

Projekt końcowy cz. 3 — frontend i integracja

Czas: 2 × 45 min

praca w zespołach 2-3-osobowych

Cel: Zbudowanie responsywnego i dostępnego interfejsu aplikacji zespołowej (Bootstrap lub React), połączenie go z backendem przez endpoint JSON i fetch oraz poprawa kodu po przeglądzie.

Po zajęciach potrafisz:

- buduje responsywny interfejs z komponentów Bootstrap (lub React) z aktywną pozycją menu
- projektuje endpoint JSON z walidacją parametrów i właściwymi kodami odpowiedzi
- pobiera dane przez fetch z obsługą ładowania, błędów i przerywania żądania oraz bezpiecznie wstawia je do DOM
- stosuje stopniowe ulepszanie: strona działa bez JS, a skrypt dodaje wygodę
- przeprowadza przegląd kodu wg listy kontrolnej i wprowadza poprawki

Sprzęt i materiały

- komputer z XAMPP, VS Code, Git, Node.js LTS (przy wariancie React); repozytorium z zadań 31-32
- folder `r2_31_projekt` z Pliki_do_zadan (lista przeglądu kodu)

Oprogramowanie

- **Bootstrap 5.3** — wariant podstawowy; wersja React — Vite jak w zadaniach 22-23 getbootstrap.com/docs/5.3/getting-started/introduction
- **Narzędzia F12 (Chrome/Edge)** — Network (fetch), tryb urządzeń, Lighthouse developer.chrome.com/docs/devtools
- **Thunder Client / curl** — test endpointu JSON (jak w zadaniu 18) thunderclient.com

Przygotowanie przed zajęciami

- Scalcie wszystkie gałęzie backendu (tag v0.2-backend) i uzupełnijcie Issues kamienia „cz. 3 frontend”.
- Przypomnij sobie z zadań 4 i 18 fetch z obsługą błędów, a z zadania 25 komponenty Bootstrapa.

Bezpieczeństwo

- Dane z API wstawiaj przez `textContent` lub `createElement` — nigdy przez `innerHTML` z danymi od użytkowników.
- Pliki z CDN dołączaj z atrybutem `integrity` i dopisz domenę CDN do nagłówka `Content-Security-Policy` zamiast go wyłączać.

Przebieg zadania

1 Układ i komponenty

Wariant React: ten sam zakres, komponenty zamiast szablonów PHP; endpointy JSON z kroku 2 są wtedy jedynym źródłem danych.

1. Dołącz Bootstrap (CDN z integrity albo npm) we wspólnym nagłówku. Zbuduj navbar zwijany na telefonie z pozycjami zależnymi od roli; bieżąca pozycja ma klasę `active` i `aria-current="page"`.
2. Przebuduj listę głównego zasobu na siatkę kart (1/2/3 kolumny), formularze na `form-control` / `form-select` z etykietami i komunikatami walidacji przy polach (`is-invalid`, `invalid-feedback`), tabele panelu na `table` w `table-responsive`.
3. Komunikaty flash wyświetl jako `alert` z `role="status"`; dodaj odnośnik „Przejdź do treści” (`visually-hidden-focusable`) i własny kolor marki przez zmienne `--bs-btn-*`. Sprawdź nagłówek CSP — Bootstrap z CDN ma się ładować bez błędów w konsoli.

2 Endpoint JSON

Przykład: `GET api/sprzet.php?q=&kat=&dzien=2026-11-04` zwraca listę sprzętu z polem `wolny` dla wybranego dnia.

```
{
  "dzien": "2026-11-04",
  "liczba": 3,
  "sprzet": [ { "id": 1, "nazwa": "...", "kategoria": "...", "stan": "sprawny", "wolny": false },
  ... ]
}
```

1. Napisz endpoint zwracający dane potrzebne interfejsowi: nagłówek `Content-Type: application/json; charset=utf-8`, `json_encode` z `JSON_UNESCAPED_UNICODE`, typy liczbowe i logiczne (nie same teksty).
2. Waliduj parametry (np. format daty, długość frazy) — błąd zwraca 422 z polem `bład`; inna metoda niż GET → 405 z nagłówkiem Allow. Endpoint korzysta z tych samych zapytań przygotowanych i reguły biznesowej co backend.
3. Przetestuj endpoint w Thunder Client lub curl: poprawne zapytanie, zły parametr, zła metoda. Zapisz kody i fragmenty odpowiedzi.

