

# Laravel — formularze, walidacja i CRUD

Czas: 2 × 45 min

praca indywidualna

**Cel:** Rozbudowa aplikacji Laravel o pełny CRUD z kontrolerem zasobów, walidacją danych, komunikatami błędów i paginacją oraz porównanie z czystym PHP.

## Po zajęciach potrafisz:

- tworzy kontroler zasobów i rejestruje 7 tras jednym wywołaniem `Route::resource`
- buduje formularze z `@csrf` i `@method('PUT'/'DELETE')`
- waliduje dane `$request->validate()` i wyświetla błędy przy polach (`@error`, `old()`)
- zapisuje, zmienia i usuwa rekordy modelem Eloquent z komunikatem flash
- dzieli listę na strony (`paginate`) i porównuje nakład pracy z aplikacją w czystym PHP

## Sprzęt i materiały

- komputer z Windows 11 i środowiskiem z zadania 20 (Herd lub PHP + Composer)
- projekt `filmoteka` z zadania 20

## Oprogramowanie

- **Laravel Herd dla Windows lub PHP + Composer** — jak w zadaniu 20  
[herd.laravel.com/windows](https://herd.laravel.com/windows)
- **Dokumentacja Laravel — walidacja** — lista reguł: Available Validation Rules  
[laravel.com/docs/13.x/validation](https://laravel.com/docs/13.x/validation)
- **Visual Studio Code**  
[code.visualstudio.com](https://code.visualstudio.com)

## Przygotowanie przed zajęciami

- Uruchom projekt z zadania 20 i sprawdź, że lista filmów działa.

## Bezpieczeństwo

- Nie wyłączaj ochrony CSRF — każdy formularz POST/PUT/DELETE musi zawierać `@csrf` (bez tokenu Laravel zwraca błąd 419).
- Do `create()` i `update()` przekazuj tylko dane zwrócone przez `validate()` — model z `$fillable` przyjmie wyłącznie dozwolone pola.

## Przebieg zadania

### 1 Kontroler zasobów i trasy

W `routes/web.php` zastąp dwie trasy filmów jedną:

```
Route::resource('filmy', FilmController::class)->parameters(['filmy' => 'film']);
```

1. Uruchom `php artisan route:list --name=filmy` i przepisz do sprawozdania 7 tras (metoda, adres, nazwa, akcja). Po co `parameters(['filmy' => 'film'])`?
2. Dopisz w kontrolerze brakujące metody: `create`, `store`, `edit`, `update`, `destroy` (możesz porównać z szablonem z `php artisan make:controller TestController --resource --model=Film` i usunąć potem plik testowy).

## 2 Dodawanie z walidacją

Reguły zapisz w prywatnej metodzie `reguly()`, aby użyć ich w `store` i `update`.

```
'tytul' => ['required', 'string', 'max:120'],
'rezyser' => ['required', 'string', 'max:80'],
'rok' => ['required', 'integer', 'between:1895,2030'],
'gatunek' => ['required', Rule::in(Film::GATUNKI)],
'ocena' => ['nullable', 'numeric', 'between:1,10'],
```

1. Utwórz częściowy widok `filmy/_form.blade.php` z polami formularza: `@csrf`, wartości `old('pole', $film->pole)`, pod każdym polem `@error('pole') <span class="blad">{{ $message }}</span> @enderror`, lista gatunków z `@selected`.
2. Widok `filmy/create.blade.php` dołącza formularz (`@include`) z akcją `route('filmy.store')`; nad formularzem pokaż liczbę błędów (`$errors->count()`). Metoda `create()` przekazuje pusty obiekt `new Film()` i listę gatunków.
3. `store()`: `$dane = $request->validate($this->reguly());`, `Film::create($dane)`, przekierowanie do szczegółów z `->with('status', 'Dodano film.')`. W widokach `index` i `show` wyświetl `session('status')`.
4. Wyślij pusty formularz i formularz z rokiem 3000 — zanotuj komunikaty. Ustaw w `.env` `APP_LOCALE=pl` i utwórz `lang/pl/validation.php` z polskimi komunikatami dla użytych reguł i nazwami pól w `attributes` (pełne tłumaczenia: pakiet `laravel-lang/lang`).

## 3 Edycja, usuwanie i paginacja

1. Widok `filmy/edit.blade.php` z tym samym `_form`, akcją `route('filmy.update', $film)` i `@method('PUT')`. `update()` waliduje i wywołuje `$film->update(...)`.
2. W liście dodaj przy każdym filmie „Edytuj” i formularz usuwania (`@csrf`, `@method('DELETE')`, potwierdzenie `confirm`). `destroy()` usuwa rekord i wraca do listy z komunikatem zawierającym tytuł.
3. Zamień `get()` w `index()` na `paginate(5)->withQueryString()`, pod tabelą wypisz `{{ $filmy->links('pagination::default') }}` (style klasy `.pagination` w CSS), a liczbę filmów pobierz `$filmy->total()`. Sprawdź filtr gatunku na stronie 2.
4. Test 419: w F12 → Elements usuń z formularza ukryte pole `_token` i wyślij formularz. Zanotuj wynik.

## 4 Porównanie z czystym PHP

1. Policz niepuste linie kodu odpowiadających sobie części w zadaniu 17 (lub 9) i w Laravel (kontroler, model, trasy, widoki filmów, układ). Uzupełnij tabelę porównania w sprawozdaniu: walidacja, ochrona CSRF, 404 dla brakującego rekordu, komunikaty flash, paginacja, escapowanie — jak zrealizowane w czystym PHP, a jak w Laravel.
2. Zrób zrzuty: lista z paginacją i komunikatem, formularz z błędami (po polsku), edycja, strona 419. Spakuj projekt bez `vendor/` i `node_modules/` jako `nazwisko_imie_zad21.zip`.

## W sprawozdaniu umieść

- ☐ tabela 7 tras resource (metoda, adres, nazwa, akcja)
- ☐ fragmenty kodu: reguły walidacji, `store`, `update`, formularz usuwania
- ☐ tabela porównania czysty PHP / Laravel z liczbą linii
- ☐ zrzuty: lista z paginacją, błędy walidacji, edycja, 419; archiwum `nazwisko_imie_zad21.zip`
- ☐ odpowiedzi na pytania kontrolne

## Pytania kontrolne

1. Dlaczego formularz HTML wysyła PUT i DELETE jako POST z polem `_method`?

2. Co dzieje się po nieudanej walidacji w `$request->validate()` (przekierowanie, sesja, `old()`, `$errors`)?
3. Przed czym chroni `$fillable` i dlaczego do `create()` przekazujemy wynik `validate()`, a nie `$request->all()`?
4. Jakie są zalety i wady frameworka w porównaniu z własnym kodem z zadania 17?

## Kryteria oceny

| Kryterium                                                                                                                          | Punkty    |
|------------------------------------------------------------------------------------------------------------------------------------|-----------|
| <code>Route::resource</code> z parametrem, tabela tras                                                                             | 2         |
| Formularz <code>_form</code> z <code>@csrf</code> , <code>old()</code> , <code>@error</code> ; dodawanie z walidacją 5 pól i flash | 5         |
| Polskie komunikaty walidacji ( <code>APP_LOCALE</code> , <code>lang/pl</code> )                                                    | 2         |
| Edycja (PUT) i usuwanie (DELETE) z potwierdzeniem i komunikatem                                                                    | 4         |
| Paginacja z zachowaniem filtra, test 419                                                                                           | 3         |
| Porównanie z czystym PHP, zrzuty, archiwum, pytania                                                                                | 4         |
| <b>Razem</b>                                                                                                                       | <b>20</b> |

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