

REST API w PHP i klient w JavaScript

Czas: 2 × 45 min

praca indywidualna

Cel: Zbudowanie w PHP interfejsu REST zwracającego JSON dla zasobu z bazy oraz klienta w JavaScript, który korzysta z niego przez `fetch`.

Po zajęciach potrafisz:

- projektuje adresy i metody HTTP dla zasobu (lista, element, dodanie, zmiana, usunięcie)
- zwraca odpowiednie kody stanu (200, 201, 204, 400, 404, 405, 422) i JSON z poprawnymi typami
- odczytuje treść żądania JSON (`php://input`) i waliduje dane
- testuje API poleceniem `curl` lub w kliencie HTTP
- pisze klienta `fetch` z obsługą stanu ładowania i błędów walidacji

Sprzęt i materiały

- komputer z Windows 11, XAMPP, przeglądarka z narzędziami F12
- `r2_17_katalog.sql` i folder `r2_18_start` z Pliki_do_zadan

Oprogramowanie

- **XAMPP**
[apachefriends.org/download.html](https://www.apachefriends.org/download.html)
- **curl** — wbudowany w Windows 11 jako `curl.exe` (w PowerShell pisz `curl.exe`, nie `curl`)
curl.se/windows
- **Postman lub rozszerzenie Thunder Client do VS Code (opcjonalnie)** — Thunder Client: <https://www.thunderclient.com/>
postman.com/downloads
- **Visual Studio Code**
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie z wykładu 2.13 metody HTTP i kody odpowiedzi oraz `fetch` z wykładu 2.03.

Bezpieczeństwo

- API zmieniające dane bez logowania jest dopuszczalne tylko na `localhost` w ćwiczeniu — w prawdziwej aplikacji wymaga uwierzytelnienia (zadanie 32).
- Nie zwracaj w odpowiedziach komunikatów błędów bazy — tylko ogólny komunikat i kod 500; szczegóły do logu.
- Dane z API wstawiaj do strony przez `textContent`, nie `innerHTML`.

Przebieg zadania

1 Projekt API i odczyt danych

Skopiuj `r2_18_start` do `htdocs\katalog_api` i upewnij się, że baza `katalog` istnieje (zadanie 17).

Docelowe zasoby:

GET	<code>api/filmy</code>	lista (opcjonalnie <code>?gatunek=komedia</code>)	200
GET	<code>api/filmy/{id}</code>	jeden film	200 / 404
POST	<code>api/filmy</code>	dodanie (JSON)	201 + Location / 400 / 422
PUT	<code>api/filmy/{id}</code>	podmiana całego filmu	200 / 404 / 422
PATCH	<code>api/filmy/{id}</code>	zmiana wybranych pól	200 / 404 / 422
DELETE	<code>api/filmy/{id}</code>	usunięcie	204 / 404
	inna metoda		405 + nagłówek Allow

1. Utwórz `api/.htaccess` kierujący żądania do `api/index.php` (jak w zadaniu 17). W `api/index.php` ustal ścieżkę zasobu (np. `/filmy/3`) i metodę; dopisz funkcje `odpowiedz(int $kod, mixed $dane)` (kod + `json_encode` z `JSON_UNESCAPED_UNICODE`) i `blad(int $kod, string $komunikat)`.
2. Zaimplementuj `GET /filmy` (z filtrem `?gatunek=` przez zapytanie przygotowane) i `GET /filmy/{id}` (404 z JSON `{"blad": ...}`, gdy brak). Funkcją `film_json()` zamień typy: `id` i `rok` na liczby całkowite, `ocena` na liczbę lub `null`.
3. Otwórz `http://localhost/katalog_api/api/filmy` w przeglądarce, a potem w konsoli: `curl.exe -i http://localhost/katalog_api/api/filmy/3`. Zanotuj nagłówki `Content-Type` i kod odpowiedzi.

