Politechnika
Śląska

PROJEKT INŻYNIERSKI

EduPHP – Interaktywna platforma webowa do nauki języka PHP

Anastasiia PASHKO
Nr albumu: 308870

Kierunek: Informatyka
Specjalność: Baza danych

PROWADZĄCY PRACĘ
Doktor inżynier Jacek Józef Szedel
KATEDRA Algorytmiki i Oprogramowania
Wydział Automatyki, Elektroniki i Informatyki

OPIEKUN, PROMOTOR POMOCNICZY
brak promotowa pomocniczego / opiekuna

Gliwice 2026

Tytuł pracy

EduPHP – Interaktywna platforma webowa do nauki języka PHP

Streszczenie

Nauka programowania w języku PHP, szczególnie w kontekście aplikacji webowych współpracujących z bazą danych, wiąże się z koniecznością konfiguracji lokalnego środowiska oraz ryzykiem uszkodzenia danych podczas eksperymentowania. Istniejące platformy edukacyjne często oferują jedynie materiały edukacyjne, proste ćwiczenia składniowe lub ogólne środowiska wykonawcze bez bezpiecznej izolacji operacji wykonywanych na bazach danych. Celem niniejszej pracy było stworzenie kompleksowej platformy internetowej, która eliminuje te bariery, oferując gotowe, bezpieczne środowisko do praktycznej nauki PHP. W ramach projektu zaprojektowano i zaimplementowano system oparty na architekturze klient-serwer z wykorzystaniem technologii Spring Boot dla warstwy backend, JavaScript dla strony frontend oraz bazy danych MySQL. Kluczowym i innowacyjnym elementem pracy jest mechanizm odizolowanego środowiska wykonawczego, który umożliwia bezpieczne uruchamianie dowolnego kodu PHP wprowadzonego przez użytkownika. Mechanizm ten automatycznie separuje dane między użytkownikami, nakłada limity czasu i zasobów oraz blokuje niebezpieczne funkcje systemowe. Powstała platforma EduPHP oferuje zestaw funkcjonalności edukacyjnych: zarządzanie kursami teoretycznymi i zadaniami praktycznymi, interaktywną konsolę kodu, system śledzenia postępów ze statystykami oraz forum zgłoszeń. Rezultatem pracy jest w pełni funkcjonalna aplikacja webowa, która stanowi bezpieczne i efektywne narzędzie do samodzielnej nauki programowania, realizując założenia projektu w zakresie funkcjonalności, użyteczności i przede wszystkim – bezpieczeństwa.

Słowa kluczowe

PHP, Spring Boot, JavaScript, REST API, MySQL

Thesis title

EduPHP – Interactive web platform for learning PHP language

Abstract

Learning PHP programming, especially in the context of web applications that interact with databases, requires configuring a local environment and carries the risk of data corruption during experimentation. Existing education platforms often offer only educational materials, simple syntax exercises, or general execution environments without

secure isolation of database operations. The aim of this work was to create a comprehensive online platform that eliminates these barriers by offering a ready-made, secure environment for practical PHP learning. The project resulted in a system based on a client-server architecture, designed and implemented with Spring Boot for the backend layer, JavaScript for the frontend, and a MySQL database. A key, innovative element of the work is the isolated execution environment mechanism, which enables user-entered PHP code to be executed safely. This mechanism automatically separates data between users, imposes time and resource limits, and blocks dangerous system functions. The resulting EduPHP platform offers a set of educational functionalities: management of theoretical courses and practical tasks, an interactive code console, a progress tracking system with statistics, and a reporting forum. The result of the work is a fully functional web application that is a safe and effective tool for independent programming learning.

Key words

PHP, Spring Boot, JavaScript, REST API, MySQL

Spis treści

Rozdział 1

Wstęp

Historia programowania zaczęła się w XIX wieku, kiedy to Ada Lovelace stworzyła pierwszy algorytm dla maszyny Charlesa Babbage'a [4, 25]. Rozwój komputerów elektronicznych rozpoczął się w latach 40. ubiegłego wieku, a od czasu opracowania pierwszego algorytmu komputery przeszły znaczące zmiany, porzucając mechaniczne przełączniki najpierw na rzecz lamp elektronowych, a potem na rzecz tranzystorów. Jednakże najważniejszym wydarzeniem było masowe upowszechnienie technologii informatycznych, które miało miejsce na przełomie lat 80. i 90. ubiegłego wieku, a także pojawienie się urządzeń mobilnych — miniaturowych komputerów, takich jak smartfony czy później smartwatche.

W ślad za sprzętem rozwijały się również metody i języki programowania [24]. Zaczynając od zapisywania kodu maszynowego w postaci liczbowej, przez język asemblera, wczesne języki strukturalne i funkcyjne, aż do języków obiektowych, a w ostatnich kilku latach – możliwości generowania kodu na podstawie języka naturalnego. Na przestrzeni lat 90. XX wieku powstały języki, z których wiele jest wykorzystywanych do dnia dzisiejszego. Język Python, którego celem było umożliwienie nauki programowania każdej osobie, język Java, który umożliwiał uruchomienie programu w każdym środowisku, a także PHP, JavaScript oraz C# stosowane także do tworzenia aplikacji internetowych.

Samodzielna nauka nowego języka programowania nie jest rzeczą łatwą. Trudności sprawia zwłaszcza konieczność nawigowania pomiędzy różnymi stronami internetowymi w poszukiwaniu niezbędnych informacji i przykładów. Choć dostępnych jest wiele źródeł, znacznie wygodniejsze i efektywniejsze może być korzystanie z jednej, kompleksowej platformy edukacyjnej, która w przystępny sposób prezentuje materiał, umożliwia śledzenie postępów i oferuje interaktywne środowisko nauki. W przypadku nauki języka PHP (oraz innych języków związanych z usługą WWW), stosowanego głównie do tworzenia dynamicznych stron internetowych, problem ten jest szczególnie widoczny. Osoba rozpoczynająca naukę musi najpierw samodzielnie skonfigurować lokalne środowisko programistyczne, obejmujące serwer, interpreter PHP oraz system zarządzania bazą danych. Jest to zadanie trudne i czasochłonne, co może zniechęcić do nauki. Nawet po poprawnej konfiguracji swobodne eksperymentowanie z kodem, a zwłaszcza z operacjami modyfiku-

jącymi dane w bazach danych, wiąże się z ryzykiem ich uszkodzenia. Istniejące platformy edukacyjne, takie jak Codecademy czy Udemy, oferują wartościowe materiały edukacyjne, ale często skupiają się na teorii lub podstawach składni, nie zapewniając zintegrowanego, bezpiecznego środowiska do praktycznej nauki tworzenia pełnych aplikacji webowych korzystających z baz danych.

W ramach referowanej pracy zrealizowano szereg zadań projektowych. Przeprowadzono analizę istniejących rozwiązań edukacyjnych oraz narzędzi do uruchamiania kodu online. Pozwoliło to na sformułowanie precyzyjnych wymagań dla projektowanego systemu. Na tej podstawie zaprojektowano architekturę aplikacji opartą na rozdzieleniu odpowiedzialności pomiędzy warstwę prezentacji realizowaną w przeglądarce, logikę serwera udostępniającą interfejs programistyczny oraz dwie odseparowane bazy danych – główną i pomocniczą, pełniącą rolę bezpiecznego środowiska testowego. Opracowano moduł środowiska izolowanego wykonania, wymagający opracowania mechanizmów automatycznej modyfikacji zapytań do bazy danych oraz zabezpieczeń blokujących wykonywanie niebezpiecznych operacji. Oprócz tego zaimplementowano pełny zestaw funkcjonalności platformy edukacyjnej, obejmujący system zarządzania użytkownikami, kursami i zadaniami, interaktywny edytor kodu, forum zgłoszeń oraz panel statystyk. Końcowy etap prac poświęcono szczegółowemu sprawdzeniu działania systemu, ze szczególnym uwzględnieniem testów bezpieczeństwa i wydajności.

Zakres pracy obejmuje wszystkie etapy procesu tworzenia oprogramowania: analizę potrzeb, projektowanie, implementację oraz weryfikację i walidację. Praca koncentruje się na praktycznych aspektach budowy aplikacji webowej, a jej największą wartością jest sprawne połączenie zaawansowanych mechanizmów bezpieczeństwa z funkcjonalnościami edukacyjnymi. Kwestie związane z wdrożeniem systemu na publicznym serwerze lub zaawansowaną automatyzacją testów uznano za potencjalne kierunki dalszego rozwoju.

Struktura pracy odzwierciedla kolejność podejmowanych działań. Pierwszym rozdziałem jest niniejszy wstęp. Rozdział drugi zawiera opis analizy tematu, prezentując istniejące rozwiązania oraz precyzyjnie formułując problem techniczny, który należy rozwiązać. Rozdział trzeci poświęcony jest wymaganiom stawianym systemowi oraz opisowi wykorzystanych narzędzi i technologii. W rozdziale czwartym dotyczącym specyfikacji zewnętrznej szczegółowo opisano działanie systemu z perspektywy użytkownika i administratora, przedstawiając scenariusze jego użytkowania. Rozdział piąty, stanowiący specyfikację wewnętrzną, omawia architekturę systemu, strukturę danych, kluczowe algorytmy i rozwiązania programistyczne. Rozdział szósty dokumentuje proces weryfikacji i walidacji, w tym przeprowadzone testy oraz napotkane i usunięte problemy. Ostatni, siódmy rozdział, podsumowuje całość prac, oceniając osiągnięte rezultaty na tle postawionych celów oraz wskazując możliwe ścieżki dalszego rozwoju platformy.

Rozdział 2

Analiza tematu

2.1 Sformułowanie problemu

Kluczowym zagadnieniem, które rozwiązuje prezentowana aplikacja, jest efektywna i bezpieczna nauka programowania w języku PHP [1] z wykorzystaniem środowiska umożliwiającego pracę wielu osobom jednocześnie. Szczególną uwagę skupiono na praktycznych aspektach tworzenia aplikacji webowych, a podstawową barierą dla początkujących programistów jest konieczność samodzielnej konfiguracji lokalnego środowiska deweloperskiego poprzez postawienie i połączenie serwera, PHP oraz bazy danych, co jest procesem czasochłonnym, podatnym na błędy i zniechęcającym.

Nawet po poprawnej konfiguracji, swobodne eksperymentowanie z kodem, a zwłaszcza z operacjami na bazie danych, takimi jak modyfikacja czy usuwanie rekordów, wiąże się z ryzykiem uszkodzenia struktury danych lub utraty wcześniej przygotowanych materiałów ćwiczeniowych. Aplikacja umożliwia utworzenie środowiska przez administratora kursu, który jako profesjonalista posiada doświadczenie w tworzeniu i konfiguracji środowisk, a następnie umożliwia studentom podłączenie się do niego bez konieczności delegowania osobnych zasobów na aplikację dla każdego z uczestników.

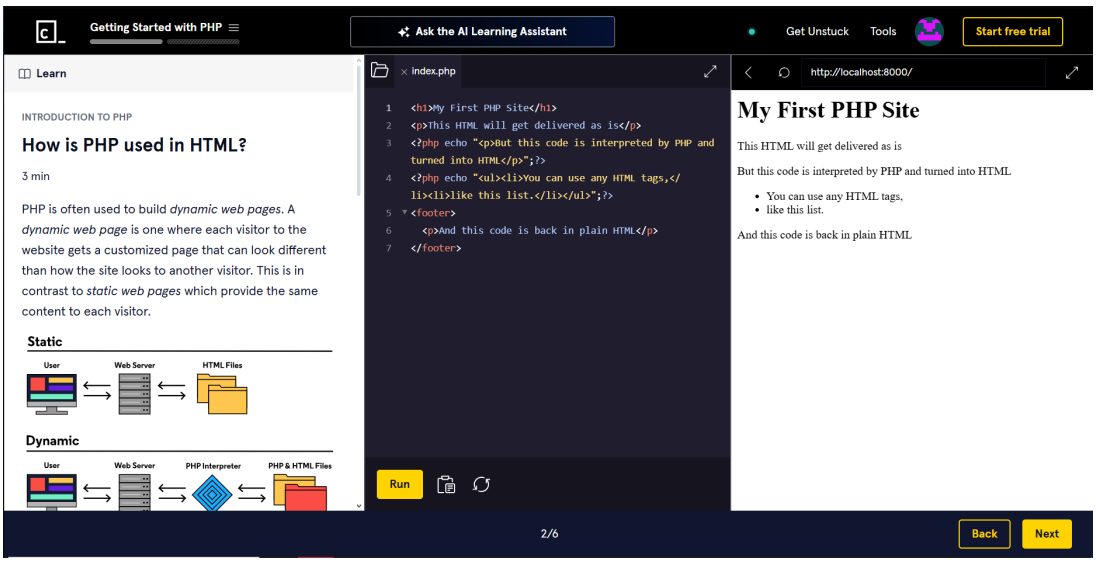
2.2 Istniejące rozwiązania

W niniejszym podrozdziale zaprezentowano rozwiązania umożliwiające naukę programowania oraz uruchamiania kodu online, które funkcjonują obecnie na rynku. Opisano trzy reprezentatywne platformy: Codecademy, Udemy oraz PHPFiddle. Analiza wykazała, że istniejące rozwiązania skupiają się albo na teorii, jak Udemy, albo na podstawach składni, jak Codecademy, albo na ogólnym uruchamianiu kodu, jak PHPFiddle. Brakuje platformy, która łączyłaby wszystkie te aspekty w spójne, bezpieczne środowisko do nauki aplikacji webowych w PHP.

2.2.1 Codecademy

Jedną z najpopularniejszych platform do interaktywnej nauki programowania jest system Codecademy, przedstawiony na rysunku 2.1. Strona WWW systemu umożliwia użytkownikowi naukę składni różnych języków, w tym podstaw PHP, poprzez pisanie kodu bezpośrednio w przeglądarce. System natychmiast weryfikuje poprawność kodu i dostarcza informacji zwrotnej. Codecademy jest wykorzystywany głównie do nauki podstaw programowania poprzez krótkie, modularne lekcje.

Należy jednak zaznaczyć, że środowisko wykonawcze Codecademy jest mocno okrojone i nie pozwala na pracę z prawdziwą bazą danych. Platforma koncentruje się głównie na ćwiczeniu pojedynczych funkcji i algorytmów, co ogranicza jej przydatność w nauce praktycznych operacji na bazie danych.



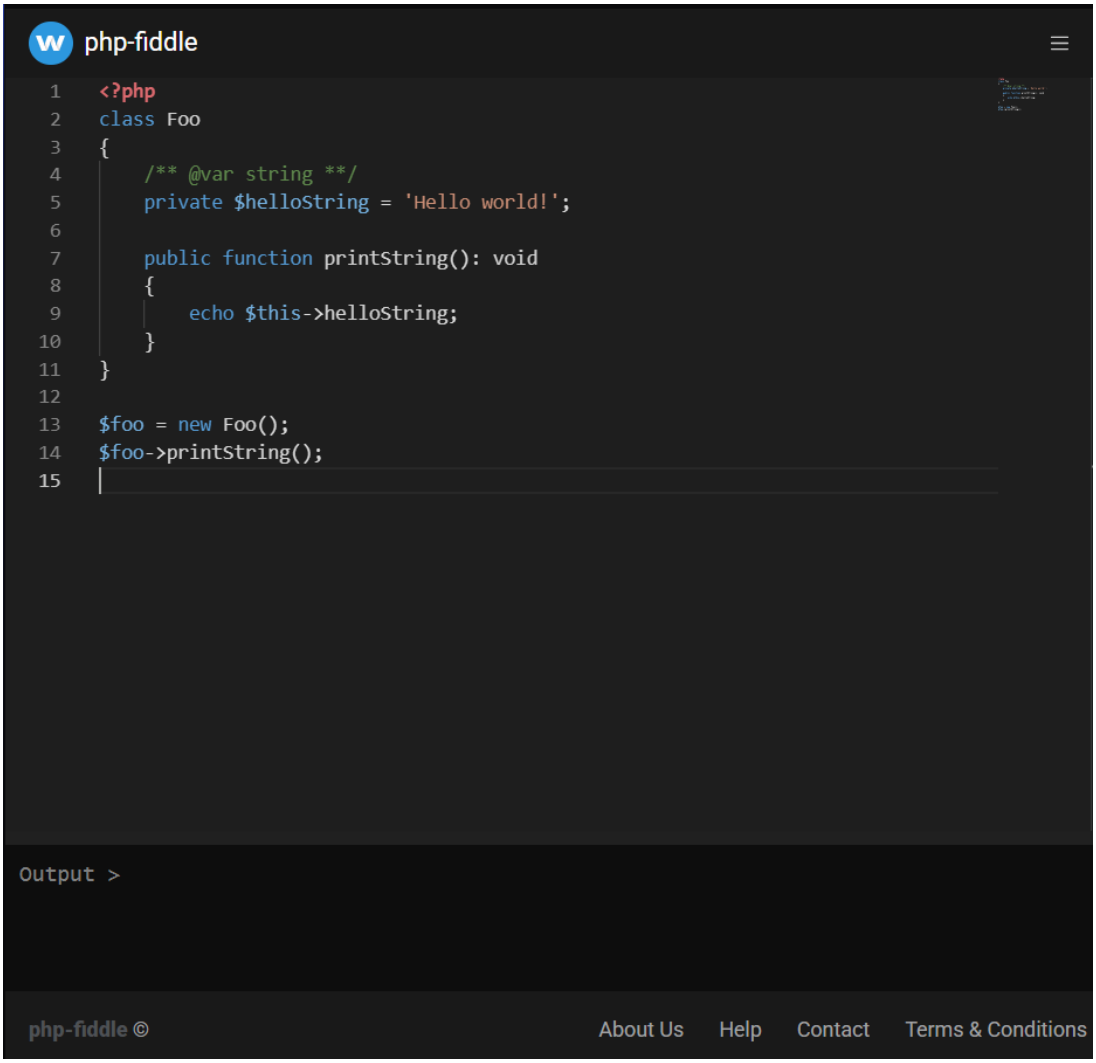
Rysunek 2.1: Interfejs lekcji PHP na platformie Codecademy

2.2.2 Udemy

Udemy, to ogromna platforma z kursami wideo, oferująca setki kursów dotyczących PHP i tworzenia aplikacji webowych. Kursy są tworzone przez indywidualnych instruktorów i zazwyczaj mają formę długich serii filmów, uzupełnionych o materiały do pobrania, quizy oraz zadania wymagające samodzielnej konfiguracji lokalnego środowiska deweloperskiego. Udemy jest popularne wśród osób poszukujących kompleksowych, teoretycznych kursów prowadzonych od podstaw do zaawansowanych zagadnień; jednak konieczność instalacji i konfiguracji wszystkich niezbędnych komponentów, takich jak serwer, PHP oraz baza danych przed rozpoczęciem praktycznej nauki, stanowi znaczącą "barierę wejścia".

2.2.3 PHPFiddle

PHPFiddle, zaprezentowany na rysunku 2.2, to proste internetowe narzędzie, które umożliwia pisanie i natychmiastowe uruchamianie kodu PHP w przeglądarce. W dyspozycji użytkownika jest edytor kodu, przycisk wykonania, a także panel z wynikami. PHPFiddle jest używany głównie przez doświadczonych programistów do szybkiego testowania fragmentów kodu, debugowania lub dzielenia się przykładami, bez konieczności zakładania pełnego projektu.



Rysunek 2.2: Środowisko wykonawcze na stronie PHPFiddle

2.3 Charakterystyczne cechy opracowanej aplikacji

Elementem, który w największym stopniu odróżnia aplikację opracowaną w ramach niniejszej pracy od rozwiązań opisanych powyżej, jest zintegrowane, bezpieczne środowisko wykonawcze PHP z izolowaną bazą danych. W przeciwieństwie do Codecademy, które skupia się na podstawach składni, EduPHP pozwala na wykonywanie pełnych skryptów

PHP z dostępem do rzeczywistej bazy danych MySQL. W odróżnieniu od kursów na Udemy, które wymagają skonfigurowania własnego środowiska, platforma dostarcza gotowe, bezpieczne środowisko, oczywiście po wstępnej konfiguracji przez administratora. Wreszcie, w przeciwieństwie do ogólnego PHPFiddle, platforma EduPHP łączy to środowisko wykonawcze ze strukturalną ścieżką nauki składającej się z kursów, lekcji, zadań o rosnącym poziomie trudności oraz zaawansowanym śledzeniem postępów użytkownika.

Kolejnym aspektem, który wyróżnia platformę EduPHP, jest jej architektura ukierunkowana na bezpieczeństwo i izolację. Aplikacja implementuje mechanizm automatycznej modyfikacji zapytań SQL [5] użytkownika, zapewniający, że operacje na bazie danych są wykonywane tylko na wydzielonym, własnym podzbiorze danych. Eliminuje to ryzyko przypadkowego uszkodzenia danych innych użytkowników, czy też struktury bazy, co jest kluczowe dla swobodnego, bezstresowego eksperymentowania z kodem. Dzięki temu aplikacja stanowi narzędzie do nauki składni.

Rozdział 3

Wymagania i narzędzia

3.1 Wymagania funkcjonalne

Wymagania funkcjonalne opisują zadania, które powinien realizować projektowany system. W niniejszym rozdziale przedstawiono pogrupowane w logiczne moduły kluczowe funkcjonalności platformy edukacyjnej, niezbędne do spełnienia założeń projektowych oraz zapewnienia użytkownikom komfortowej i efektywnej pracy.

