

Projekt końcowy cz. 1 — specyfikacja, baza i repozytorium

Czas: 2 × 45 min

praca w zespołach 2–3-osobowych

Cel: Zaplanowanie zespołowej aplikacji webowej: wymagania i przypadki użycia, model danych (ERD i skrypt SQL z danymi testowymi), makiety oraz repozytorium z podziałem zadań.

Po zajęciach potrafisz:

- formułuje wymagania funkcjonalne i нефункционалне oraz ustala ich priorytety metodą MoSCoW
- opisuje przypadki użycia ze scenariuszem głównym i alternatywnym
- projektuje bazę z kluczami obcymi i ograniczeniami oraz pisze skrypt SQL z danymi testowymi
- przygotowuje makiety ekranów dla telefonu i komputera
- zakłada repozytorium zespołu z Issues, kamieniami milowymi i podziałem pracy

Sprzęt i materiały

- komputer z XAMPP, VS Code i Git; konto GitHub każdej osoby
- folder `r2_31_projekt` z Pliki_do_zadan (tematy.txt, szablon_specyfikacji.md i listy kontrolne)

Oprogramowanie

- **draw.io (diagrams.net)** — ERD i makiety; alternatywa: phpMyAdmin → Projektant app.diagrams.net
- **Penpot (opcjonalnie)** — darmowe makiety w przeglądarce penpot.app
- **XAMPP / phpMyAdmin** — test skryptu SQL apachefriends.org/download.html
- **GitHub** — Issues, Milestones, Projects github.com

Przygotowanie przed zajęciami

- W zespole wybierzcie 2 tematy z pliku `tematy.txt` (kolejność preferencji) — nauczyciel przydziela tematy bez powtórzeń.
- Przypomnij sobie pracę z gałęziami i pull requestami (zadanie 24) oraz relacje i klucze obce (przedmiot bazy danych).

Bezpieczeństwo

- Dane testowe mają być fikcyjne — nie wpisuj prawdziwych danych kolegów (imion z nazwiskami, telefonów, e-maili).
- Do repozytorium nie dodawaj haseł; w danych testowych zapisuj tylko skróty haseł z `password_hash`.

Przebieg zadania

1 Temat i wymagania

Wymagania minimalne (tematy.txt): min. 3 powiązane tabele z tabelą użytkowników, 2 role, CRUD głównego zasobu, co najmniej jedna reguła biznesowa, jeden endpoint JSON, responsywny interfejs.

1. Skopiuj `szablon_specyfikacji.md` jako `docs/specyfikacja.md`. Opisz cel aplikacji i aktorów (role) z ich uprawnieniami.
2. Spisz min. 8 wymagań funkcjonalnych z priorytetami MoSCoW (co najmniej 5 Must) i min. 4 wymagania niefunkcjonalne (środowisko, responsywność, dostępność, bezpieczeństwo).
3. Opisz min. 3 przypadki użycia: aktor, warunek wstępny, kroki scenariusza głównego i co najmniej jeden scenariusz alternatywny (np. błąd walidacji, kolizja terminu).

2 Model danych

1. Narysuj ERD w draw.io: encje, klucze główne i obce, typy relacji (1-N, N-M rozbite tabelą pośrednią). Zapisz jako `docs/erd.png`.
2. Napisz skrypt `sql/baza.sql`: SET NAMES utf8mb4, CREATE DATABASE z utf8mb4_polish_ci, tabele InnoDB z PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, ENUM lub CHECK tam, gdzie mają sens, indeks pod najczęstsze wyszukiwanie.
3. Dodaj dane testowe: min. 2 konta (po jednym na rolę, hasła jako skróty z `php -r "echo password_hash('...', PASSWORD_DEFAULT);"`) i min. 8 rekordów głównego zasobu. Uruchom skrypt w phpMyAdmin bez błędów.
4. Napisz i sprawdź 3 zapytania, których aplikacja będzie potrzebować (np. lista z JOIN, sprawdzenie reguły biznesowej, podsumowanie dla administratora). Zapisz je w `sql/zapytania.sql` z wynikami w komentarzach.

3 Makiety

Makieta to szkic układu (prostokąty, podpisy), nie gotowa grafika.

1. Przygotuj makiety min. 3 ekranów: lista głównego zasobu, formularz (dodawanie / rezerwacja), panel administratora — każdy w wersji telefon (360 px) i komputer.
2. Zaznacz na makietach elementy nawigacji i komunikaty (błąd walidacji, pusta lista). Zapisz w `docs/makiety/`.

4 Repozytorium i plan pracy

1. Utwórz repozytorium zespołu, dodaj współpracowników, `.gitignore` (np. `.env`, `node_modules/`, `vendor/`), `.env.example` i strukturę katalogów `public/`, `src/`, `sql/`, `docs/`.
2. Utwórz kamienie milowe (Issues → Milestones) „cz. 2 backend”, „cz. 3 frontend”, „cz. 4 wdrożenie” i min. 10 Issues z wymagań Must/Should, przypisanych do osób i kamieni milowych. Opcjonalnie: tablica Projects (To do / In progress / Done).
3. Uzupełnij w specyfikacji tabelę podziału pracy. Każda osoba wykonuje min. 1 commit przez pull request z przeglądem drugiej osoby.
4. Przedstaw nauczycielowi specyfikację i ERD (2 min na zespół) i zapisz uwagi jako Issues.

W sprawozdaniu umieść

- ☐ adres repozytorium i skład zespołu z rolami
- ☐ specyfikacja: wymagania MoSCoW, wymagania niefunkcjonalne, 3 przypadki użycia
- ☐ ERD, skrypt baza.sql (działa bez błędów) i 3 zapytania z wynikami
- ☐ makiety 3 ekranów (telefon i komputer), zrzut listy Issues z kamieniami milowymi
- ☐ odpowiedzi na pytania (indywidualnie)

Pytania kontrolne

1. Czym różni się wymaganie funkcjonalne od нефункционального? Podaj przykład z waszego projektu.
2. Jak zapisaliście w bazie relację N–M (jeśli występuje) lub dlaczego jej nie ma?
3. Jaka reguła biznesowa jest w waszej aplikacji i jak ją sprawdzicie w SQL?
4. Dlaczego warto zrobić makiety przed pisanem kodu?

Kryteria oceny

Kryterium	Punkty
Wymagania funkcjonalne (min. 8, MoSCoW) i нефункциональные (min. 4)	3
3 przypadki użycia ze scenariuszem głównym i alternatywnym	3
ERD zgodny ze skryptem SQL; klucze, ograniczenia, indeks	4
Skrypt SQL z danymi testowymi (hasła haszowane) i 3 zapytania z wynikami	3
Makiety 3 ekranów w dwóch wersjach	2
Repozytorium: struktura, .gitignore, Milestones, min. 10 Issues, PR każdej osoby	3
Odpowiedzi na pytania kontrolne (indywidualnie)	2
Razem	20

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