tę samą książkę (status 201).
- **GET /api/reviews - Lista recenzji:**
  - Sprawdzenie, czy wszystkie recenzje są zwracane bez autoryzacji (status 200, posortowane po dacie).
  - Sprawdzenie filtrowania recenzji po `book_id` (tylko recenzje danej książki).
  - Sprawdzenie poprawności dołączonych danych użytkownika i książki w odpowiedzi.
  - Sprawdzenie obsługi pustych wyników dla nieistniejącej książki (status 200, pusta lista).
- **GET /api/reviews/:id - Pobieranie pojedynczej recenzji:**
  - Sprawdzenie, czy pojedyncza recenzja jest zwracana z poprawnymi danymi oraz uzupełnionymi informacjami o użytkowniku i książce (status 200).
  - Sprawdzenie błędu 404 dla nieistniejącej recenzji.
- **PUT /api/reviews/:id - Aktualizacja recenzji:**
  - Sprawdzenie, czy autor może zaktualizować własną recenzję (status 200).

  - Sprawdzenie, że administrator może zaktualizować dowolną recenzję (status 200).
  - Sprawdzenie odrzucenia aktualizacji przez innego użytkownika (status 403).
  - Sprawdzenie odrzucenia nieprawidłowej oceny (status 400).
  - Sprawdzenie błędu 404 dla nieistniejącej recenzji.
  - Sprawdzenie odrzucenia nieautoryzowanej aktualizacji (status 401).

- **DELETE /api/reviews/:id - Usuwanie recenzji:**
  - Sprawdzenie, czy autor może usunąć własną recenzję (status 204, recenzja nie istnieje).
  - Sprawdzenie, że administrator może usunąć dowolną recenzję (status 204).
  - Sprawdzenie odrzucenia usuwania przez innego użytkownika (status 403).
  - Sprawdzenie błędu 404 dla nieistniejącej recenzji.
  - Sprawdzenie odrzucenia nieautoryzowanego usuwania (status 401).

- **Przypadki brzegowe i obsługa błędów:**
  - Sprawdzenie obsługi nieprawidłowych ObjectId (status 500).
  - Sprawdzenie ścisłego sprawdzania granic oceny (0.9 i 5.1 odrzucone, 1–5 akceptowane).