3.1.1 Zarządzanie treścią edukacyjną

Platforma edukacyjna powinna zapewniać kompleksowe narzędzia do zarządzania, prezentacji materiałów dydaktycznych oraz kontrolowania postępów. Do kluczowych funkcjonalności w tym obszarze należą:

- Edycja materiałów - System powinien zapewnić możliwość przeglądania uporządkowanych materiałów niezbędnych do zapoznania się i skutecznej nauki programowania w języku PHP.
- Zarządzanie kursami - Platforma powinna umożliwiać dodawanie, edycję i usuwanie kursów oraz zadań przez upoważnionego użytkownika, administratora.
- Podgląd aktywności - System powinien udostępnić użytkownikowi podgląd dotyczący jego ostatniej aktywności, takiej jak ukończone zadanie, wraz z poszczególnymi informacjami na temat tego zadania.

3.1.2 Zarządzanie użytkownikami

Niezbędną częścią platformy jest zapewnienie porządku wśród użytkowników oraz umożliwienie im zarządzania własnymi danymi. Poniższe funkcjonalności służą realizacji tych celów:

- Blokowanie użytkowników - System powinien umożliwiać administratorowi blokowanie użytkowników w przypadku niestosownego zachowania wobec innych użytkowników lub podejmowania próby uruchomienia złośliwego kodu w konsoli lub formularzach, mających na celu uzyskanie dostępu do zabezpieczonych danych, modyfikację struktury bazy danych.
- Edycja profilu - Platforma powinna umożliwiać użytkownikowi zarządzanie własnym profilem, w tym edycję imienia i nazwiska, zmianę zdjęcia profilowego, aktualizację opisu oraz zmianę hasła.

3.1.3 System zadań i interaktywna konsola

Najważniejszym elementem opracowanej aplikacji jest moduł z zadaniami, w którym użytkownik może sprawdzać opanowaną wiedzę. Funkcje omówione poniżej mają ułatwić pracę związaną z rozwiązywaniem zadań programistycznych.

- Zarządzanie zadaniami - System powinien zawierać moduł zarządzania zadaniami, w którym każde zadanie będzie miało opis poleceń, które ma wykonać użytkownik, a także dane o oczekiwanych wynikach.
- Konsola edycji kodu - Platforma powinna posiadać konsolę, która pozwala na przetestowanie kodu napisanego przez użytkownika. Powinna ona umożliwiać obsługę błędów oraz wyświetlanie wyników zaraz po wykonaniu zadania.
- Sygnalizowanie błędów - W przypadku wystąpienia błędów konsola powinna wyświetlać komunikaty o błędach występujących w kodzie użytkownika.
- Funkcja podpowiedzi - Platforma powinna posiadać funkcję podpowiedzi dla każdego zadania, jednak skorzystanie z takiej pomocy powinno powodować wyzerowanie punktów otrzymanych za dane rozwiązanie.
- Resetowanie zadania - Resetowanie zadania musi umożliwiać ponowne podejście do jego rozwiązania, co pozwoli zdobyć maksymalną liczbę punktów oraz sprawdzić opanowanie danego materiału.

3.1.4 Śledzenie postępów użytkowników

Kluczowym elementem efektywnej nauki jest możliwość monitorowania własnych postępów i ocen dotyczących poziomu zaawansowania. System będzie realizował ten cel dzięki spełnieniu wymienionych poniżej wymagań.

- Wizualizacja postępów - Platforma powinna posiadać funkcję graficznej wizualizacji postępów użytkownika, z możliwością wyboru okresu czasu, w którym rozwiązano określone zadanie, czyli zakres z dokładnością do tygodni, miesięcy i lat.

- Prezentacja statystyk - Platforma powinna umożliwić prezentację statystyk dotyczących postępów użytkownika w formie wykresów liniowego, kołowego i słupkowego, obejmujących odpowiednio średnią ocen, liczbę zadań wykonanych na danym poziomie trudności oraz dane o aktywności użytkownika, liczbę wszystkich zadań ukończonych w danym dniu, tygodniu, miesiącu, roku.

- Historia aktywności - System powinien rejestrować i przechowywać historię ukończonych zadań w postaci tabeli zawierającej następujące informacje o każdym zadaniu: datę wykonania, nazwę, poziom trudności, czas wykonania, ocenę, liczbę podjętych prób oraz status.

3.1.5 Uwierzytelnienie i autoryzacja

Podstawowym wymogiem każdej platformy jest zapewnienie bezpieczeństwa danych użytkowników, w szczególności ich haseł. Poniższe wymagania funkcjonalne służą realizacji tego celu:

- System logowania - Platforma powinna udostępniać system rejestracji i logowania użytkowników, zapewniający możliwość tworzenia konta, uwierzytelniania oraz bezpiecznego zarządzania danymi dostępowymi.

- Zarządzanie danymi - System powinien umożliwiać wyłącznie zalogowanym użytkownikom zarządzanie kontem, przeglądanie postępów, dostęp do materiałów edukacyjnych oraz wykonywanie zadań.

3.1.6 Obsługa zgłoszeń

Dodatkową funkcjonalnością platformy będzie system komunikacji, umożliwiający użytkownikom zgłaszanie problemów administratorom. Poniższe wymagania zapewnią możliwość komunikacji i wymianę wiadomości pomiędzy użytkownikami:

- Obsługa zgłoszeń - System powinien umożliwiać użytkownikom tworzenie, edytowanie, usuwanie swoich zgłoszeń dotyczących problemów oraz komentowanie zgłoszeń, przy czym każdy użytkownik może przeglądać i komentować wszystkie istniejące zgłoszenia, lecz edytować oraz usuwać tylko swoje.

- Publikacja fragmentów kodu - System powinien umożliwiać użytkownikom publikowanie zgłoszeń zawierających fragmenty kodu PHP; kod w wiadomości powinien być odpowiednio zabezpieczony i wyświetlany w formie tekstowej, tak aby znaki specjalne były poprawnie widoczne i nie były interpretowane przez przeglądarkę.

3.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne zawierają informacje dotyczące tego, jakie cechy ma projektowany system oraz jakie są jego ograniczenia. W niniejszym podrozdziale opisano wymagania niefunkcjonalne sformułowane w odniesieniu do projektowanego systemu.

3.2.1 Wydajność

Ważną cechą użytkową każdej aplikacji internetowej jest czas, jaki upływa od momentu kliknięcia mającego uruchomić konkretną funkcję do momentu wyświetlenia oczekiwanego wyniku. Do określenia tego czasu mają zostać użyte poniższe metryki:

- czas ładowania strony interfejsu wraz z danymi użytkownika nie może przekraczać 3 sekund;
- czas wykonania kodu użytkownika w konsoli, łącznie z wygenerowaniem wyniku, nie może przekraczać 5 sekund; Powyższe będzie w szczególności odnosić się do przesyłania, interpretowania i wyświetlania wyników zadań.

3.2.2 Użyteczność

Poniższe wymagania niefunkcjonalne opisują, jak platforma ma być przyjazna oku użytkownika, aby poruszanie się po niej było na tyle wygodne i wydajne, żeby nie zanikła chęć dalszego rozwoju jego umiejętności.

- Łatwość użytkowania - Platforma musi zapewniać intuicyjny, przejrzysty i czytelny interfejs użytkownika oraz responsywność umożliwiającą komfortowe korzystanie zarówno na komputerach stacjonarnych, jak i laptopach.
- Język materiałów - Materiały tłumaczące, czego dany użytkownik będzie się uczył w danym kursie, będą napisane w języku polskim, z zachowaniem angielskiej terminologii.

3.2.3 Bezpieczeństwo

Bezpieczeństwo danych jest bardzo ważnym aspektem współczesnego oprogramowania. Poniższe wymagania opisują, w jaki sposób aplikacja zostanie zabezpieczona przed potencjalnymi atakami i niepożądanym dostępem do danych.

- Hasła - Hasła użytkowników muszą być przechowywane w bazie danych w postaci zaszyfrowanej z tzw. solą [9] (ang. *salt*).
- Zabezpieczenie przed niepożądanym kodem - System musi być zabezpieczony przed atakami typu SQL Injection [18] oraz XSS (Cross-Site Scripting) [26].

3.2.4 Architektura systemu

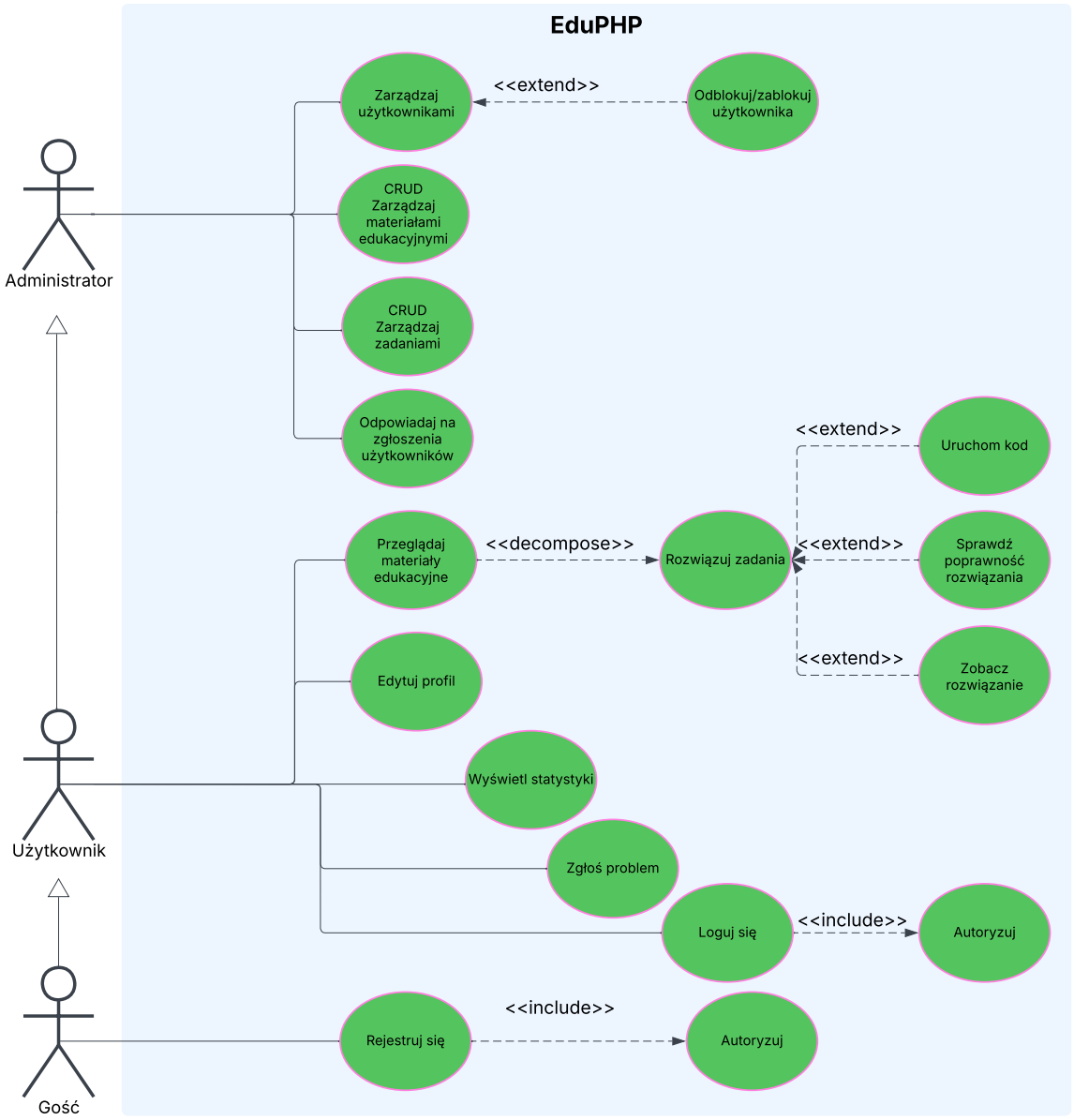
Przemyślana architektura projektu już na początku projektu, decyduje o ostatecznym kształcie systemu. Dobra architektura znacznie podnosi prawdopodobieństwo uzyskania odpowiedniej jakości oprogramowania oraz ułatwia jego przyszłą rozbudowę i serwisowanie. Poniżej przedstawiono wymagania dotyczące architektury systemu.

- System powinien być zrealizowany w oparciu o architekturę REST [12], przy czym warstwa front-end powinna być odseparowana od warstwy back-end, a komunikacja między nimi odbywać się będzie za pomocą tzw. punktów końcowych (ang. *endpoints*).

3.3 Przypadki użycia

Diagramy przypadków użycia są elementem metodyki zorientowanej na przypadki użycia (ang. *Use-Case Driven Development*) [23], która stanowi kluczową część procesu wytwarzania oprogramowania w podejściu obiektowym, takim jak RUP (Rational Unified Process) [15] lub metodyki zwinne. Metoda ta koncentruje się na identyfikacji, specyfikacji i realizacji funkcjonalnych wymagań systemu z perspektywy jego użytkowników końcowych, czyli aktorów. Diagram przypadków użycia modeluje interakcję między aktorami zewnętrznymi a systemem, przedstawiając zestaw przypadków użycia (usług/funkcji), które system oferuje, wraz z relacjami między nimi. Stanowi on punkt wyjścia do dalszej analizy, projektowania i tworzenia testów akceptacyjnych.

Na rysunku 3.1 pokazano diagram przypadków użycia, ilustrujący relacje między aktorami a poszczególnymi scenariuszami dostępnymi w systemie EduPHP.



Rysunek 3.1: Diagram przypadków użycia systemu EduPHP

Tabela 3.1 przedstawia szczegółowy opis przypadków użycia zidentyfikowanych na podstawie załączonego diagramu dla systemu EduPHP.

Tabela 3.1: Przypadki użycia systemu EduPHP

Aktor	**Przypadek użycia**	**Opis**
Użytkownik	Rejestracja	Niezarejestrowana osoba zakłada nowe konto w systemie, podając wymagane dane.
Użytkownik	Logowanie się	Użytkownik uwierzytelnia się w systemie za pomocą loginu i hasła, uzyskując dostęp do spersonalizowanych funkcji
Użytkownik	Przeglądanie materiałów edukacyjnych	Użytkownik przegląda listę dostępnych lekcji i zapoznaje się z ich treścią teoretyczną.
Użytkownik	Rozwiązywanie zadań	Użytkownik wybiera zadanie, pisze kod w interaktywnej konsoli i uruchamia go.
Użytkownik	Sprawdzanie poprawności rozwiązania	System automatycznie weryfikuje kod użytkownika i zwraca wynik (poprawny-/błędny).
Użytkownik	Otrzymywanie komunikatów diagnostycznych	W przypadku błędów w rozwiązaniu, system wyświetla użytkownikowi komunikaty informujące o rodzaju problemu.
Użytkownik	Przeglądanie historii rozwiązań	Użytkownik ma wgląd do archiwum swoich dotychczasowo rozwiązanych zadań wraz z wynikami.
Użytkownik	Przeglądanie statystyk	Użytkownik przegląda swoje statystyki, gdzie wyświetlane są jego postępy, takie jak liczba rozwiązanych zadań, średnia ocen czy ogólny postęp.
Użytkownik	Edycja profilu	Użytkownik może modyfikować swoje dane konta, takie jak login, hasło, imię, nazwisko, opis.
Użytkownik	Zgłaszanie problemu	Użytkownik może wysłać zgłoszenie, np. w przypadku znalezienia błędu w zadaniu, problemów technicznych, lub też pytania dotyczące materiałów i zadań, na które mogą odpowiadać wszyscy użytkownicy.
Administrator	Zarządzanie użytkownikami	Administrator ma możliwość przeglądania listy użytkowników, blokowania lub odblokowywania ich kont.
Administrator	CRUD dla materiałów edukacyjnych	Administrator tworzy, edytuje, przegląda i usuwa lekcje teoretyczne dostępne w systemie.
Administrator	CRUD dla zadań	Administrator tworzy, edytuje, przegląda i usuwa zadania programistyczne.
Administrator	Odpowiadanie na zgłoszenia użytkowników	Administrator odbiera zgłoszenia od użytkowników i udziela na nie odpowiedzi, rozwiązując zgłoszone problemy.

Jak wynika z analizy przedstawionej w tabeli 3.1, system EduPHP obsługuje dwa typy aktorów: Użytkownika oraz Administratora. Użytkownik ma możliwość korzystania z funkcji edukacyjnych platformy, podczas gdy Administrator odpowiada za zarządzanie treścią i utrzymanie systemu.

3.4 Opis narzędzi i metodyka pracy

W niniejszym podrozdziale zawarto opis elementów stosu technologicznego zastosowanego w implementacji opracowanej aplikacji, a także wykorzystanych narzędzi. Przedstawiono uzasadnienie wyboru poszczególnych technologii zarówno dla warstwy back-end, jak i front-end, a także dla systemu zarządzania bazą danych oraz środowiska programistycznego. W podrozdziale omówiono także metodykę pracy nad projektem.

3.4.1 Narzędzia

Wyboru narzędzi zastosowanych w referowanym projekcie dokonano na podstawie analizy wymagań funkcjonalnych i niefunkcjonalnych opisanych w poprzedniej sekcji. Podczas podejmowania decyzji z tym związanych kierowano się popularnością danego narzędzia, jego niezawodnością, a także osobistymi preferencjami, głównie znajomością konkretnego rozwiązania. Dodatkowo pod uwagę brano takie kryteria, jak: wydajność, bezpieczeństwo, responsywność interfejsu użytkownika oraz dostępność narzędzi programistycznych.

Warstwa back-end

Do implementacji logiki biznesowej systemu wykorzystano język Java [16] wraz z frameworkiem Spring Boot [7]. Wybór ten podyktowany był następującymi czynnikami:

- bogatym ekosystemem Spring Framework zapewniającym gotowe komponenty do budowy aplikacji webowych,
- wbudowaną obsługą tworzenia API REST [19] w narzędziu Spring Web,
- możliwością automatycznego generowania zapytań bazodanowych oraz obsługi transakcji uzyskiwanymi dzięki Spring Data JPA [29] i interfejsom CRUD Repository [22],
- wbudowanymi mechanizmami mapowania obiektowo-relacyjnego (ORM) [14], umożliwiającego przeprowadzanie operacji na danych za pośrednictwem obiektów języka Java z automatycznie generowanymi funkcjami get i set,
- możliwością definiowania relacji między encjami (One-to-Many, Many-to-One) bez konieczności pisania zapytań SQL,

- integracją z systemami bezpieczeństwa realizowaną za pomocą technologii Spring Security [28].

Warstwa front-end

Interfejs użytkownika został zaimplementowany z wykorzystaniem następujących technologii:

- **HTML5** [20] - najnowszy standard języka znaczników służący do strukturyzacji i prezentacji treści w w sieci Internet.
- **CSS3** [6] - modularny standard języka arkuszy stylów stanowiący zaawansowane narzędzie do kontroli prezentacji i układu dokumentów HTML.
- **JavaScript (ES6+)** [27] - nowoczesny standard języka skryptowego stanowiący zaawansowane narzędzie dla implementacji interaktywnej logiki biznesowej po stronie klienta w aplikacjach webowych, z wykorzystaniem modułów ES6, `const/let`, async/await oraz nowoczesnych API w przeglądarkach.

Decyzja o zastosowaniu JavaScript zamiast gotowego frameworka front-end została podjęta po analizie specyfikacji projektu. Wybór ten uzasadniony jest prostą architekturą aplikacji, która nie wymaga zaawansowanych mechanizmów zarządzania stanem charakterystycznych dla dużych frameworków. JavaScript zapewnia minimalny narzut przetwarzania i optymalne wykorzystanie zasobów przeglądarki, co przekłada się na krótszy czas ładowania strony oraz lepsze doświadczenie użytkownika. Brak zależności od zewnętrznych frameworków umożliwia pełną kontrolę nad implementacją, precyzyjne dostosowanie kodu do specyficznych wymagań projektu oraz ułatwia proces debugowania. W przypadku interfejsu o umiarkowanej złożoności, JavaScript stanowi wystarczające narzędzie i pozwala uniknąć niepotrzebnego komplikowania architektury, jednocześnie redukując całkowity rozmiar aplikacji poprzez pominięcie dużych bibliotek frameworkowych.

Baza danych

Do przechowywania danych zastosowano system zarządzania bazą danych MySQL [21]. Wybór ten uzasadniony jest następującymi czynnikami:

- dojrzałością technologiczną i stabilnością systemu - MySQL jest sprawdzonym rozwiązaniem o długiej historii rozwoju i szerokim wsparciu społeczności;
- dobrą integracją z frameworkiem Spring Boot - poprzez Spring Data JPA, co zapewnia automatyczną konfigurację połączenia, mapowanie obiektowo-relacyjne oraz generowanie zapytań SQL przez interfejsy CRUD Repository;

- obsługą transakcji ACID [2] - gwarantujących niezawodność operacji na danych poprzez atomowość (atomicity), spójność (consistency), izolację (isolation) i trwałość (durability);
- wydajnością w operacjach CRUD - zoptymalizowanym wykonaniem podstawowych operacji na danych: tworzenia (Create), odczytu (Read), aktualizacji (Update) i usuwania (Delete), co jest kluczowe dla responsywności interfejsu użytkownika;
- powszechnym zastosowaniem w projektach webowych, co przekłada się na bogatą dokumentację, która jest niezbędna podczas pracy, szerokie wsparcie społeczności oraz dostępność gotowych rozwiązań dla typowych problemów implementacyjnych.