2 Zapis danych: POST, PUT, PATCH, DELETE

Treść JSON odczytasz przez `json_decode(file_get_contents('php://input'), true)`; jeśli wynik nie jest tablicą → 400.

```
curl.exe -i -X POST -H "Content-Type: application/json" -d @film.json
http://localhost/katalog_api/api/filmy
curl.exe -i -X PATCH -H "Content-Type: application/json" -d @zmiana.json
http://localhost/katalog_api/api/filmy/9
curl.exe -i -X DELETE http://localhost/katalog_api/api/filmy/9
```

1. Napisz `waliduj(array $f): array` zwracającą błędy w postaci `pole => komunikat` (tytuł 1–120, reżyser 1–80, rok — liczba całkowita 1895–2030, gatunek z listy, ocena 1–10 lub `null`). Błędy → 422 z `{"blad": ..., "bledy": {...}}`.
2. POST: zapis, kod 201, nagłówek `Location` z adresem nowego filmu i JSON utworzonego rekordu. PUT: wymaga wszystkich pól. PATCH: połącz dane z bazy z przesłanymi (`array_merge`) i zwaliduj całość. DELETE: 204 bez treści.
3. Dla innych metod zwróć 405 z nagłówkiem `Allow`. Błędy PDO przechwyc (`try / catch`) i zwróć 500 z ogólnym komunikatem.
4. Przetestuj curl (ramka) wszystkie przypadki z tabeli zasobów, także błędne: zły JSON, puste pola, nieistniejące id, `DELETE` na `/filmy`. Wyniki wpisz do tabeli w sprawozdaniu.

3 Klient w JavaScript

Plik `index.html` jest gotowy (tabela `#lista`, pole `#stan`, filtr `#filtr`, formularz `#formularz`, lista `#bledy`). Kod pisz w `app.js` (moduł).

1. Funkcja `async wczytaj()`: pokaż „Ładowanie...”, pobierz listę (z filtrem, `encodeURIComponent`), sprawdź `odp.ok`, zbuduj wiersze tabeli przez `createElement` i `textContent`, pokaż liczbę filmów; błąd sieci lub HTTP — komunikat w `#stan`.
2. Obsłuż formularz: `preventDefault()`, zbuduj obiekt z `FormData` (rok jako liczba, pusta ocena jako `null`), wyślij `POST` z nagłówkiem `Content-Type: application/json`. Kod 201 → wyczyść formularz i odśwież listę; 422 → wypisz błędy pól w `#bledy`.
3. Dodaj przycisk „Usuń” w każdym wierszu: `confirm()`, żądanie `DELETE`, przy 204 odśwież listę. Zmiana filtra odświeża listę.
4. W F12 → Network obejrzyj żądania klienta (metoda, kod, nagłówki, Payload, Response). Zrób zrzuty: aplikacja, błędy walidacji, karta Network z żądaniem POST. Spakuj projekt jako `nazwisko_imie_zad18.zip` — użyjesz go w zadaniu 23.

W sprawozdaniu umieść

- ☐ tabela testów API: metoda, adres, dane, oczekiwany i otrzymany kod
- ☐ fragmenty kodu: rozpoznanie trasy, POST z walidacją, PATCH, funkcja `wczytaj()`
- ☐ zrzuty: odpowiedź curl z nagłówkami dla 201, aplikacja klienta, błędy 422, karta Network
- ☐ archiwum `nazwisko_imie_zad18.zip` i odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się PUT od PATCH? Który z nich wykonany dwa razy daje ten sam wynik?
2. Dlaczego po dodaniu zasobu zwracamy 201 i nagłówek `Location`, a po usunięciu 204?
3. Kiedy zwrócić 400, a kiedy 422?
4. Dlaczego `fetch` nie zgłasza wyjątku przy odpowiedzi 404 i jak to obsłużyć?
5. Po co w API nagłówki CORS i dlaczego klient z tego samego adresu ich nie potrzebuje?

Kryteria oceny

Kryterium	Punkty
GET listy (z filtrem) i elementu, typy JSON, 404 z komunikatem	4
POST z walidacją: 201 + Location, 400, 422 z błędami pól	4
PUT, PATCH, DELETE (204), 405 + Allow, obsługa błędów 500	4
Tabela testów curl z kodami	2
Klient: lista z ładowaniem i błędem, filtr, dodawanie z błędami 422, usuwanie	4
Zrzuty, archiwum, odpowiedzi na pytania	2
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).