3 Interakcje JS i fetch

Stopniowe ulepszanie: formularz filtrów działa także bez JS (GET), a skrypt przejmuje go i odświeża tylko listę.

1. W pliku JS przechwyc `submit` i `change` formularza filtrów, pobierz dane z endpointu (`URLSearchParams` z `FormData`), pokaż stan „Ładowanie...” w elemencie `aria-live="polite"`, a przy błędzie komunikat z treścią z pola `bład`.
2. Zbuduj karty funkcją tworzącą elementy (`createElement`, `textContent`); dodaj plakietki stanu (np. wolny/zajęty w wybranym dniu) i licznik wyników. Wyszukiwanie wywołuj z opóźnieniem 300 ms po wpisaniu (debounce), poprzednie żądanie przerywaj `AbortController`.
3. Aktualizuj adres strony `history.replaceState`, aby filtry można było skopiować w linku. Sprawdź w F12 → Network, że przy pisaniu nie powstają dziesiątki żądań.
4. Dodaj drobne interakcje: potwierdzenie przed usunięciem (formularze z klasą `potwierdz`) i kontrolę dat w formularzu (`setCustomValidity`, data „do” nie wcześniejsza niż „od”). Wyłącz JS w przeglądarce i sprawdź, że strona nadal działa.

4 Responsywność, dostępność i przegląd kodu

1. Sprawdź wszystkie ekrany przy 360, 768 i 1280 px oraz obsługę klawiaturą (Tab, Enter, Esc w menu). Uruchom Lighthouse Accessibility dla 2 stron i popraw uwagi (cel ≥ 90).
2. Każda osoba otwiera pull request ze swoją częścią; recenzent przechodzi `lista_przeglądu_kodu.txt` i zostawia min. 3 komentarze do linii. Autor poprawia i odpowiada na komentarze, potem scalenie.
3. Oznacz wersję tagiem `v0.3-frontend` i zrób zrzuty głównych ekranów (telefon i komputer) do folderu `docs/`.

W sprawozdaniu umieść

- ☐ zrzuty głównych ekranów przy 360 i 1280 px, w tym rozwinięte menu
- ☐ opis endpointu (metoda, adres, parametry, przykładowa odpowiedź, kody błędów) i wyniki testu w Thunder Client/curl
- ☐ fragment kodu fetch z obsługą błędów i budowaniem elementów
- ☐ wynik Lighthouse Accessibility dla 2 stron i lista poprawek
- ☐ odnośniki do PR z komentarzami przeglądu i wprowadzonymi poprawkami, odpowiedzi na pytania

Pytania kontrolne

1. Na czym polega stopniowe ulepszanie i co zyskuje użytkownik, gdy JS się nie ładuje?
2. Dlaczego dane z serwera wstawiasz przez `textContent`, a nie `innerHTML`?
3. Po co `debounce` i `AbortController` przy wyszukiwaniu na żywo?
4. Dlaczego endpoint zwraca 422, a nie 200 z pustą listą, gdy data ma zły format?

Kryteria oceny

Kryterium	Punkty
Responsywny układ Bootstrap/React: navbar z aktywną pozycją, karty, formularze z walidacją, tabele, alerty	4
Endpoint JSON: nagłówki, typy danych, walidacja (422), metoda (405), test	4
fetch: ładowanie, błąd, AbortController, debounce, bezpieczne budowanie DOM, <code>replaceState</code>	4
Działanie bez JS, potwierdzenie usuwania, kontrola dat	2
Dostępność (Lighthouse ≥ 90 , klawiatura) i przegląd kodu z poprawkami w PR	4
Odpowiedzi na pytania kontrolne (indywidualnie)	2
Razem	20

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