Architektura systemu

System został zbudowany w oparciu o architekturę REST API (Representational State Transfer), która zapewnia:

- luźne powiązanie między warstwami - niezależny rozwój front-end i back-end dzięki standaryzowanej komunikacji,
- skalowalność i rozszerzalność - możliwość łatwego dodawania nowych modułów i punktów końcowych bez wpływu na istniejącą funkcjonalność,
- Standaryzację komunikacji - wykorzystanie protokołu HTTP [3] z semantycznymi metodami (GET, POST, PUT, DELETE) i kodami statusu,
- przejrzystą strukturę punktów końcowych - logiczne URI odzwierciedlające zasoby systemu,
- bezstanowość - każde żądanie zawiera wszystkie informacje potrzebne do jego przetworzenia,
- możliwość przyszłej integracji - potencjał rozbudowy o aplikację mobilną lub integrację z zewnętrznymi systemami.

3.4.2 Środowisko programistyczne

W niniejszym podrozdziale scharakteryzowano środowisko programistyczne oraz narzędzia wspomagające wykorzystane podczas prac programistycznych.

Środowisko

Podczas rozwoju projektu wykorzystano dwa zintegrowane środowiska programistyczne dostosowane do specyfiki poszczególnych warstw systemu:

- **Visual Studio Code** - dla warstwy front-end, zapewniające:
  - zaawansowane wsparcie dla technologii webowych (HTML5, CSS3, JavaScript ES6+),
  - debugowanie kodu klienckiego w przeglądarce,
  - integrację z systemem kontroli wersji Git,
  - rozszerzenia do zarządzania projektami front-end,
- **Eclipse IDE** - dla warstwy back-end, oferujące [8]:
  - kompleksowe wsparcie dla języka Java i frameworka Spring Boot,
  - zaawansowane narzędzia debugowania aplikacji serwerowych,
  - integrację z Maven do zarządzania zależnościami,
  - wsparcie dla Spring Data JPA i mapowania ORM.

Architektura systemu

System został zbudowany w oparciu o architekturę REST API z wykorzystaniem wzorca MVC (Model-View-Controller). Komunikacja między front-end a back-end wymagała konfiguracji CORS (Cross-Origin Resource Sharing) ze względu na uruchomienie warstw na różnych portach, co zapewnia:

- bezpieczną komunikację między domenami,
- kontrolę nad dozwolonymi źródłami żądań,
- obsługę żądań między front-end a back-end,
- luźne powiązanie między warstwą front-end a back-end,
- możliwość łatwego rozszerzania systemu o nowe moduły.

Kontrola wersji

Do zarządzania kodem źródłowym wykorzystano system Git z repozytorium hostowanym na platformie GitHub. Umożliwiło to:

- śledzenie historii zmian kodu źródłowego,
- współpracę zdalną nad kodem.

Narzędzia dodatkowe

W procesie dewelopmentu wykorzystano również:

- **Postman** - do testowania punktów końcowych API REST,
- **Maven** - do zarządzania zależnościami i budowania projektu,
- **MySQL Workbench** - do zarządzania bazą danych,
- **narzędzia deweloperskie dla przeglądarek WWW** - do debugowania front-end.

3.5 Metodyka pracy

Niniejszy podrozdział poświęcono zagadnieniom organizacyjnym procesu wytwórczego. Opisano w nim przyjętą metodykę rozwoju oprogramowania, poszczególne etapy prac oraz standardy mające na celu utrzymanie wysokiej jakości kodu.

3.5.1 Przyjęta metodyka pracy

Prace nad projektem realizowano z wykorzystaniem iteracyjno-przyrostowego modelu rozwoju oprogramowania [10]. Wybór takiego podejścia podyktowany był potrzebą stopniowego rozbudowywania systemu oraz elastycznością w dostosowywaniu architektury i funkcjonalności w trakcie implementacji.

3.5.2 Etapy realizacji projektu

Realizację projektu podzielono na pięć głównych faz, odpowiadających kolejnym przyrostom funkcjonalności systemu.

Faza I: Przygotowanie i projektowanie

Początkowy etap prac koncentrował się na stworzeniu podstaw dla dalszego rozwoju systemu. W ramach tej fazy wykonano:

- opracowanie koncepcji systemu EduPHP i wstępnego projektu bazy danych,
- konsultacje i korekty modelu danych,
- inicjalizacja projektu Spring Boot z podziałem na warstwy: model, controller, service, repository, DTO (ang. *Data Transfer Objects*).

Faza II: Zasadnicza część systemu

Etap ten poświęcono implementacji podstawowych mechanizmów systemu oraz integracji kluczowych komponentów. W jego ramach zrealizowano następujące zadania:

- implementację encji i relacji bazodanowych w Spring Data JPA,
- zaprojektowanie interfejsu użytkownika w HTML/CSS/JavaScript,
- integrację front-end z back-end poprzez konfigurację CORS,
- testy integracyjne - rejestracja i logowanie użytkowników.

Faza III: Rozbudowa funkcjonalności

W tej fazie skupiono się na dodaniu głównych modułów funkcjonalnych platformy edukacyjnej. Rozwój systemu na tym etapie systemu:

- system zarządzania kursami i materiałami edukacyjnymi,
- moduł zadań programistycznych z formularzami CRUD,
- profil użytkownika z edycją danych, zdjęcia profilowego, loginu i hasła,
- system zgłoszeń i forum dyskusyjnego.

Faza IV: Zaawansowane moduły

Etap ten obejmował implementację najbardziej złożonych funkcjonalności, stanowiących unikalną wartość dodaną platformy. Przeprowadzone zostały:

- implementacja interaktywnej konsoli PHP z kompilatorem kodu,
- baza danych dla operacji użytkownika (INSERT, DELETE, UPDATE, SELECT),
- system bezpiecznego wrappera dla kodu PHP,
- system statystyk i wizualizacji postępów użytkownika,
- mechanizm oceniania rozwiązań i resetowania zadań,
- rozszerzenie forum o możliwość odpowiadania na zgłoszenia przez wszystkich użytkowników,
- integracja modułów i optymalizacja wydajności.

Faza V: Testowanie i prace końcowe

Ostatni etap prac poświęcono weryfikacji jakości systemu i przygotowaniu go do wdrożenia. W ramach tej fazy przeprowadzono:

- kompleksowe testy manualne wszystkich funkcjonalności,
- testy integracyjne współdziałania modułów systemu,
- testy bezpieczeństwa mechanizmów uwierzytelniania,
- testy użyteczności interfejsu użytkownika,
- przygotowanie dokumentacji technicznej.

3.5.3 Zarządzanie jakością kodu

Kluczowym aspektem utrzymania długoterminowej przejrzystości i stabilności kodu źródłowego było wdrożenie zestawu praktyk deweloperskich. W trakcie implementacji systemu stosowano następujące standardy pracy:

- systematyczne operacje commit - regularne zapisywanie postępów w repozytorium Git,
- refaktoryzacja - ciągłe ulepszanie struktury kodu podczas rozwoju,
- analiza kodu w IDE - wykorzystanie wbudowanych narzędzi do kontroli jakości,
- dokumentowanie architektury - opis kluczowych rozwiązań implementacyjnych,
- testy deweloperskie - ręczna weryfikacja działania implementowanych funkcjonalności.

Rozdział 4

Specyfikacja zewnętrzna

Poprawne uruchomienie i funkcjonowanie systemu EduPHP w środowisku lokalnym uzależnione jest od spełnienia przez komputer hosta oraz stację roboczą użytkownika końcowego określonych wymagań. W niniejszej sekcji szczegółowo określono konfigurację, w której opracowano i przetestowano aplikację. W rozdziale przedstawiono także kategorie użytkowników oraz najważniejsze scenariusze użytkowania systemu. Szczegółowe scenariusze działania, zilustrowane zrzutami ekranu, prezentują typowe ścieżki wykorzystania platformy zarówno przez zwykłych użytkowników, jak i administratorów.

4.1 Wymagania środowiska hosta

Platformę zaimplementowano i zweryfikowano w lokalnym środowisku programistycznym. Poniższa konfiguracja stanowi wzorcowe środowisko uruchomieniowe.

- System operacyjny: Microsoft Windows 10/11 (64-bit) lub inny system wspierający wirtualną maszynę Javy oraz serwer MySQL.
- Środowisko uruchomieniowe Java: Wymagana jest instalacja Java Development Kit (JDK) w wersji 17 lub nowszej. Back-end systemu, zbudowany w oparciu o framework Spring Boot, jest pakowany jako samodzielny, wykonywalny plik archiwum `.jar` zawierający wbudowany serwer aplikacyjny (Apache Tomcat), co eliminuje konieczność jego osobnej instalacji i konfiguracji.
- System zarządzania bazą danych: Wymagana jest lokalna instalacja serwera MySQL w wersji 8.0 Community Edition. W procesie rozwoju wykorzystano narzędzie administracyjne MySQL Workbench 8.0 CE. Dla zapewnienia pełnej kompatybilności z mechanizmem mapowania obiektowo-relacyjnego (ORM) frameworka Spring Data JPA oraz poprawnej obsługi zestawu znaków, kluczowe są następujące ustawienia serwera bazy danych:
  - domyślne kodowanie znaków (character set): `utf8mb4`,

  - domyślne porównywanie znaków (collation): `utf8mb4_0900_ai_ci`,
  - domyślny silnik składowania tabel: `InnoDB`,
  - poziom izolacji transakcji: `READ-COMMITTED` [11].

  Struktura bazy danych dla głównej domeny aplikacji składa się z 9 tabel i zajmuje ok. 352 KiB, natomiast osobna baza dla operacji wykonywanych w konsoli PHP zawiera 3 tabele i zajmuje ok. 64 KiB (to są minima niezbędne do sprawnego przetestowania tego projektu).

- Wymagania sprzętowe hosta:
  - Procesor (CPU): Architektura 64-bitowa. System został pomyślnie przetestowany na procesorze Intel Core i3-6006U (2 rdzenie, 4 wątki, taktowanie 2.00 GHz) z mikroarchitekturą Skylake. Jest to minimalna, wystarczająca konfiguracja.
  - Pamięć operacyjna (RAM): Zalecane minimum to 8 GB. Taka ilość pamięci zapewnia komfortowe jednoczesne działanie środowiska programistycznego (IDE), serwera bazy danych MySQL, procesu back-end Spring Boot oraz przeglądarki internetowej z otwartym interfejsem użytkownika.
  - Dysk twardy: Całość kodu źródłowego, skompilowanych artefaktów oraz niezbędnych narzędzi zajmuje na dysku ok. 180 MB wolnej przestrzeni.

4.2 Wymagania klienckie (stacja robocza użytkownika końcowego)

Platforma EduPHP jest aplikacją webową typu client-server, dostępną wyłącznie za pośrednictwem aktualnych przeglądarek internetowych. Wymagania po stronie klienta zostały odpowiednio zminimalizowane; podano je poniżej.

- Przeglądarka internetowa: System został zweryfikowany pod kątem kompatybilności z aktualnymi wersjami następujących przeglądarek:
  - Google Chrome (wersja 90 i nowsze),
  - Mozilla Firefox (wersja 88 i nowsze),
  - Microsoft Edge (wersja oparta na silniku Chromium, wersja 90 i nowsze),
  - Opera (wersja 76 i nowsze).

  Aby zapewnić pełną funkcjonalność interfejsu, w szczególności działanie interaktywnej konsoli kodu, w ustawieniach przeglądarki należy włączyć obsługę JavaScript.

Ponadto, ze względu na separację warstw i komunikację między domenami związaną z osobnym uruchomieniem frontend i back-end, przeglądarka musi obsługiwać mechanizm CORS (Cross-Origin Resource Sharing, który domyślnie jest spełniony przez wszystkie wymienione przeglądarki.

- Serwer deweloperski dla front-end: Ze względu na mechanizm bezpieczeństwa CORS oraz konieczność poprawnego ładowania modułów ES6, pliki front-end (`.html`, `.css`, `.js`) nie mogą być otwierane bezpośrednio z dysku (schemat `file://`). Wymagane jest ich udostępnienie przez lokalny serwer HTTP. W procesie rozwoju wykorzystano rozszerzenie Live Server dla środowiska Visual Studio Code, które automatycznie uruchamia serwer deweloperski (domyślnie na porcie 5500 lub 3000).

- Rozdzielczość ekranu i interfejs: Interfejs użytkownika zaprojektowano z wykorzystaniem zasad Responsive Web Design (RWD), dostosowując układ strony do różnych rozmiarów okna przeglądarki. Dla poprawienia tzw. doświadczenia użytkownika (ang. *user experience*), zalecane jest korzystanie z ekranu o minimalnej rozdzielczości 1366x768 pikseli. Interfejs jest zoptymalizowany do pracy w trybie pełnoekranowym.

- Połączenie sieciowe: W kontekście środowiska lokalnego, nie jest wymagane zewnętrzne połączenie internetowe. Niezbędna jest jedynie funkcjonalność pętli zwrotnej (*localhost*) w systemie operacyjnym dla komunikacji między przeglądarką a serwerem aplikacji.

4.3 Instalacja i konfiguracja systemu

Niniejsza sekcja zawiera szczegółowy opis instalacji systemu EduPHP w środowisku lokalnym. Opisane kroki umożliwiają uruchomienie wszystkich komponentów platformy: bazy danych, oprogramowania back-end oraz front-end, wraz z konfiguracją ich wzajemnej komunikacji.

4.3.1 Krok 1: Przygotowanie środowiska bazowego

Przed rozpoczęciem instalacji należy upewnić się, że na komputerze hosta spełnione są wszystkie wymagania opisane w sekcji 4.1. W szczególności muszą być zainstalowane:

- Java Development Kit (JDK) w wersji 17 lub nowszej,

- serwer bazy danych MySQL w wersji 8.0 Community Edition (np. z narzędziem MySQL Workbench),

- system kontroli wersji Git (opcjonalnie, do sklonowania repozytorium),

- środowisko programistyczne Visual Studio Code z rozszerzeniem *Live Server* (lub dowolny inny lokalny serwer HTTP) do uruchomienia warstwy front-end.

4.3.2 Krok 2: Pobranie kodu źródłowego

Kod źródłowy systemu EduPHP jest dostępny w publicznym repozytorium GitHub. Należy go pobrać na dysk lokalny.

- Opcja A (zalecana): Używając klienta Git, należy wykonać polecenie w wybranym katalogu:

```
git clone https://github.com/anastasiiapashko/EduPHP.git
```

- Opcja B: Należy pobrać archiwum ZIP bezpośrednio z repozytorium GitHub (`https://github.com/anastasiiapashko/EduPHP`) i wypakować je w wybranej lokalizacji.

W strukturze pobranego projektu znajdują się główne katalogi: `EduPHP_Backend/` (aplikacja Spring Boot), `EduPHP_Frontend/` (pliki HTML, CSS, JavaScript) oraz `php/` (binaria interpretera PHP).

4.3.3 Krok 3: Konfiguracja baz danych

System wykorzystuje dwie niezależne bazy danych MySQL: główną dla aplikacji i pomocniczą dla środowiska piaskownicy (*sandbox*) zadań PHP. Ta separacja zwiększa bezpieczeństwo, izolując operacje wykonywane przez użytkowników w konsoli PHP.

1. Uruchom serwer MySQL oraz narzędzie administracyjne (np. MySQL Workbench).
2. Utwórz dwóch użytkowników i puste bazy danych, wykonując następujące zapytania SQL:

```
-- Baza główna aplikacji
CREATE DATABASE eduphp CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci;
CREATE USER 'root'@'localhost' IDENTIFIED BY '6eKeVBMV89EE';
GRANT ALL PRIVILEGES ON eduphp.* TO 'root'@'localhost';
FLUSH PRIVILEGES;

-- Baza piaskownicy dla zadań PHP
CREATE DATABASE eduphp_sandbox CHARACTER SET utf8mb4
    COLLATE utf8mb4_0900_ai_ci;
```

```
CREATE USER 'php_sandbox_user'@'localhost'
    IDENTIFIED BY 'sandbox_password123';

-- Ograniczone uprawnienia: tylko podstawowe operacje DML
GRANT SELECT, INSERT, UPDATE, DELETE ON eduphp_sandbox.*
    TO 'php_sandbox_user'@'localhost';
FLUSH PRIVILEGES;
```

Uwaga dotycząca bezpieczeństwa: Użytkownik `php_sandbox_user` ma celowo ograniczone uprawnienia wyłącznie do operacji DML (`SELECT`, `INSERT`, `UPDATE`, `DELETE`). Nie ma on praw do modyfikacji struktury bazy danych (`DROP`, `ALTER`, `CREATE TABLE`), co stanowi dodatkową warstwę zabezpieczeń przed potencjalnie szkodliwym kodem PHP wykonywanym przez użytkowników platformy.

3. System nie wymaga ręcznego importowania wstępnej struktury tabel. Podczas pierwszego uruchomienia back-end, framework Spring Data JPA automatycznie utworzy niezbędne tabele do głównej bazy danych dzięki ustawieniu `spring.jpa.hibernate.ddl-auto=update` w pliku konfiguracyjnym.

4. Wypełnienie bazy danych można zrealizować własnoręcznie lub skorzystać z pliku z przygotowanymi danymi *daneDlaEduphp.sql*.

5. Następnie zaimportuj plik *sandboxSQL.sql* do bazy danych sandbox. Jest to plik o przykładowej zawartości danych.

4.3.4 Krok 4: Konfiguracja i uruchomienie back-end - Spring Boot

Backend jest aplikacją samodzielną (*fat JAR*), która zawiera wbudowany serwer aplikacyjny. Należy wykonać następujące kroki:

1. przejście do katalogu `EduPHP_Backend/`,

2. (opcjonalnie) sprawdzenie i ewentualnie dostosowanie pliku konfiguracyjnego `src/main/resources/application.properties`. Kluczowe parametry połączenia z bazą danych są już predefiniowane i zgodne z zapytaniami z kroku 3. Należy zweryfikować ścieżkę do interpretera PHP:

   ```
   php.executor.path:C:\\ścieżka\\do\\EduPHP\\php\\php.exe
   ```

Ścieżka ta musi odpowiadać lokalizacji wypakowanego interpretera PHP w sklonowanym projekcie.

3. uruchomienie aplikacji (dwa sposoby):
   - z poziomu IDE (np. Eclipse, IntelliJ): zaimportować projekt jako projekt Maven i uruchomienie głównej klasy aplikacji (`EduPHPApplication.java`).
   - z wiersza poleceń: w katalogu projektu wykonać komendę Maven: `mvn spring-boot:run`; aplikacja zostanie skompilowana i uruchomiona.
4. po poprawnym starcie w konsoli pojawią się logi Spring Boot, a oprogramowanie back-end będzie nasłuchiwało żądań na porcie 8082. Konfiguracja CORS jest ustawiona na zezwalanie żądań z adresów front-end (`localhost:5500`).

4.3.5 Krok 5: Uruchomienie front-end

Warstwa front-end to zbiór statycznych plików, które muszą być udostępnione przez serwer HTTP –

1. przejście do katalogu `EduPHP_Frontend/`.
2. uruchomienie lokalnego serwera deweloperskiego: w środowisku Visual Studio Code z zainstalowanym rozszerzeniem *Live Server* wystarczy kliknąć prawym przyciskiem myszy na plik `index.html` i wybrać opcję *"Open with Live Server"*,
3. front-end zostanie automatycznie uruchomiony, domyślnie na porcie 5500 (`http://127.0.0.1:5500`). Jest to adres skonfigurowany dla warstwy back-end jako dozwolone źródło (*origin*) dla żądań CORS.

4.3.6 Krok 6: Weryfikacja i pierwsze uruchomienie

Poniższa procedura opisuje kroki niezbędne do pierwszego uruchomienia platformy EduPHP i zainicjowania jej podstawowej konfiguracji poprzez utworzenie konta administratora. Ponieważ weryfikacja poprawnego uruchomienia systemu stanowi kluczowy etap procesu instalacji, potwierdzający, że wszystkie komponenty platformy — baza danych, backend oraz frontend — komunikują się ze sobą poprawnie, a użytkownik może rozpocząć korzystanie z aplikacji.

1. Otwarcie przeglądrki internetowej i przejście pod adres, na którym działa oprogramowanie front-end (np. `http://localhost:5500`).
2. Ponieważ bazy danych startują jako puste, pierwszym krokiem musi być utworzenie konta administratora. Należy to zrobić za pomocą narzędzia do testowania API (np. Postman), wysyłając żądanie HTTP POST na punkt końcowy back-endu:

- **URL:** `http://localhost:8082/api/saveData`
- **Method:** POST
- **Header:** `Content-Type: application/json`
- **Body (JSON):**

```
{

    "firstName": "Anastasiia",
    "rola": "admin",
    "login": "admin",
    "passwd": "admin123",
    "secondName": "Pashko"
}
```

W przypadku pomyślnej rejestracji punkt końcowy zwróci obiekt użytkownika z odpowiednim potwierdzeniem. Kluczowym polem jest pole roli: „ADMIN", które nadaje użytkownikowi uprawnienia administracyjne. Jeżeli punkt końcowy będzie wysłany bez wskazania roli administratora, to automatycznie utworzy się rola typu "USER".

3. Po utworzeniu konta administratora, można zalogować się do systemu przez interfejs przeglądarki (używając loginu i hasła podanych w kroku powyżej) i rozpocząć konfigurację platformy: dodawać kursy, zadania oraz zarządzać użytkownikami.

4. Zwykli użytkownicy (z rolą `USER`) mogą samodzielnie rejestrować się poprzez formularz rejestracji dostępny na stronie głównej platformy, bez konieczności użycia narzędzi zewnętrznych.

4.4 Kategorie użytkowników i ich role

Platforma EduPHP obsługuje dwie podstawowe kategorie użytkowników, wyróżnione na podstawie przydzielonych uprawnień: Zwykłego Użytkownika oraz Administratora. Taki podział zapewnia odpowiednią separację obowiązków: użytkownik koncentruje się na nauce, podczas gdy administrator zarządza treścią, utrzymuje porządek w systemie i również może testować każde zadanie jak zwykły użytkownik.

4.4.1 Zwykły użytkownik

Użytkownik o tej roli jest uczestnikiem procesu edukacyjnego. Jego głównym celem jest nauka programowania w PHP poprzez zapoznawanie się z materiałami i rozwiązywanie zadań. Do jego kluczowych uprawnień i możliwych akcji należą:

- Rejestracja - założenie nowego konta w systemie poprzez podanie wymaganych danych.
- Logowanie - uwierzytelnienie się za pomocą loginu i hasła w celu uzyskania dostępu do spersonalizowanych funkcji.
- Przeglądanie materiałów edukacyjnych - przeglądanie listy dostępnych lekcji teoretycznych i zapoznawanie się z ich treścią.
- Rozwiązywanie zadań - wybór zadania programistycznego, napisanie kodu w interaktywnej konsoli PHP oraz jego uruchomienie.
- Sprawdzanie poprawności rozwiązania - automatyczna weryfikacja kodu przez system i otrzymanie wyniku (poprawny/błędny).
- Otrzymywanie komunikatów diagnostycznych – w przypadku błędów w kodzie system wyświetla komunikaty informujące o rodzaju problemu.
- Przeglądanie historii rozwiązań – wgląd do archiwum wszystkich dotychczas rozwiązanych zadań wraz z ich wynikami.
- Przeglądanie statystyk – analizowanie swoich postępów w nauce na podstawie statystyk, takich jak liczba rozwiązanych zadań, średnia ocen czy ogólny postęp.
- Edycja profilu – modyfikacja danych własnego konta, takich jak login, hasło, imię, nazwisko oraz opis.
- Wysyłanie i komentowanie zgłoszeń – wysyłanie zgłoszeń w przypadku znalezienia błędu w zadaniu, napotkania problemów technicznych lub zadawania pytań dotyczących materiałów. Użytkownik może także przeglądać i komentować zgłoszenia innych użytkowników.

4.4.2 Administrator

Administrator pełni rolę nadzorczą, odpowiadając za utrzymanie platformy, aktualność materiałów i bezpieczeństwo. Dysponuje on wszystkimi uprawnieniami zwykłego użytkownika, a dodatkowo posiada pełną kontrolę administracyjną nad systemem, obejmującą:

- Zarządzanie użytkownikami – przeglądanie listy wszystkich zarejestrowanych użytkowników oraz możliwość blokowania lub odblokowywania ich kont, np. w przypadku niestosownego zachowania lub próby uruchomienia złośliwego kodu.
- CRUD materiałów dla edukacyjnych – pełne zarządzanie treścią dydaktyczną: tworzenie, edytowanie, przeglądanie i usuwanie lekcji teoretycznych.

- CRUD dla zadań programistycznych – pełne zarządzanie zadaniami: tworzenie, edytowanie, przeglądanie i usuwanie zadań wraz z ich opisami, testami oraz oczekiwanymi wynikami.

Podsumowując, system implementuje model uprawnień oparty na dwóch rolach, gdzie zwykły użytkownik skupia się wyłącznie na korzystaniu z funkcji edukacyjnych, a administrator posiada uprawnienia nadrzędne do zarządzania całą platformą i jej użytkownikami.

4.5 Scenariusze działania systemu

Niniejsza sekcja stanowi przewodnik po interfejsie użytkownika platformy EduPHP, opisując główne ścieżki nawigacji oraz sposób korzystania z kluczowych funkcjonalności zarówno z perspektywy zwykłego użytkownika – rola USER, jak i administratora – rola ADMIN.

4.5.1 Wspólna struktura nawigacji i układ interfejsu

Po pomyślnym zalogowaniu użytkownik, niezależnie od roli, zostaje przekierowany do głównego pulpitu, który pełni rolę punktu centralnego aplikacji. Interfejs ekranu oparty jest na responsywnym układzie składającym się z kilku kluczowych komponentów:

Pasek nawigacyjny

Pasek nawigacyjny podzielony jest na logiczne części:

1. Lewa część, w skład której wchodzą logo oraz menu główne. Dla zwykłego użytkownika dostępne są następujące pozycje:

   - *Kursy* - przejście do listy wszystkich dostępnych kursów teoretycznych,
   - *Zadania* - dostęp do listy zadań programistycznych,
   - *Zgłoszenia* - panel systemu wsparcia i komunikacji,
   - *Statystyki* - podsumowanie postępów użytkownika.

   Użytkownik z rolą administratora widzi dodatkowo dwie pozycje:

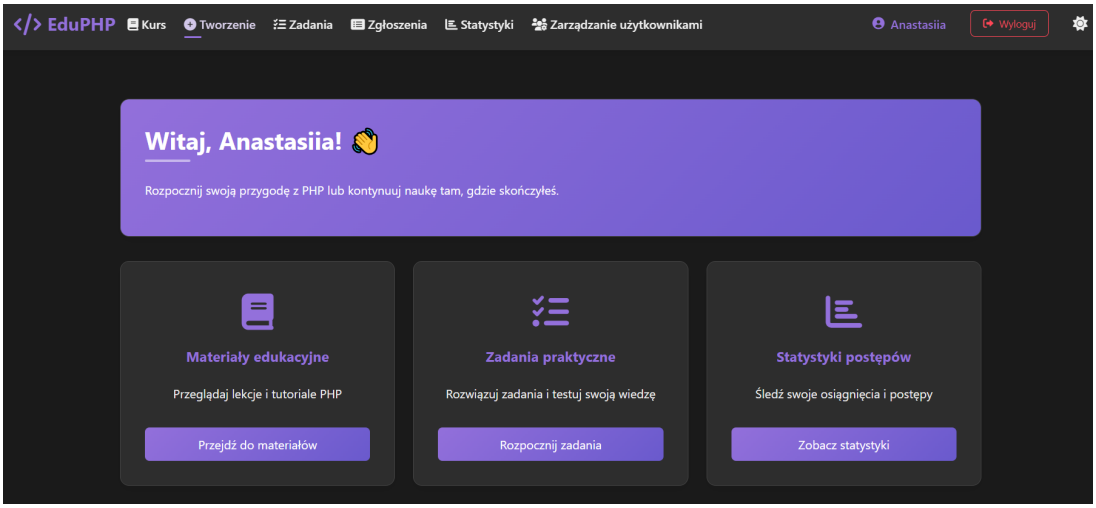
   - *Tworzenie* - panel administracyjny do zarządzania kursami i zadaniami,
   - *Zarządzanie użytkownikami* - panel administracyjny do zarządzania kontami użytkowników.

2. Prawa część, składająca się z panelu użytkownika: Zawiera przycisk profilu, z wyświetlanym imieniem użytkownika; przycisk wylogowania oraz przełącznik motywu interfejsu jasny lub ciemny zależnie od wyboru.

Główny pulpit

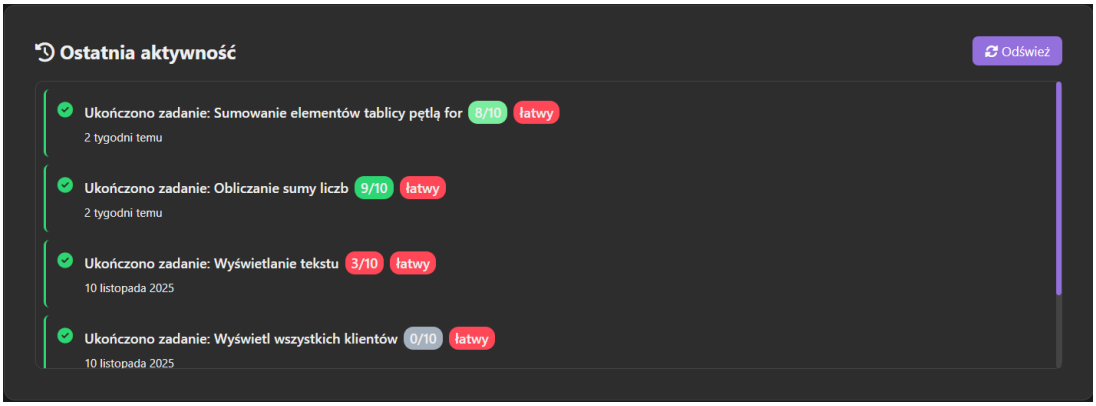
Pulpit prezentuje przywitanie oraz zestaw kart szybkiego dostępu:

- *Materiały edukacyjne* - prowadzi bezpośrednio do listy kursów,
- *Zadania praktyczne* - przekierowuje do listy wszystkich zadań,
- *Statystyki postępów* - otwiera szczegółowe statystyki użytkownika.



Rysunek 4.1: Główny pulpit platformy EduPHP z paskiem nawigacyjnym

Ponadto, w sekcji *Ostatnia aktywność* wyświetlana jest lista najnowszych działań użytkownika, między innymi jak ostatnio rozwiązane zadania, którą można odświeżyć.

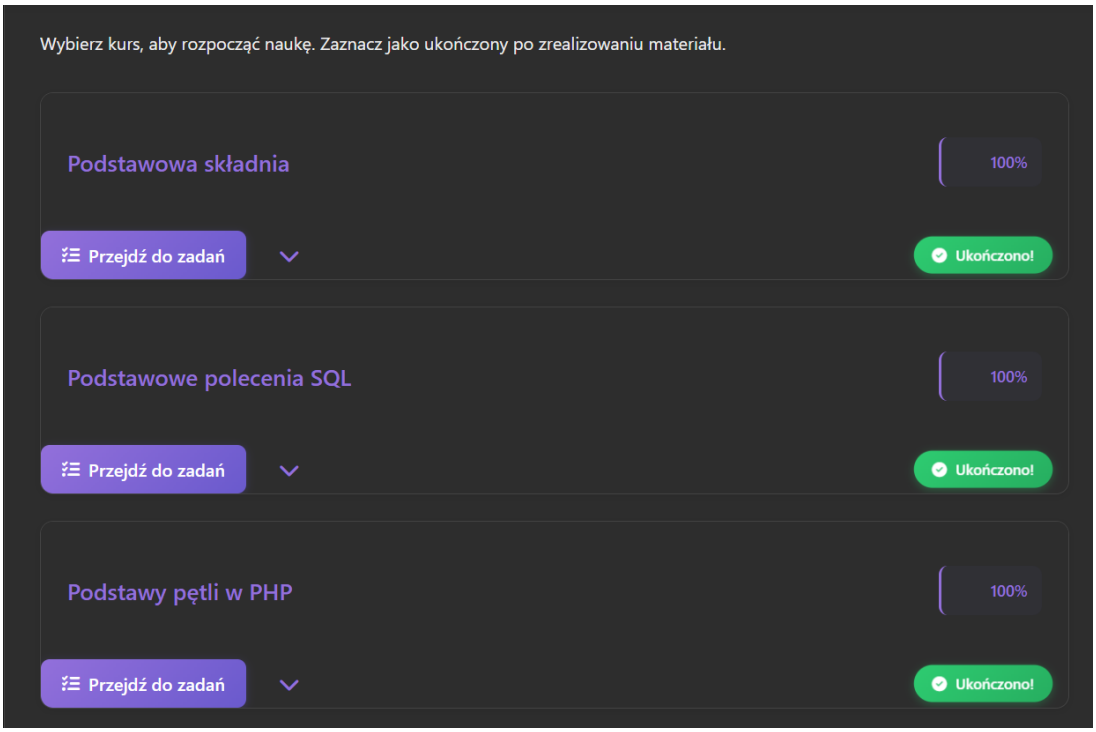


Rysunek 4.2: Panel reprezentujący ostatnią aktywność

4.5.2 Przeglądanie i realizacja kursów

Kliknięcie przycisku *Kursy* w menu głównym przenosi użytkownika do strony listy kursów.

1. Na stronie zostaje zaprezentowany moduł każdego dostępnego kursu, w skład którego wchodzą: Tytuł, opis, opcjonalny osadzony film instruktażowy, oraz przycisk *Rozpocznij Kurs*.
2. Po kliknięciu przycisku użytkownik uzyskuje dostęp do treści teoretycznej kursu.
3. W nagłówku strony widoczne jest podsumowanie postępu: liczba ukończonych kursów oraz ogólny procent ukończenia.
4. Bezpośrednio z tej strony użytkownik ma możliwość przejścia do formularza zgłoszenia za pomocą dedykowanego przycisku.



Rysunek 4.3: Lista dostępnych kursów edukacyjnych

4.5.3 Rozwiązywanie zadań programistycznych

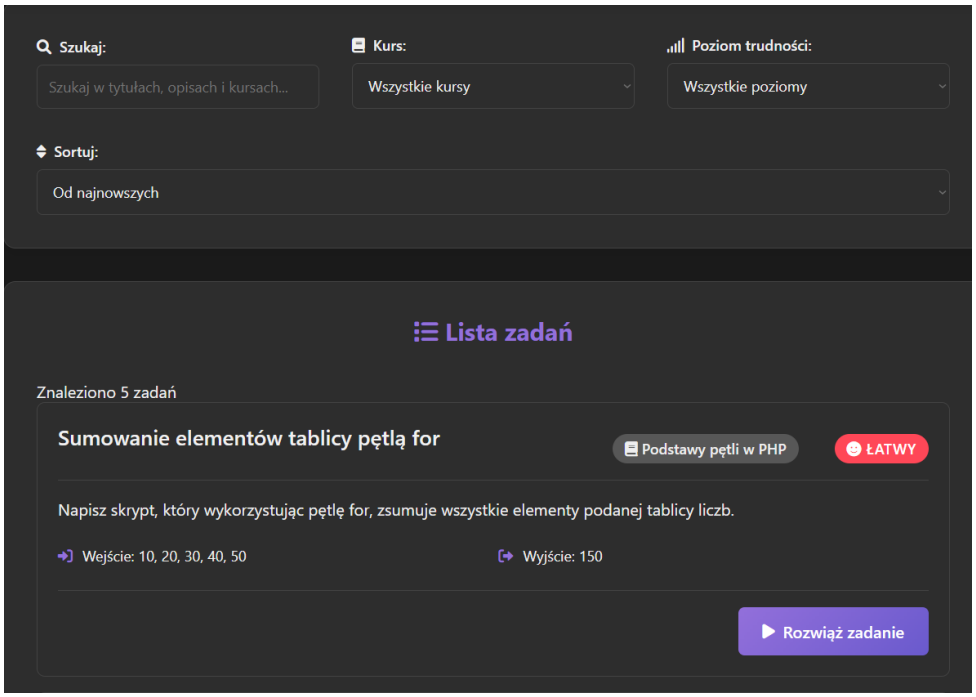
Ścieżka pracy z zadaniami składa się z dwóch etapów: przeglądania listy i rozwiązywania konkretnego zadania. Dostępna jest dla użytkowników obu ról.

Przeglądanie listy zadań

Przeglądanie listy zadań stanowi punkt wyjścia do pracy z modułem zadań platformy. Jego interfejs zaprojektowano w sposób intuicyjny, zapewniając użytkownikowi możliwość łatwego odnalezienia odpowiedniego zadania:

- lista zadań zawiera filtrowanie według: kursu, poziomu trudności (łatwy, średni, trudny) oraz sortowanie (data, trudność),

- każde zadanie na liście prezentuje swój tytuł, przypisany kurs, poziom trudności, status ukończenia oraz przycisk *Rozwiąż zadanie*.



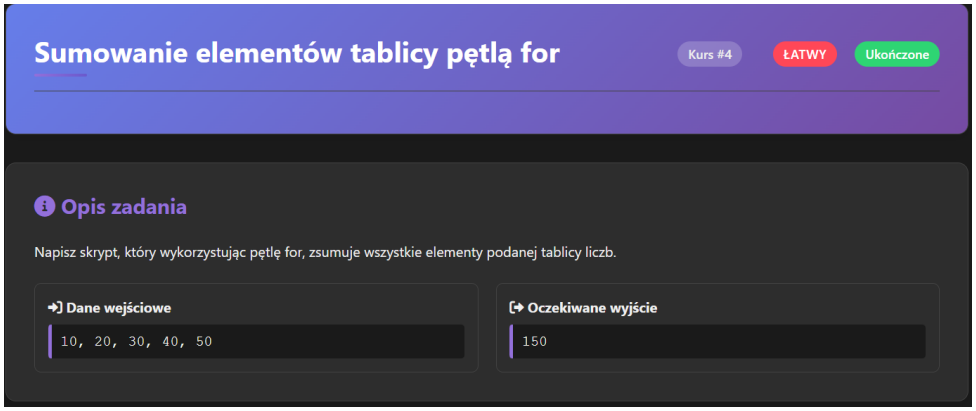
Rysunek 4.4: Lista zadań z możliwością filtrowania i sortowania

Ekran rozwiązania zadania

Po wybraniu zadania użytkownik trafia na dedykowany ekran podzielony na sekcje:

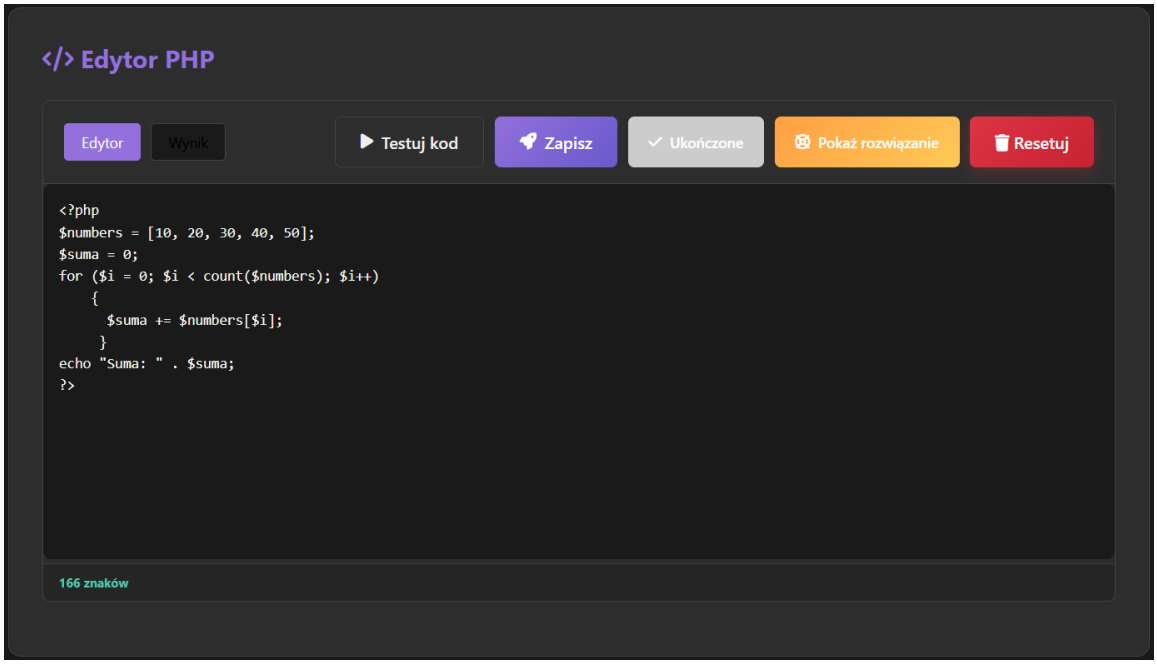
1. Nagłówek: Tytuł zadania, kurs źródłowy, poziom trudności *i indywidualny status (np. „Ukończone”,* „W trakcie”).

2. Opis zadania: Szczegółowy opis problemu do rozwiązania wraz z sekcjami *„Dane wejściowe”* oraz *„Oczekiwane wyjście”*.



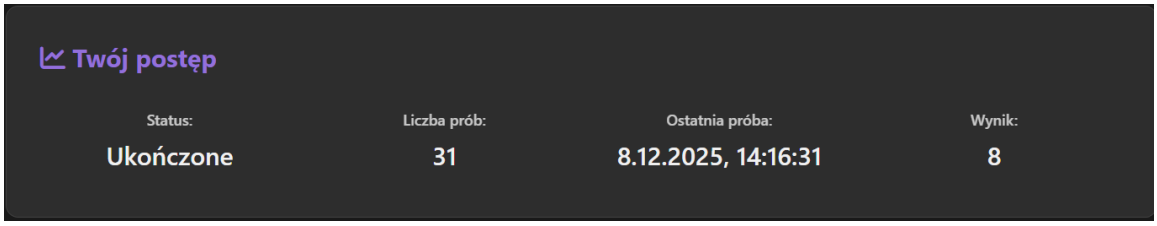
Rysunek 4.5: Opis oraz szczegóły zadania

3. Edytor kodu PHP: Główny panel interaktywnej konsoli. Zawiera on:
   - edytor tekstowy do wpisywania kodu PHP,
   - przyciski sterujące: *Testuj kod* (weryfikacja bez zapisu), *Zapisz* (zapisanie bieżącego stanu), *Skończyłem* (ostateczne zgłoszenie rozwiązania),
   - przyciski pomocnicze: *Pokaż rozwiązanie* (pokazuje gotowe rozwiązanie, ale zeruje punkty), *Resetuj* (przywraca kod początkowy),
   - panel wynikowy na osobnej zakładce, wyświetlający wynik wykonania kodu lub komunikaty błędów.



Rysunek 4.6: Ekran rozwiązania zadania z edytorem kodu PHP

4. Panel postępu: podsumowanie liczby podjętych prób, daty ostatniej próby oraz aktualnego wyniku.

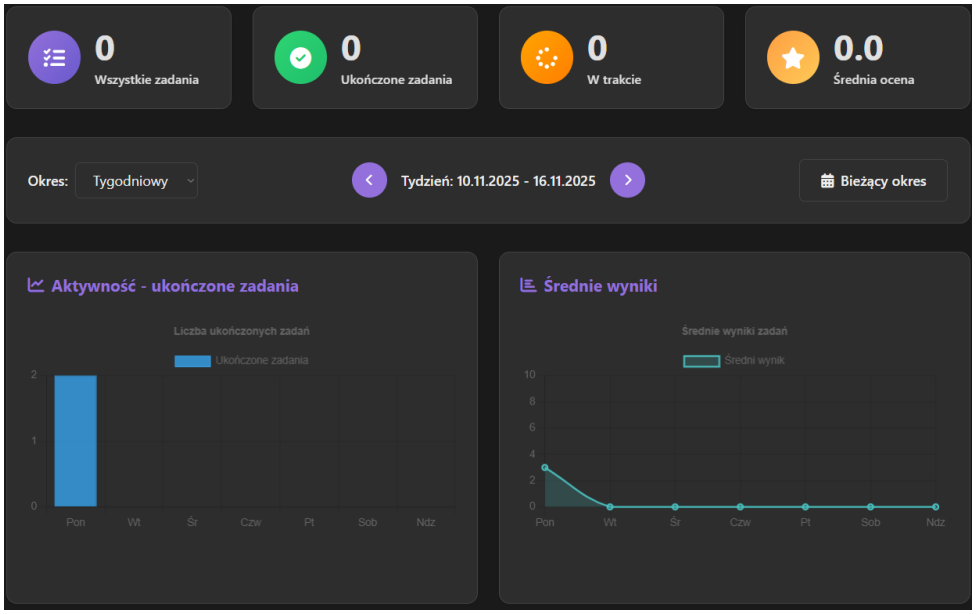


Rysunek 4.7: Wyniki po ukończeniu zadania

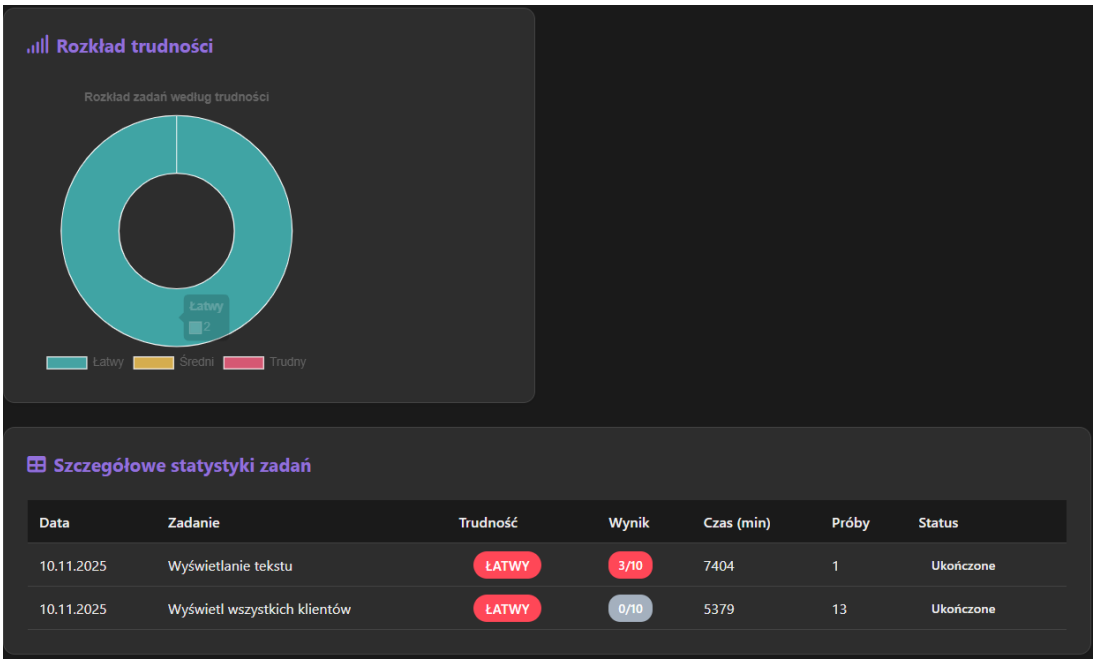
4.5.4 Monitorowanie postępów – statystyki

Na tej stronie oferowany jest kompleksowy przegląd osiągnięć użytkownika z wykorzystaniem wizualizacji i tabel.

- karty podsumowujące: wyświetlają łączną liczbę zadań, liczbę ukończonych zadań, zadań w trakcie realizacji oraz średnią ocen.
- kontrolki okresu: umożliwiają analizę danych w perspektywie tygodniowej, miesięcznej lub rocznej oraz nawigację w czasie,
- wykresy: trzy główne wykresy liniowy, słupkowy, kołowy, prezentujące odpowiednio: aktywność w czasie (ukończone zadania), średnie wyniki oraz rozkład zadań według poziomu trudności,
- tabela szczegółów: zawiera pełną historię wszystkich prób rozwiązania zadań z danymi: data, nazwa zadania, trudność, wynik, czas wykonania, liczba prób oraz status.



Rysunek 4.8: Panel statystyk z wykresami i podsumowaniami część 1



Rysunek 4.9: Panel statystyk z wykresami i podsumowaniami część 2

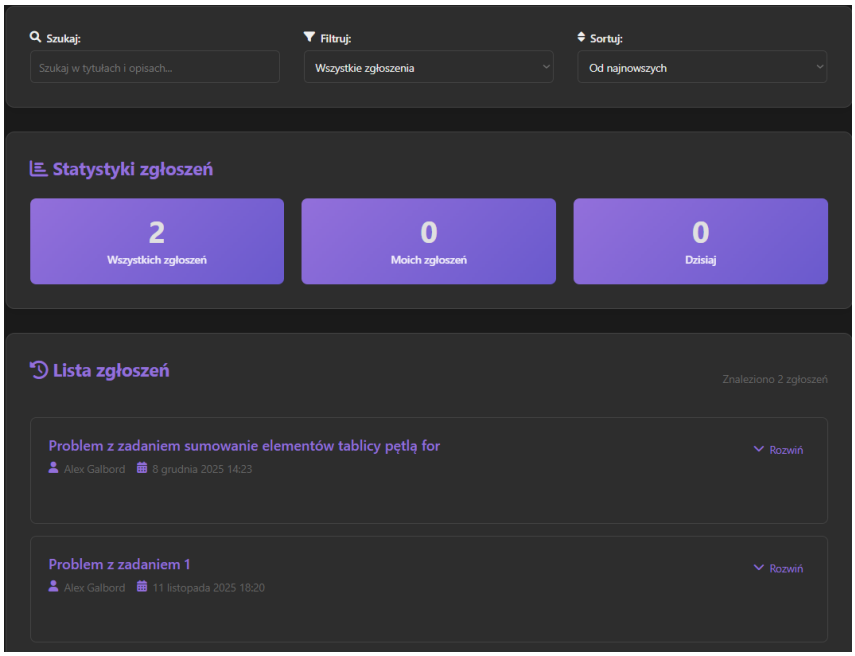
4.5.5 System zgłoszeń i pomocy

Platforma posiada zintegrowany system zgłoszeń, służący do komunikacji z administratorem oraz innymi użytkownikami w sprawie potencjalnych problemów. System ten jest dostępny dla obu ról.

Przeglądanie i filtrowanie zgłoszeń

Panel zgłoszeń służy jako centralny system komunikacji między użytkownikami a administratorem platformy. Umożliwia zgłaszanie problemów, zadawanie pytań:

- użytkownik widzi listę wszystkich zgłoszeń w systemie, które może filtrować, według wszystkie lub tylko moje oraz sortować,
- dla każdego zgłoszenia widoczny jest jego tytuł, autor, data utworzenia, status oraz liczba komentarzy,
- kliknięcie na zgłoszenie rozwija jego szczegóły, pozwalając na przeczytanie opisu i dodanie własnego komentarza. Administrator może dodatkowo zmieniać status zgłoszenia.

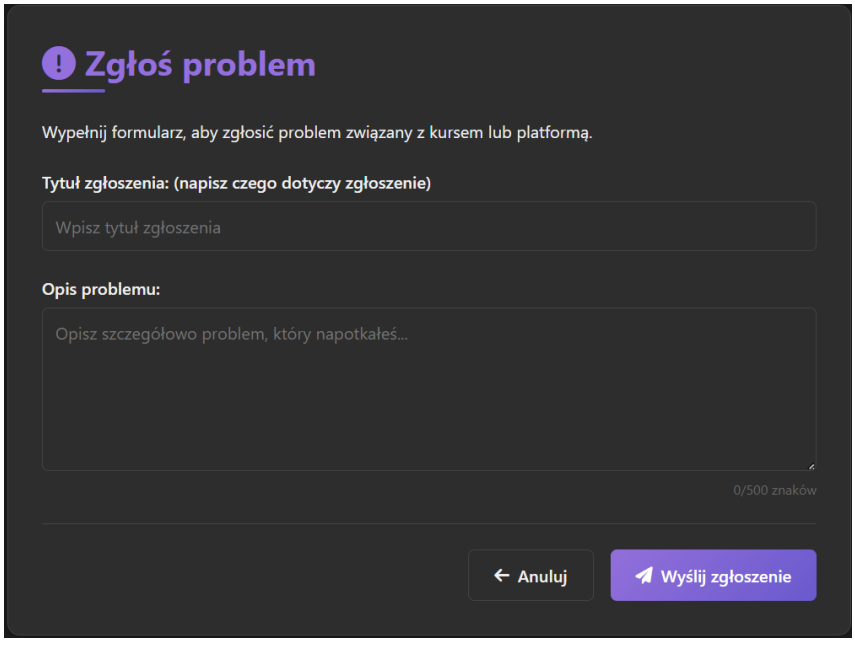


Rysunek 4.10: Lista zgłoszeń z możliwością filtrowania

Tworzenie nowego zgłoszenia

Formularz zgłoszeniowy został zaprojektowany w sposób minimalny, aby ułatwić szybkie opisanie napotkanego problemu:

- formularz wymaga podania tytułu oraz szczegółowego opisu problemu,
- po wysłaniu zgłoszenie trafia do ogólnej listy, gdzie może być komentowane przez innych użytkowników oraz administratorów.

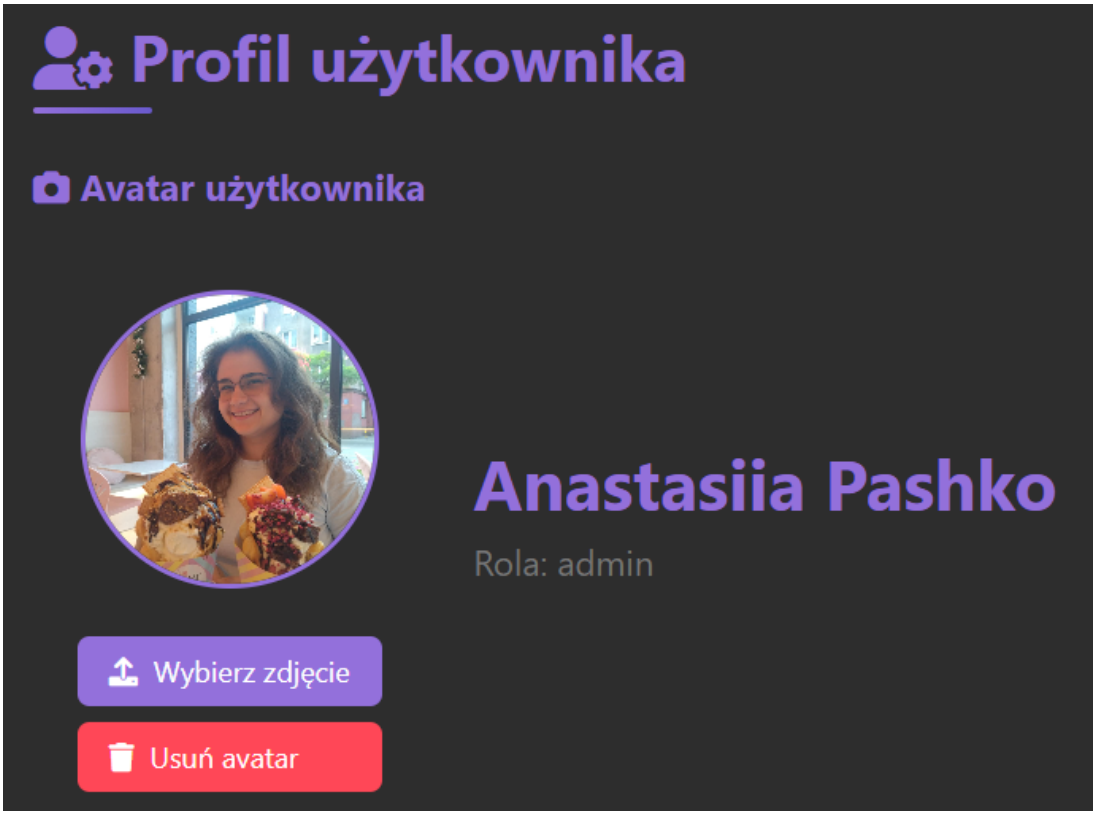


Rysunek 4.11: Formularz to tworzenia zgłoszeń

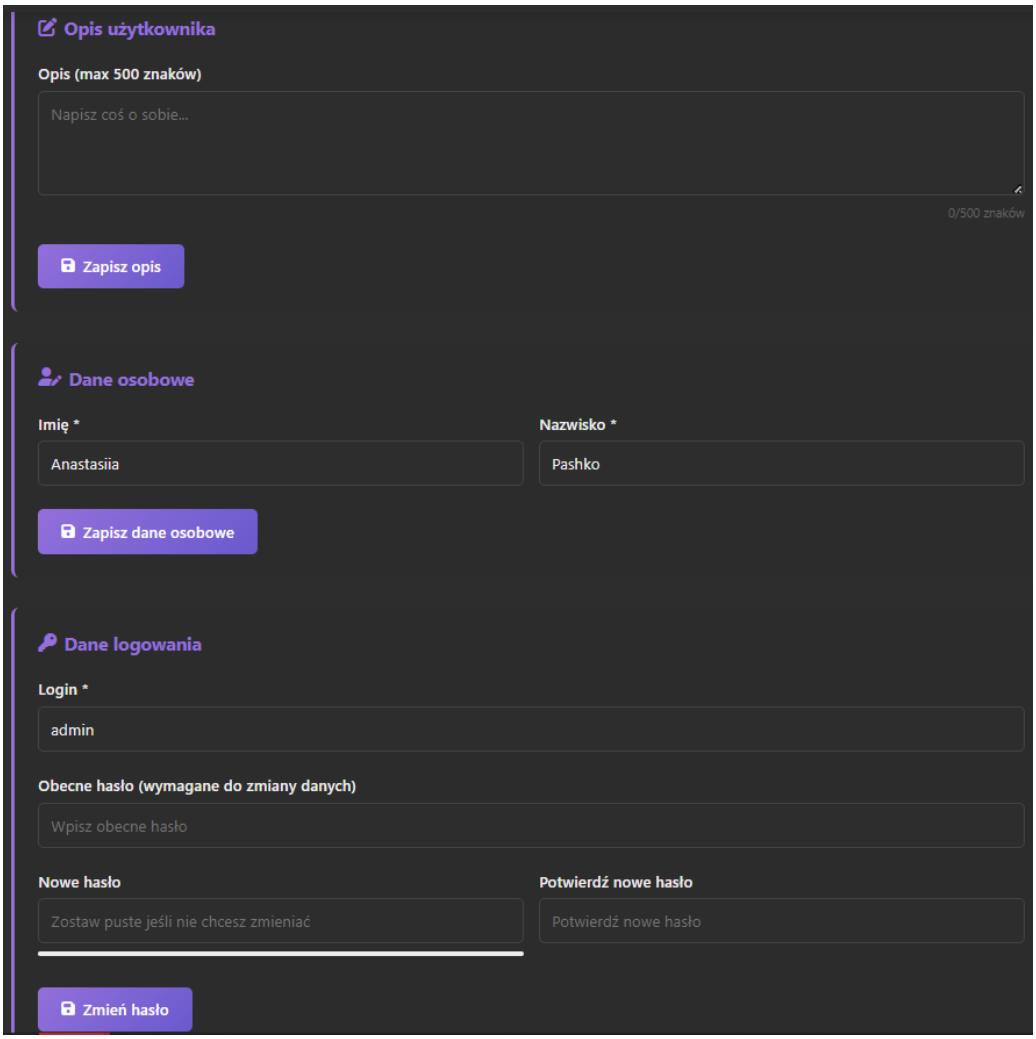
4.5.6 Zarządzanie profilem użytkownika

Strona profilu umożliwia zarządzanie danymi konta w czterech sekcjach:

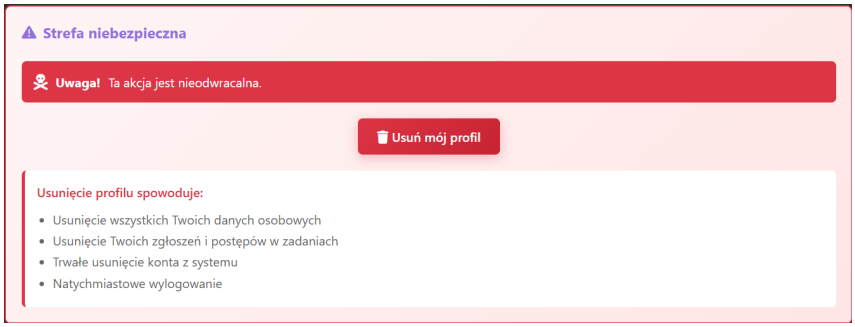
1. *Awatar*: Możliwość przesłania, podglądu i usunięcia zdjęcia profilowego na rysunku 4.12,
2. *Opis użytkownika*: Edycja pola opisu o długości do 500 znaków,
3. *Dane osobowe*: Modyfikacja imienia i nazwiska,
4. *Dane logowania*: Zmiana loginu oraz hasła, wymagane podanie obecnego hasła w celu weryfikacji, na rysunku 4.13,
5. *Usunięcie konta*: Sekcja *„Strefa niebezpieczna”* umożliwia trwałe usunięcie profilu po potwierdzeniu hasłem, na rysunku 4.14.



Rysunek 4.12: Zdjęcie profilowe użytkownika



Rysunek 4.13: Panel zarządzania profilem użytkownika



Rysunek 4.14: Usunięcie profilu użytkownika

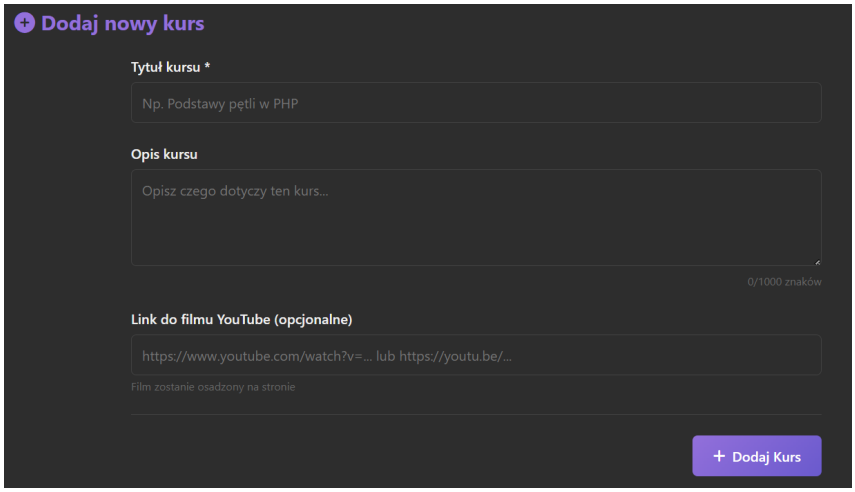
4.5.7 Funkcjonalności administracyjne

Użytkownik z rolą administratora posiada dostęp do dodatkowych paneli zarządzania:

Tworzenie i zarządzanie kursami oraz zadaniami

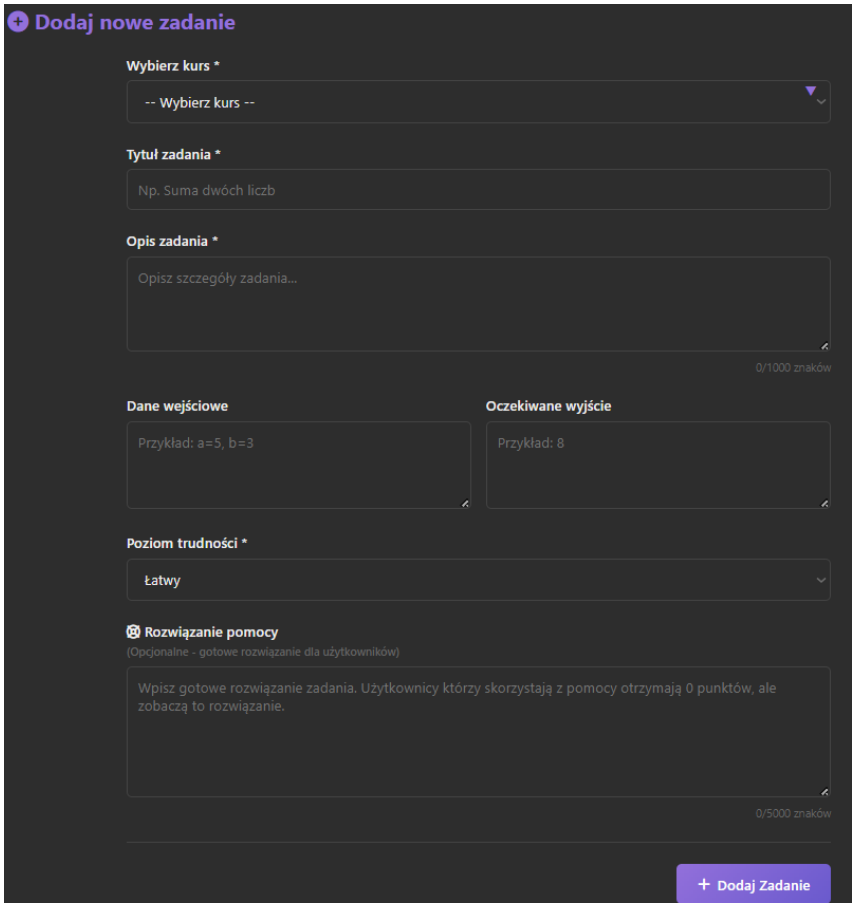
Panel ten daje możliwość administratorowi:

- dodawanie nowego kursu poprzez formularz z polami: tytuł, opis i opcjonalny link do filmu YouTube,



Rysunek 4.15: Formularz tworzenia nowego kursu

- dodawanie nowego zadania poprzez formularz z polami: wybór kursu, tytuł, opis, dane wejściowe, oczekiwane wyjście i poziom trudności,



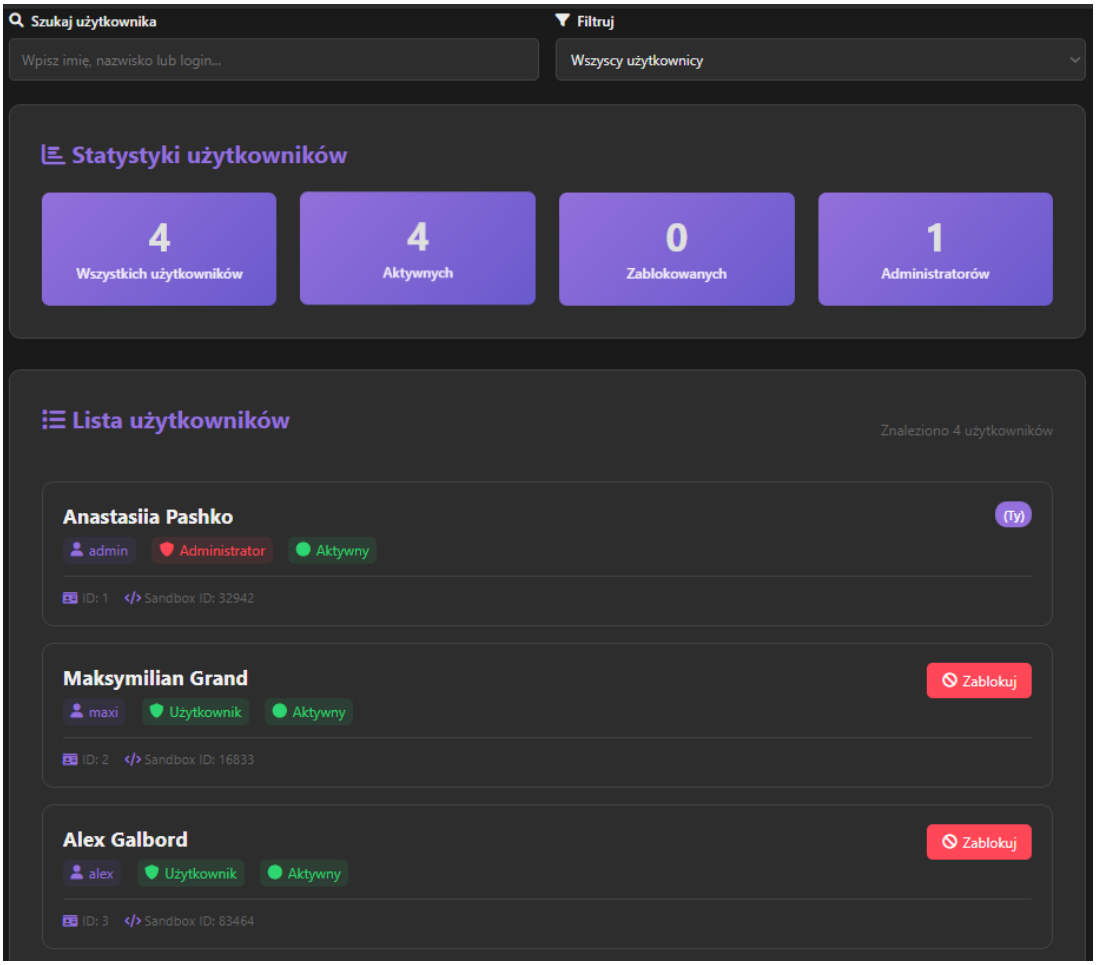
Rysunek 4.16: Formularz tworzenia nowego zadania

- przeglądanie, edycja i usuwanie istniejących kursów oraz zadań.

Zarządzanie użytkownikami

Przez ten panel administrator ma możliwość:

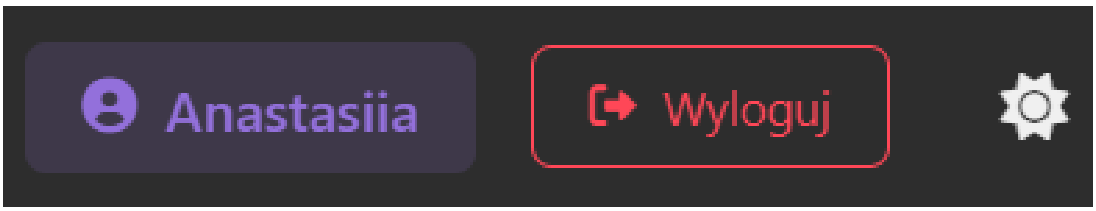
- przeglądanie listy wszystkich zarejestrowanych użytkowników z możliwością filtrowania (wszyscy, aktywni, zablokowani, administratorzy, zwykli użytkownicy),
- blokowanie i odblokowywanie kont użytkowników,
- podgląd statystyk użytkowników: liczba wszystkich, aktywnych, zablokowanych i administratorów.



Rysunek 4.17: Panel zarządzania użytkownikami z listą i filtrami

4.5.8 Wylogowywanie

Proces kończenia sesji jest jednoetapowy - użytkownik naciska przycisk *Wyloguj* znajdujący się w prawym górnym rogu interfejsu. Po kliknięciu sesja zostaje zakończona, a użytkownik jest przekierowywany do strony głównej platformy .



Rysunek 4.18: Przycisk wylogowania użytkownika z profilu

4.6 Kwestie bezpieczeństwa

Platforma EduPHP, jako system umożliwiający wykonanie dowolnego kodu PHP wprowadzonego przez użytkowników, stanowiła szczególne wyzwanie w zakresie bezpieczeństwa. Głównym celem było stworzenie środowiska piaskownicy (ang. sandbox), które całkowicie izoluje kod użytkownika od krytycznych zasobów systemowych, baz danych aplikacji oraz innych użytkowników, przy jednoczesnym zachowaniu pełnej funkcjonalności edukacyjnej. Wdrożono kompleksową strategię bezpieczeństwa, obejmującą kilka uzupełniających się warstw.

4.6.1 Izolacja środowiska wykonawczego

Kluczowym elementem architektury bezpieczeństwa jest fizyczne i logiczne oddzielenie środowiska wykonania kodu użytkownika od głównej aplikacji.

- Oddzielna baza danych piaskownicy: Kod PHP wykonywany przez użytkowników działa wyłącznie na dedykowanej bazie danych. Jest ona całkowicie niezależna od głównej bazy danych aplikacji. Użytkownik piaskownicy posiada celowo ograniczone uprawnienia (`SELECT`, `INSERT`, `UPDATE`, `DELETE`), bez możliwości modyfikacji schematu bazy (`CREATE`, `ALTER`, `DROP`).

- Izolacja danych użytkowników: Każdy użytkownik platformy otrzymuje unikalny, losowo wygenerowany identyfikator (zakres 999 - 99999). Mechanizm automatycznie dołącza warunek `WHERE user_id = [id]` do wszystkich zapytań `SELECT`, `UPDATE` i `DELETE` oraz automatycznie wstawia ten identyfikator przy operacjach `INSERT`. Dzięki temu użytkownik widzi i modyfikuje wyłącznie swoje dane, nawet jeśli wykonuje „surowy” kod SQL. Dane innych użytkowników oraz dane administracyjne są dla niego niedostępne.

- Wykonywanie w osobnym procesie: Kod użytkownika jest uruchamiany jako zewnętrzny proces interpretera PHP, zarządzany przez aplikację Java. Proces ten ma nałożone restrykcyjne limity czasowe (czas wykonania nie może przekraczać 10 sekund) oraz pamięciowe (limit 128 MB), a po ich przekroczeniu jest automatycznie przerywany.

4.6.2 Restrykcje na poziomie interpretera PHP

Przed wykonaniem kod wprowadzony przez użytkownika jest automatycznie opakowywany w warstwę zabezpieczającą generowaną przez serwis wykonujący kod PHP.

- Wyłączenie niebezpiecznych funkcji: Interpreter PHP jest uruchamiany ze specjalną konfiguracją, która całkowicie blokuje dostęp do funkcji stanowiących zagrożenie bezpieczeństwa. Zablokowane funkcje obejmują:
  - Funkcje wykonujące polecenia systemowe: `system()`, `exec()`, `shell_exec()`, `passthru()`
  - Funkcje tworzące procesy: `popen()`, `proc_open()`
  - Funkcje dynamicznego wykonania kodu: `eval()`, `assert()`
  - Funkcje sieciowe: `curl_exec()`

  Blokada jest wymuszana na poziomie konfiguracji procesu PHP, co uniemożliwia jej obejście przez kod użytkownika.

- Automatyczne modyfikacje zapytań SQL: Platforma wykorzystuje własną implementację połączenia z bazą danych, która automatycznie dodaje zabezpieczenia do wszystkich zapytań SQL:
  - Do każdego zapytania `SELECT` dodawany jest warunek izolacji danych użytkownika oraz ograniczenie liczby zwracanych rekordów (maksymalnie 100).
  - Operacje `INSERT` automatycznie otrzymują identyfikator użytkownika i są limitowane pod względem liczby wstawianych danych.
  - Operacje `UPDATE` i `DELETE` wymuszają filtrowanie tylko do danych należących do bieżącego użytkownika.

4.6.3 Bezpieczeństwo aplikacji webowej (Backend – Spring Boot)

Warstwa aplikacyjna została zabezpieczona zgodnie z najlepszymi praktykami dla frameworka Spring.

- Bezpieczne przechowywanie haseł: Hasła użytkowników są przechowywane w bazie danych w postaci zahaszowanej przy użyciu algorytmu BCrypt, implementowanego przez Spring Security. Podczas rejestracji hasło jawne jest natychmiast przekształcane w bezpieczny skrót przed zapisem do bazy. W procesie logowania wprowadzone hasło jest porównywane z zahaszowaną wartością, co eliminuje konieczność przechowywania haseł w formie jawnej.
- Rozszerzona walidacja wejścia: Zaimplementowano wielopoziomową walidację danych rejestracyjnych:
  - Walidacja długości: hasło musi mieć od 6 do 100 znaków.
  - Walidacja formatu: login i dane osobowe są sprawdzane pod kątem dozwolonych znaków.
  - Ochrona przed atakami: analiza pod kątem wzorców SQL Injection, XSS, Command Injection.
  - Ochrona przed znakami specjalnymi: filtrowanie null bytes, sekwencji path traversal.
- Uwierzytelnianie i autoryzacja: Dostęp do punktów końcowych API jest kontrolowany. Kluczowe operacje (np. zarządzanie kursami, użytkownikami) wymagają roli `ADMIN`.
- Ochrona przed atakami webowymi: Skonfigurowany mechanizm CORS (Cross-Origin Resource Sharing) precyzyjnie definiuje dozwolone źródła żądań, zapobiegając nieautoryzowanym wywołaniom API.
- Bezpieczna komunikacja z procesem PHP: Ścieżka do interpretera PHP oraz dane dostępowe do bazy piaskownicy są konfigurowane za pomocą zmiennych środowiskowych, a nie na stałe w kodzie.

4.6.4 Bezpieczeństwo danych użytkownika i prywatność

Dane użytkowników są chronione na kilka sposobów. System zabezpiecza informacje przed nieautoryzowanym dostępem i stara się pozostawiać jak najmniejszy ślad działania na serwerze

- Bezpieczne przechowywanie rozwiązań: Kod użytkownika jest zapisywany na dysku serwera w postaci plików `.php`, ale zawsze z nałożoną warstwą zabezpieczającą. Pliki są organizowane w osobnych katalogach dla każdego użytkownika.
- Czyszczenie plików tymczasowych: Pliki tworzone dla pojedynczego wykonania kodu są niezwłocznie usuwane po zakończeniu procesu, ograniczając ślad na dysku.

- Ochrona przed enumeracją: Losowo generowane identyfikatory w systemie piaskownicy uniemożliwiają łatwe zgadywanie identyfikatorów innych użytkowników.

4.6.5 Podsumowanie strategii bezpieczeństwa

Zastosowana w EduPHP wielowarstwowa strategia „obrony w głąb" (ang. defense in depth) zapewnia, że nawet w przypadku obejścia jednego zabezpieczenia (np. błędnej walidacji wejścia), kolejne warstwy (izolacja bazy, ograniczenia PHP, limitowanie SQL) skutecznie powstrzymują potencjalnie szkodliwy kod. Główna aplikacja i jej dane pozostają w pełni odizolowane od środowiska wykonawczego udostępnionego użytkownikom końcowym. Mechanizmy te pozwalają na bezpieczne oferowanie interaktywnej nauki programowania w środowisku zbliżonym do produkcyjnego.

4.7 Przykład działania

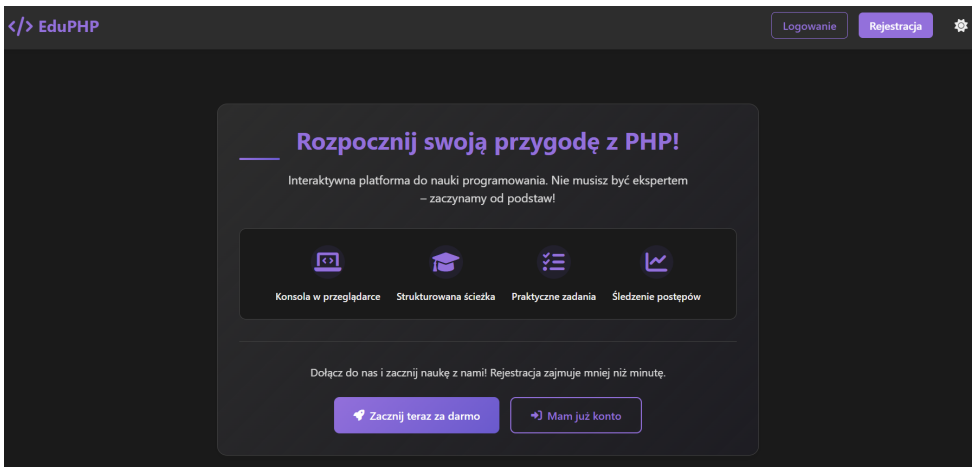
W niniejszej sekcji przedstawiono typowe ścieżki użytkowania platformy EduPHP, ilustrujące sposób realizacji kluczowych celów przez różnych użytkowników systemu. Każdy scenariusz opisuje sekwencję kroków prowadzących do osiągnięcia konkretnego celu edukacyjnego lub administracyjnego.

4.7.1 Scenariusz 1: Nauka przez rozwiązywanie zadań

Celem tego scenariusza jest pokazanie, jak użytkownik korzysta z platformy do nauki programowania w PHP poprzez rozwiązywanie zadań praktycznych.

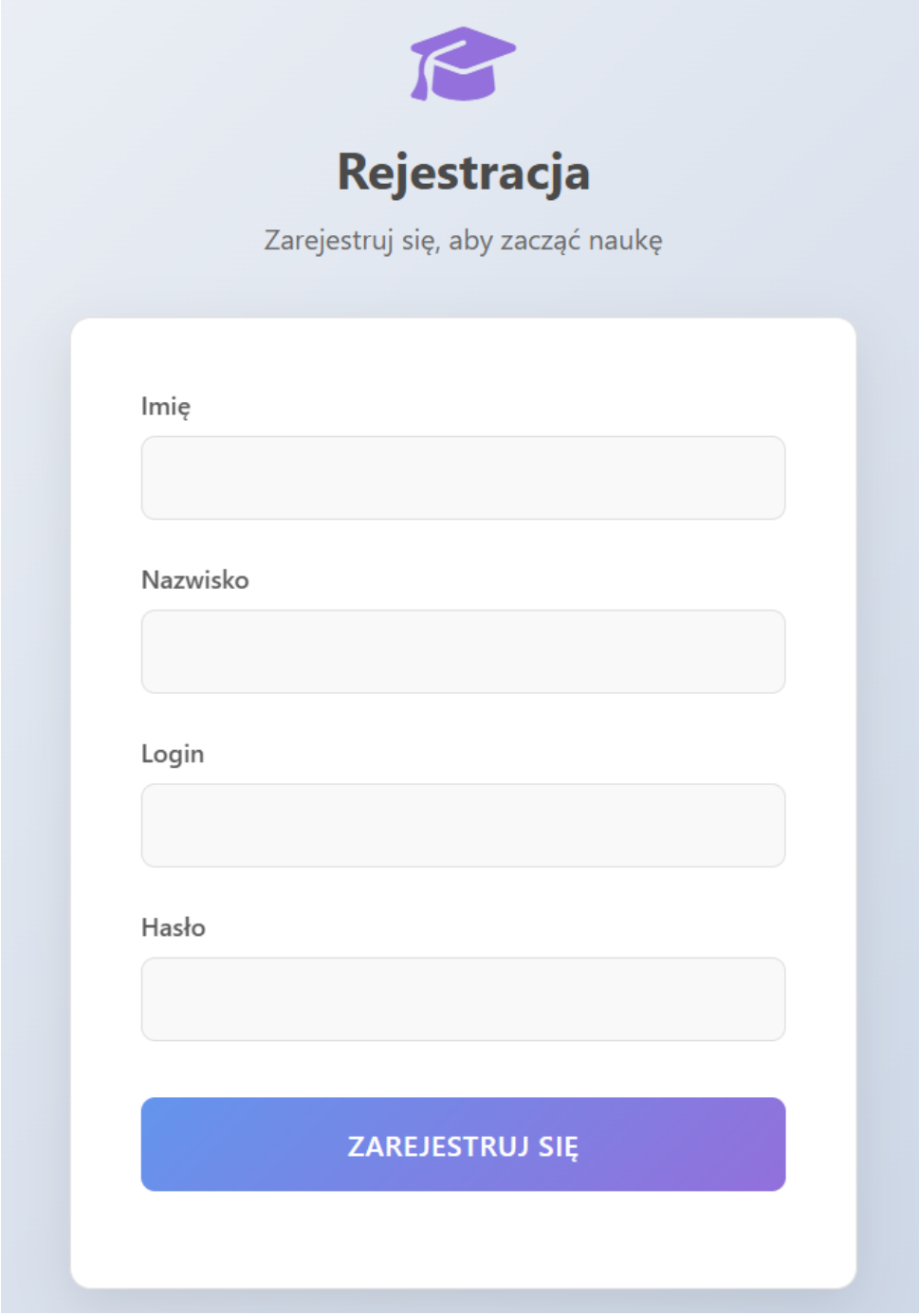
Krok 1: Logowanie do systemu

Po uruchomieniu przeglądarki i otwarciu adresu platformy, użytkownikowi wyświetla się strona z możliwością zalogowania się lub zarejestrowania się.



Rysunek 4.19: Główna, reprezentacyjna strona platformy EduPHP

Po wybraniu opcji *Rejestracja*, użytkownik zostaje przekierowany do formularza, w którym podaje wymagane informacje.



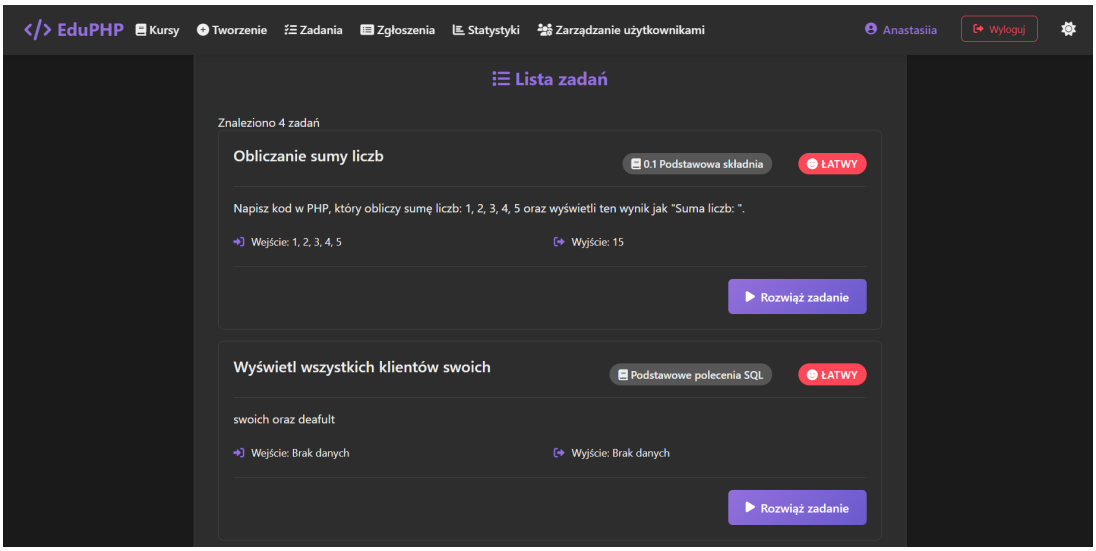
Rysunek 4.20: Formularz rejestracji nowego użytkownika

Po rejestracji następuje autoryzacja, a użytkownik zostaje przeniesiony do formularza logowania. Wprowadza swój login i hasło, po czym klika przycisk *Zaloguj się*. System weryfikuje dane i w przypadku powodzenia przekierowuje użytkownika na główny pulpit.

Rysunek 4.21: Formularz logowania do platformy EduPHP

Krok 2: Przeglądanie dostępnych zadań

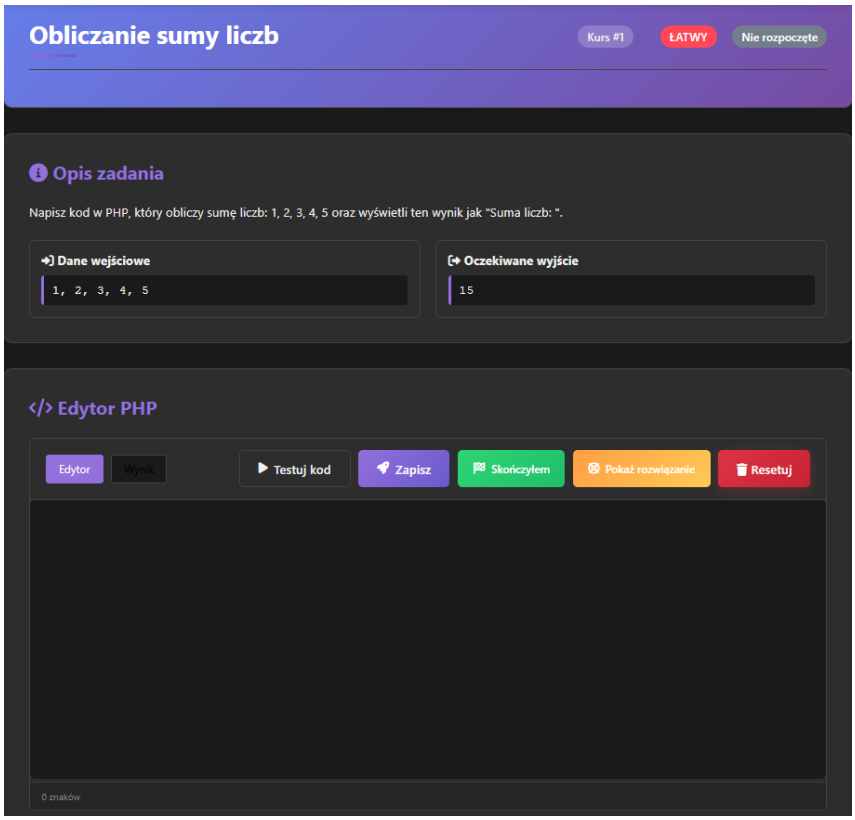
Z głównego pulpitu użytkownik wybiera zakładkę *Zadania* w menu nawigacyjnym; można się do niej również dostać poprzez zakładkę *Kurs*, klikając na przycisk przekierowujący do listy zadań. Otwiera się strona z listą wszystkich dostępnych zadań, które można filtrować według kursu, poziomu trudności oraz statusu ukończenia. Każde zadanie przedstawione jest w postaci kafelka z podstawowymi informacjami: tytuł, przypisany kurs, poziom trudności i status.



Rysunek 4.22: Lista zadań z możliwością filtrowania

Krok 3: Rozpoczęcie rozwiązania zadania

Użytkownik wybiera zadanie obliczanie sumy liczb o łatwym poziomie trudności, klikając przycisk *Rozwiąż zadanie* na odpowiednim kafelku. Otwiera się dedykowany ekran zadania, podzielony na cztery główne sekcje: opis zadania, edytor kodu, panel wynikowy i panel postępu.



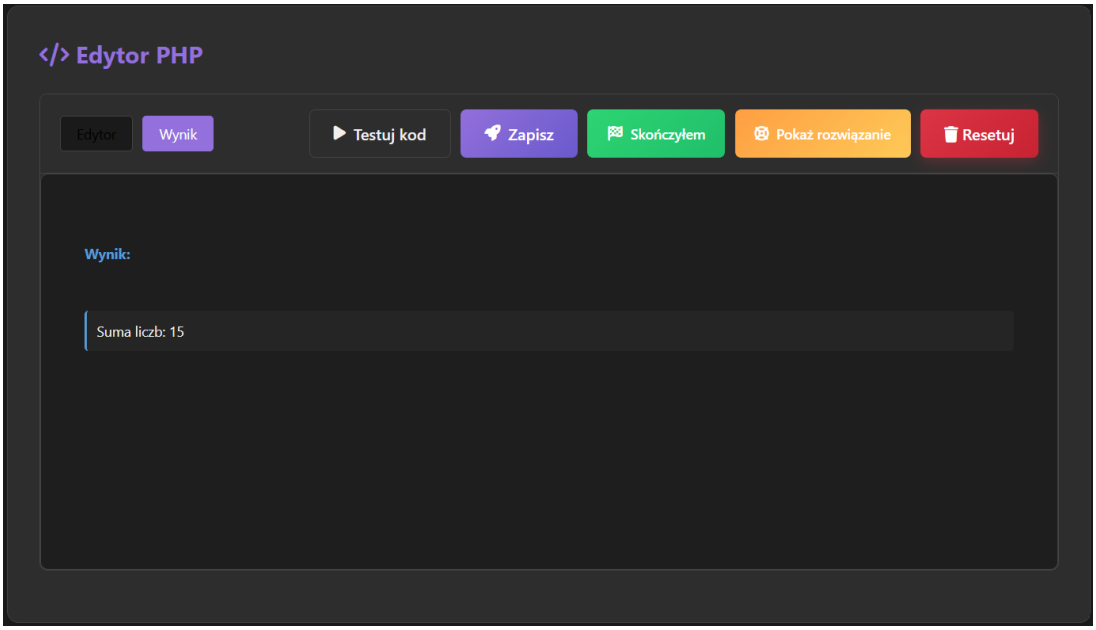
Rysunek 4.23: Ekran rozwiązania zadania z edytorem kodu

Krok 4: Pisanie i testowanie kodu PHP

W edytorze kodu użytkownik wpisuje swoje rozwiązanie:

```
<?php
$liczby = [1, 2, 3, 4, 5];
$suma = array_sum($liczby);
echo "Suma liczb: " . $suma;
?>
```

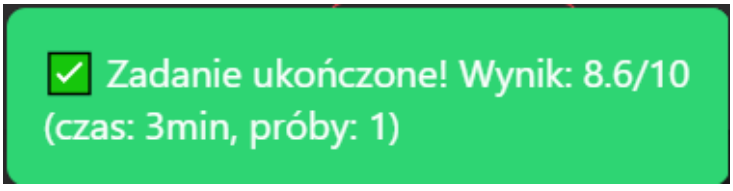
Następnie klika przycisk *Testuj* kod, aby zweryfikować poprawność rozwiązania bez jego ostatecznego zapisania. W panelu wynikowym pojawia się komunikat: "Suma liczb: 15", co potwierdza poprawne działanie kodu.



Rysunek 4.24: Wynik wykonania kodu PHP w konsoli

Krok 5: Zapisanie i ukończenie zadania

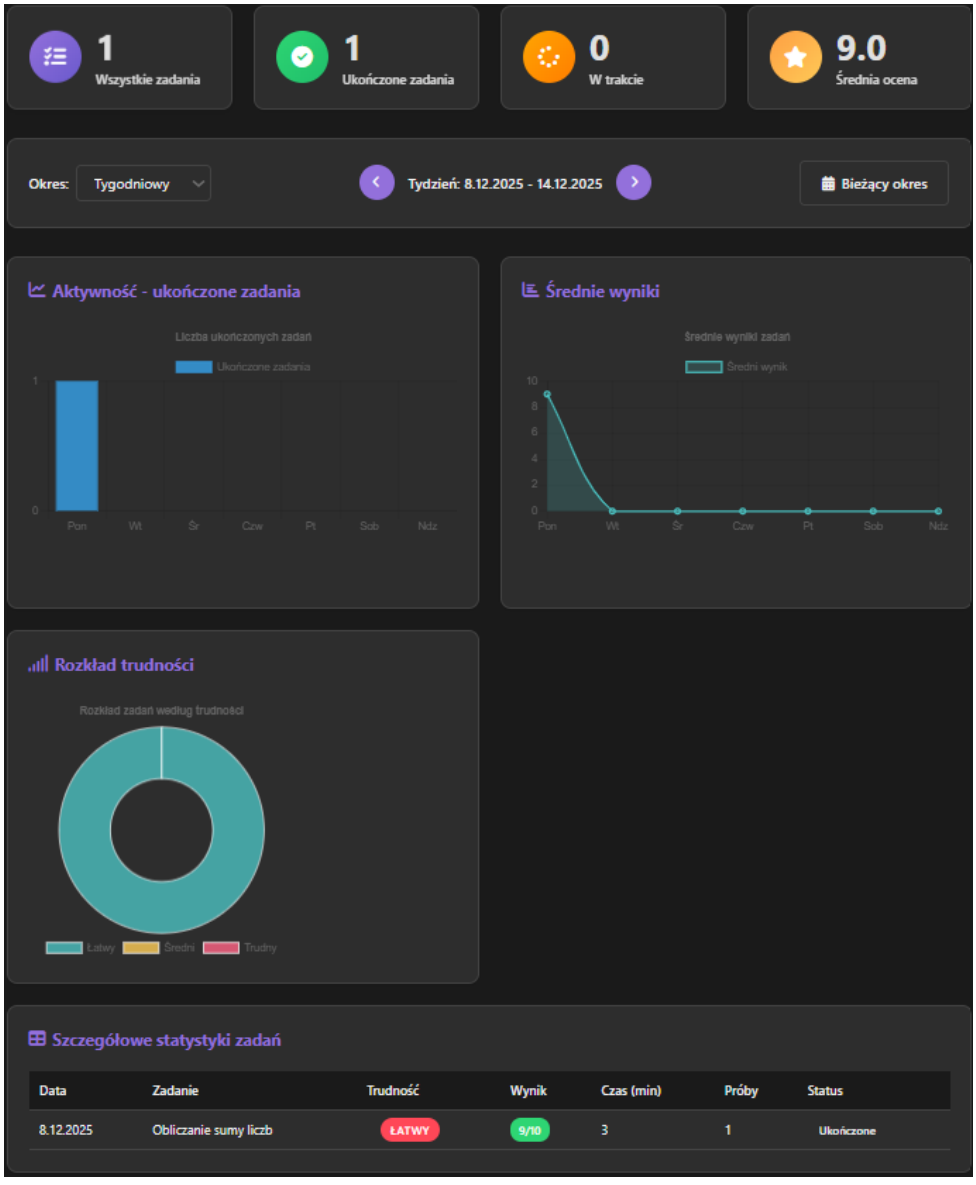
Po pozytywnym przetestowaniu, użytkownik klika przycisk *Zapisz*, co powoduje zapisanie bieżącej wersji kodu. Następnie klika *Skończyłem*, co oznacza ostateczne zgłoszenie rozwiązania. System wyświetla potwierdzenie ukończenia zadania, aktualizuje statystyki użytkownika i przyznaje punkty w zależności od czasu rozwiązania i liczby podjętych prób.



Rysunek 4.25: Potwierdzenie ukończenia zadania

Krok 6: Sprawdzenie postępów

Użytkownik przechodzi do zakładki *Statystyki*. Widzi zaktualizowane wykresy przedstawiające jego postępy: liczbę ukończonych zadań, średni wynik, aktywność w czasie oraz rozkład zadań według trudności.



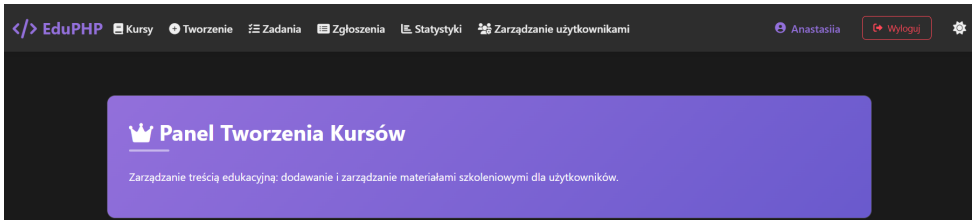
Rysunek 4.26: Panel statystyk użytkownika

4.7.2 Scenariusz 2: Zarządzanie treścią edukacyjną przez administratora

Celem scenariusza jest zademonstrowanie tworzenia nowego materiału edukacyjnego i przypisania do niego zadań praktycznych przez administratora.

Krok 1: Dostęp do panelu administracyjnego

Administrator loguje się do systemu i z menu głównego wybiera *Tworzenie* – dedykowany panel zarządzania treścią edukacyjną.



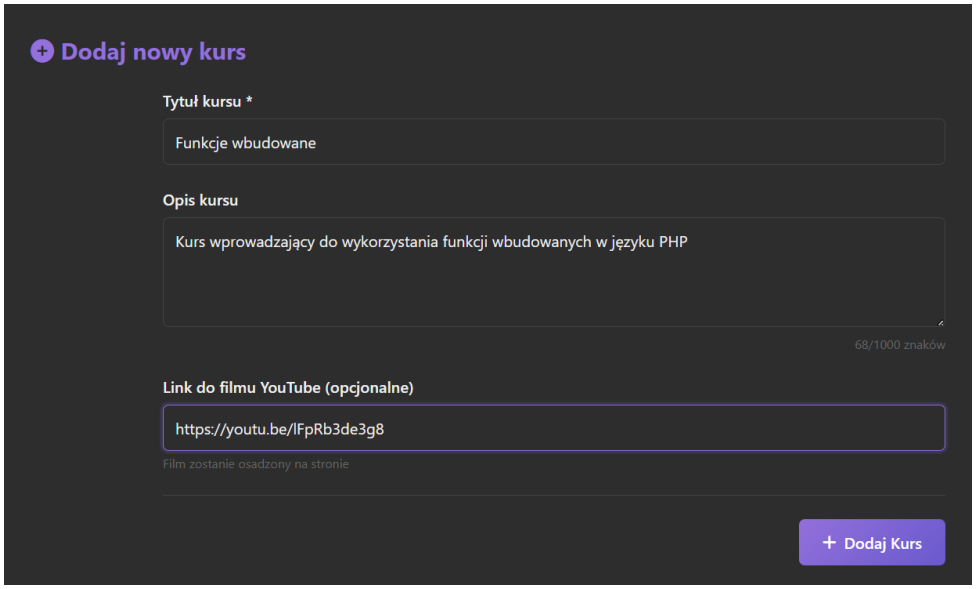
Rysunek 4.27: Panel administracyjny Tworzenie

Krok 2: Utworzenie nowego kursu

W panelu Kursy administrator klika przycisk Dodaj nowy kurs i wypełnia formularz:

- *Tytuł*: Podstawy pętli w PHP,
- *Opis*: Kurs wprowadzający do wykorzystania pętli for, while i foreach w języku PHP,
- *Link do filmu YouTube*: opcjonalny link do materiału wideo na YouTube.

Po kliknięciu Zapisz nowy kurs pojawia się na liście dostępnych kursów.



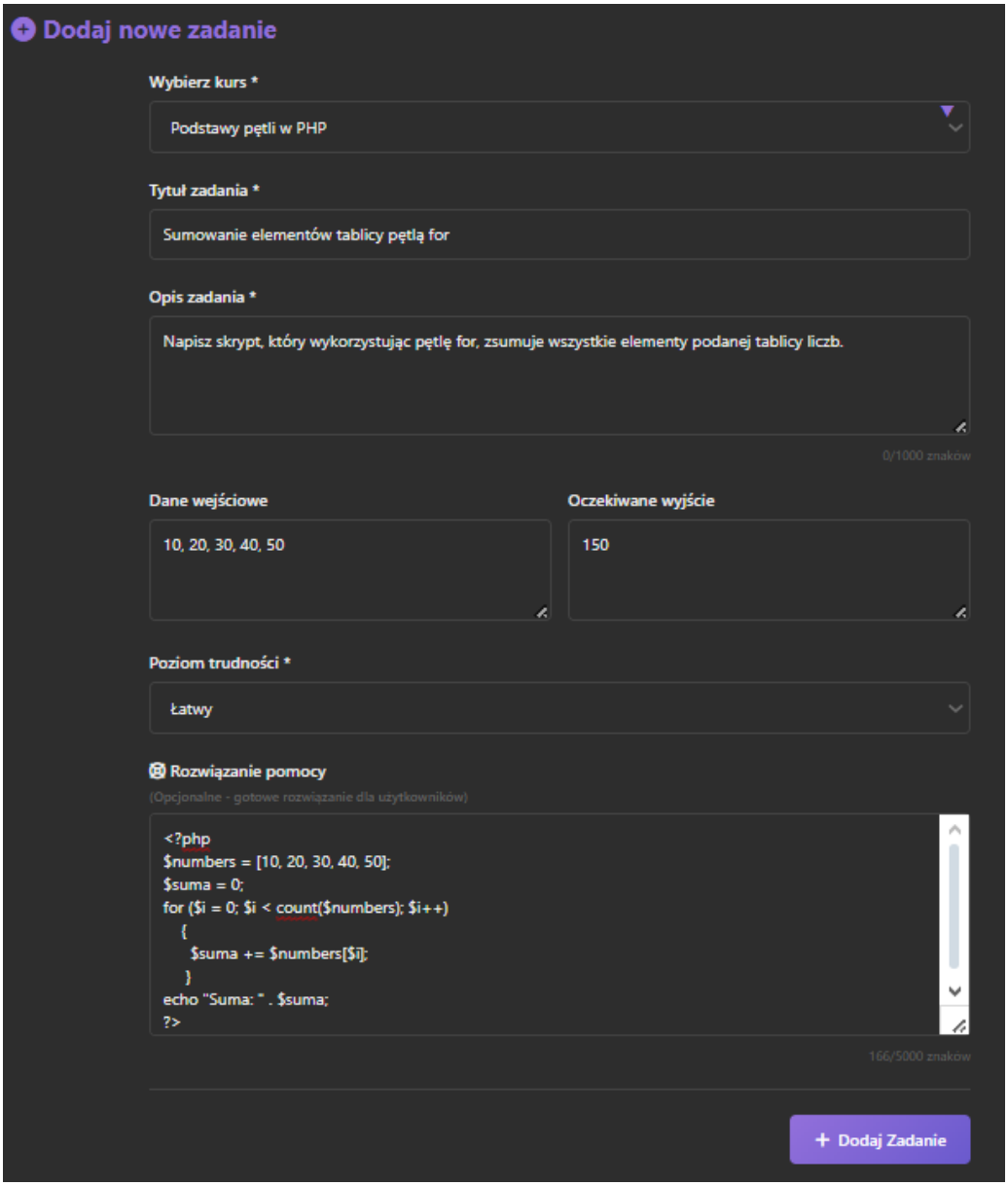
Rysunek 4.28: Formularz dodawania nowego kursu

Krok 3: Dodanie zadania do kursu

Administrator wybiera nowo utworzony kurs i przechodzi do sekcji *Zadania* tego kursu. Klikając *Dodaj zadanie*, otwiera formularz tworzenia zadania:

- *Tytuł*: Sumowanie elementów tablicy pętlą for
- *Opis*: Napisz skrypt, który wykorzystując pętlę for, zsumuje wszystkie elementy podanej tablicy liczb.
- *Dane wejściowe*: $numbers = [10, 20, 30, 40, 50];
- *Oczekiwane wyjście*: Suma: 150
- *Poziom trudności*: Łatwy
- *Rozwiązanie wzorcowe*: opcjonalny kod demonstrujący poprawne rozwiązanie

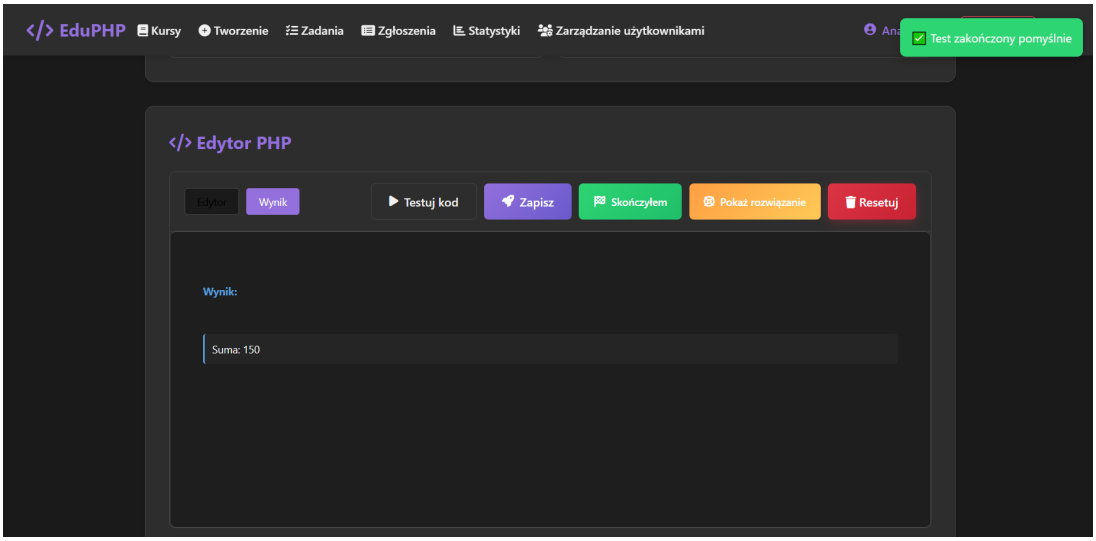
Formularz zostaje zapisany, a zadanie staje się dostępne dla użytkowników zapisanych na ten kurs.



Rysunek 4.29: Formularz dodawania nowego zadania

Krok 4: Weryfikacja funkcjonalności

Administrator przełącza się na widok użytkownika i testuje nowo utworzone zadanie, pisząc i wykonując przykładowy kod PHP w konsoli, aby upewnić się, że wszystkie mechanizmy, czyli wykonanie kodu, walidacja, zapis rozwiązań, działają poprawnie.



Rysunek 4.30: Testowanie zadania przez administratora

4.7.3 Scenariusz 3: Rozwiązywanie problemów poprzez system zgłoszeń

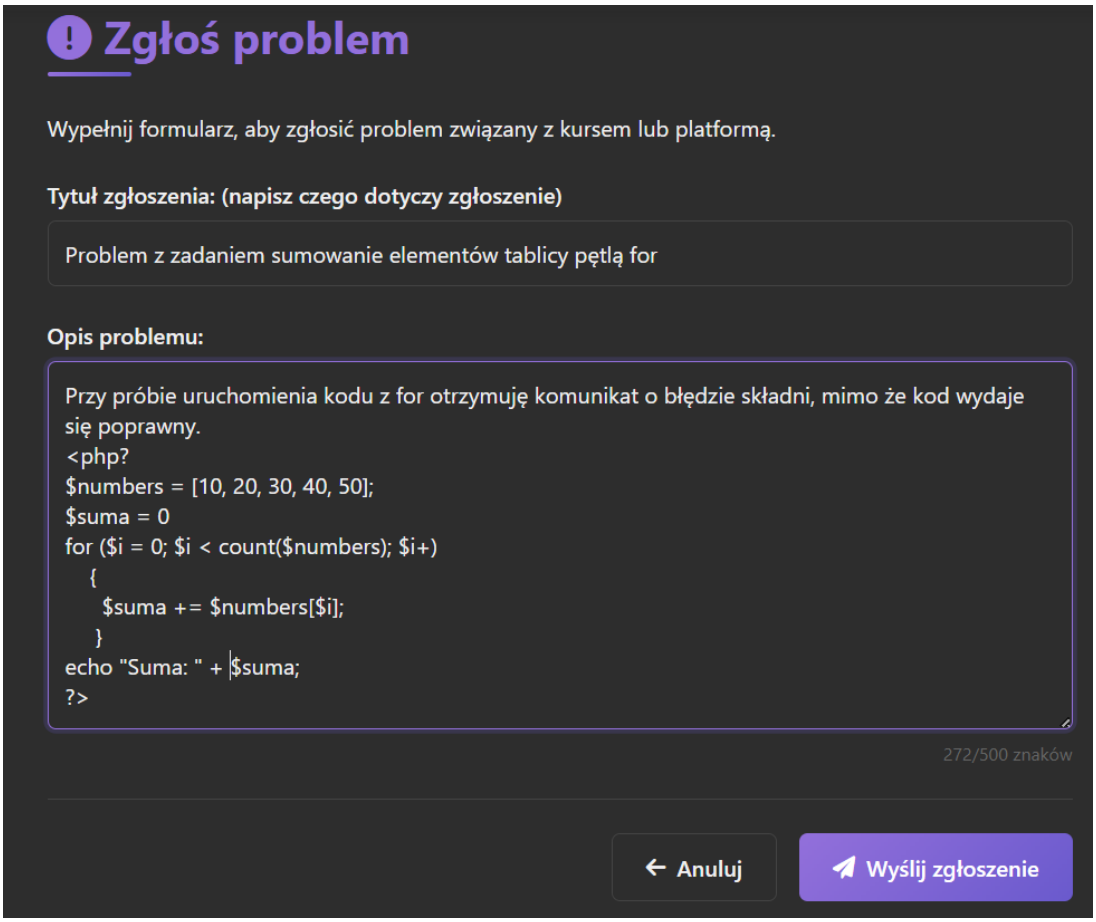
Celem tego scenariusza jest uruchomienie mechanizmu komunikacji między użytkownikami a administratorem przydatnego w przypadku napotkania problemów.

Krok 1: Utworzenie zgłoszenia przez użytkownika

Użytkownik, napotykając problem z zadaniem "sumowanie elementów tablicy pętlą for", przechodzi do zakładki *Zgłoszenia*, klikając *Nowe zgłoszenie*, wypełnia formularz:

- *Tytuł*: "Problem z zadaniem sumowanie elementów tablicy pętlą for"
- *Opis*: "Przy próbie uruchomienia kodu z pętlą for otrzymuję komunikat o błędzie składni, mimo że kod wydaje się poprawny".

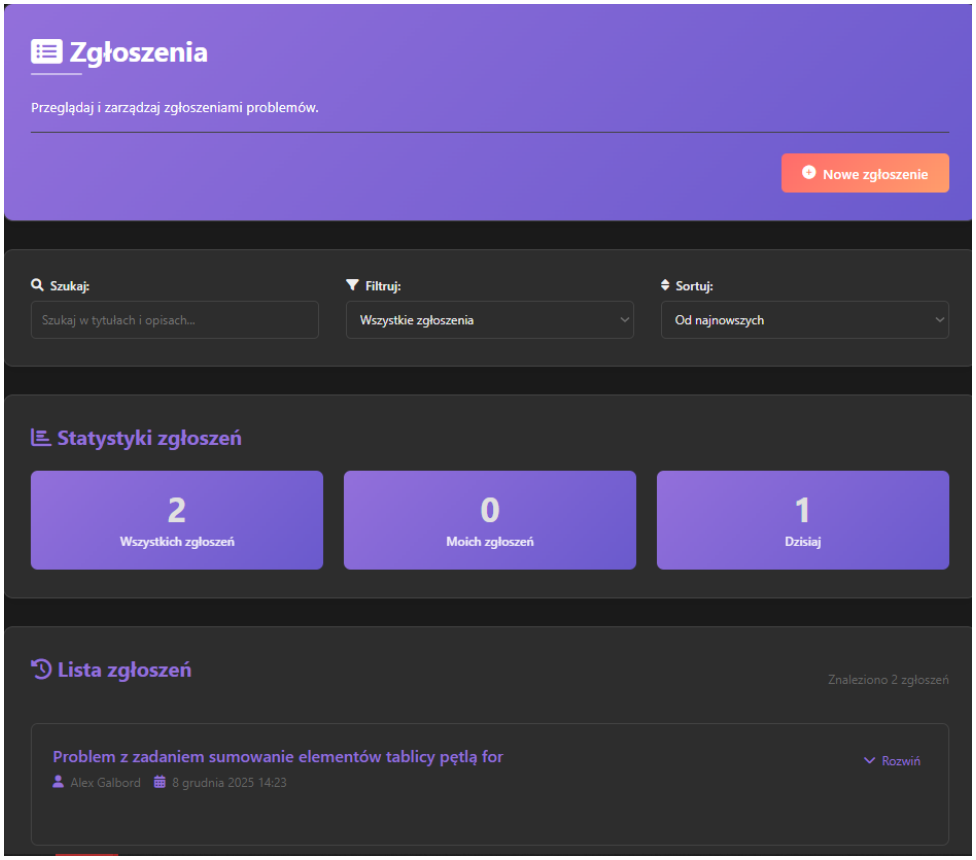
Zgłoszenie zostaje dodane do publicznej listy.



Rysunek 4.31: Formularz tworzenia nowego zgłoszenia

Krok 2: Analiza zgłoszenia przez administratora

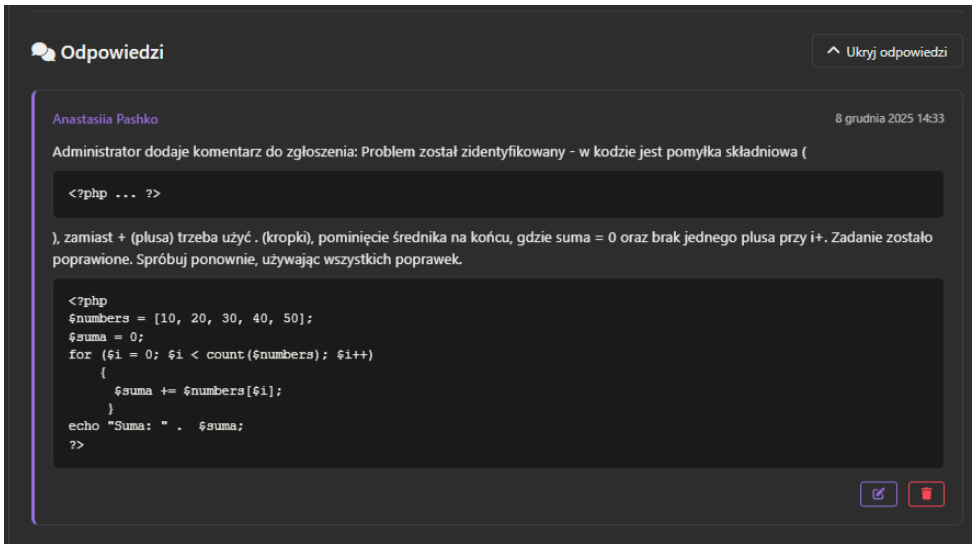
Administrator w panelu zgłoszeń widzi nowe zgłoszenie, otwiera je i analizuje opis problemu. Następnie testuje wskazane zadanie, identyfikując, że problem wynika z brakujących elementów składniowych.



Rysunek 4.32: Lista zgłoszeń - widok administratora

Krok 3: Odpowiedź na zgłoszenie

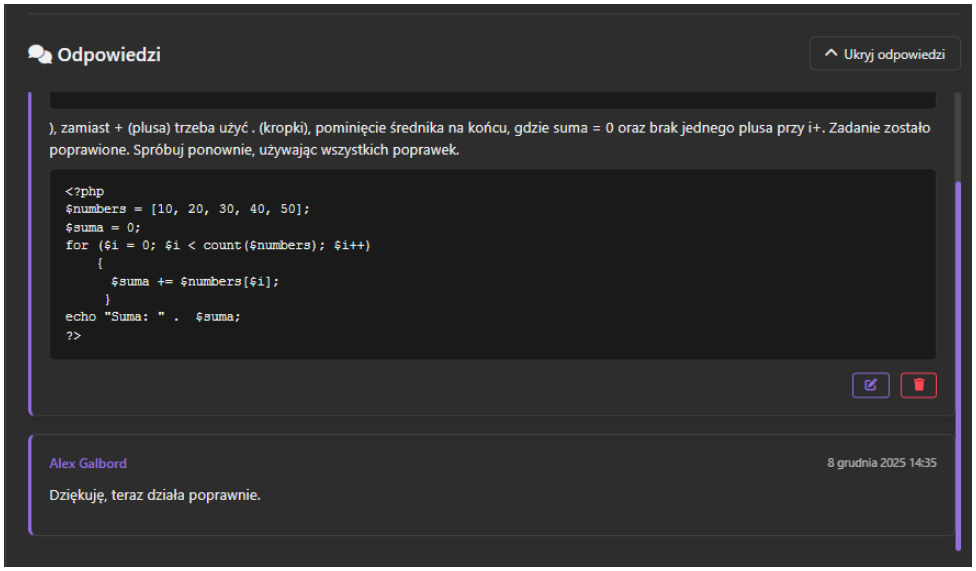
Administrator dodaje komentarz do zgłoszenia: "Problem został zidentyfikowany - w kodzie jest pomyłka składniowa (<?php ... ?>), zamiast + (plusa) trzeba użyć . (kropki), pominięcie średnika na końcu, gdzie suma = 0 oraz brak jednego plusa przy i+. Zadanie zostało poprawione. Spróbuj ponownie, używając wszystkich poprawek".



Rysunek 4.33: Odpowiedź administratora na zgłoszenie

Krok 4: Potwierdzenie rozwiązania przez użytkownika

Użytkownik otrzymuje powiadomienie o aktualizacji zgłoszenia, odczytuje komentarz administratora, ponownie próbuje rozwiązać zadanie z poprawionym kodem i potwierdza, że problem został rozwiązany, dodając komentarz: "Dziękuję, teraz działa poprawnie".



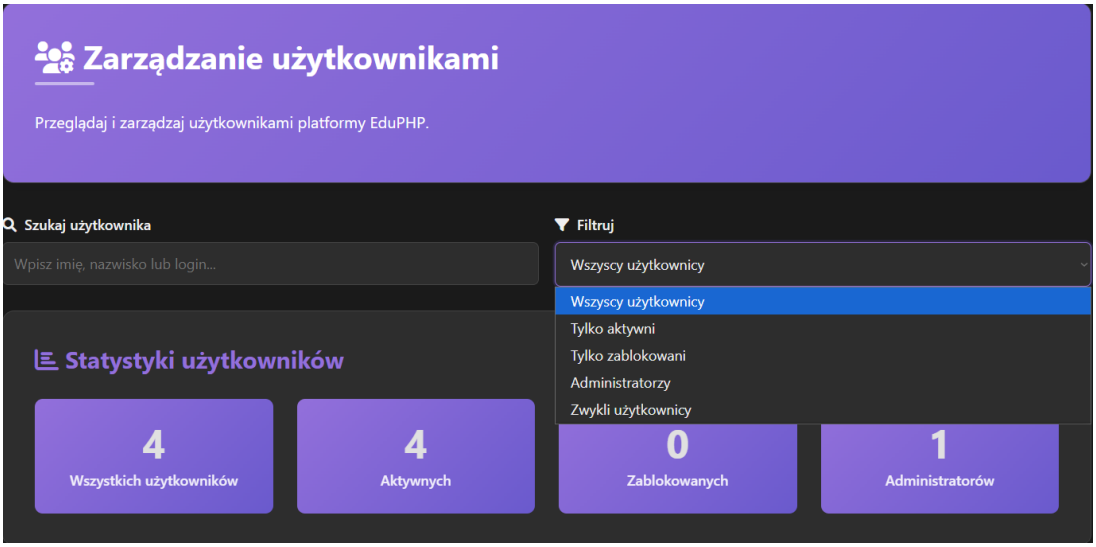
Rysunek 4.34: Potwierdzenie rozwiązania problemu przez użytkownika

4.7.4 Scenariusz 4: Blokowanie użytkownika przez administratora

Celem tego scenariusza jest zademonstrowanie procesu zarządzania kontami użytkowników przez administratora, w szczególności mechanizmu blokowania dostępu do platformy w przypadku niestosownego zachowania lub podejrzeń o złośliwe działanie.

Krok 1: Dostęp do panelu zarządzania użytkownikami

Administrator, po zalogowaniu się do systemu, wybiera z menu głównego pozycję *Zarządzanie użytkownikami*. Otwiera się dedykowany panel administracyjny, w którym widoczna jest lista wszystkich zarejestrowanych użytkowników platformy.



Rysunek 4.35: Panel zarządzania użytkownikami z listą wszystkich kont

Krok 2: Przeglądanie i filtrowanie listy użytkowników

W panelu dostępne są opcje filtrowania użytkowników według różnych kryteriów:

- *Wszyscy* - wyświetla wszystkich zarejestrowanych użytkowników,
- *Aktywni* - pokazuje tylko użytkowników z aktywnym dostępem do platformy,
- *Zablokowani* - wyświetla użytkowników, którym zablokowano dostęp,
- *Administratorzy* - lista użytkowników z uprawnieniami administracyjnymi,
- *Zwykli użytkownicy* - użytkownicy z rolą USER.

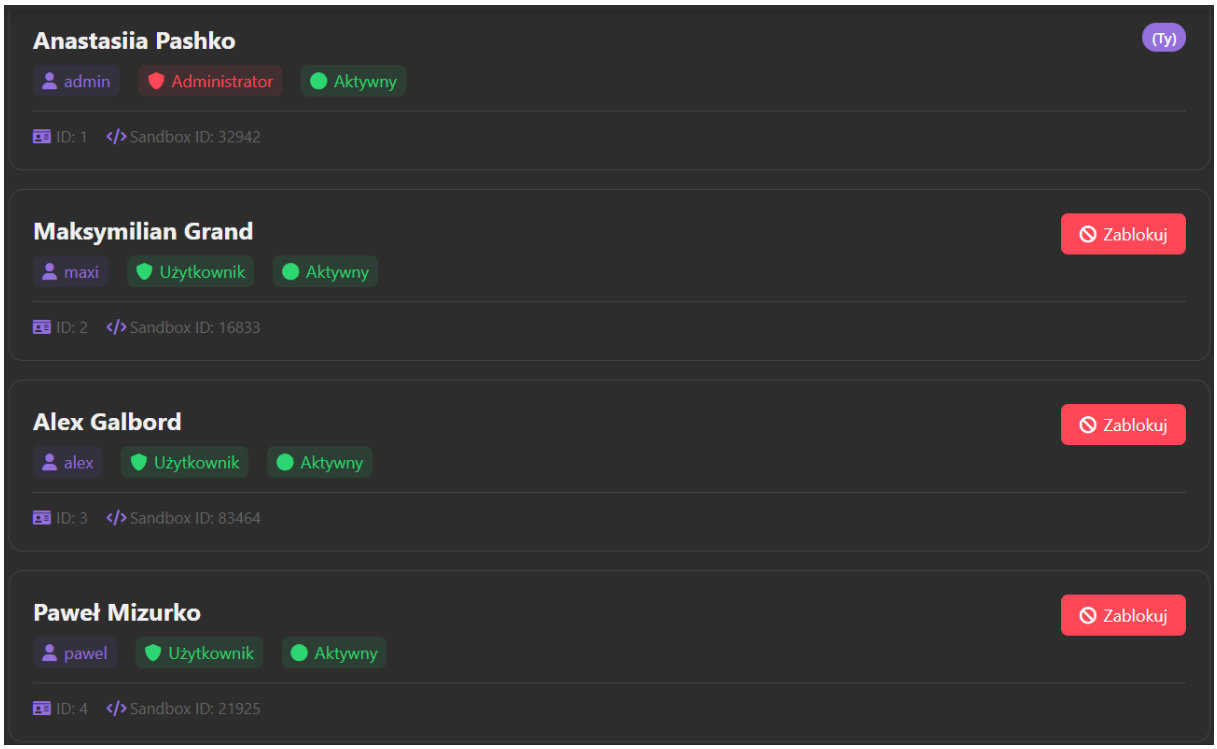
Dla każdego użytkownika na liście widoczne są podstawowe informacje: imię i nazwisko, login, data rejestracji, status konta (aktywny/zablokowany) oraz liczba ukończonych zadań.

Krok 3: Identyfikacja użytkownika do zablokowania

Administrator zauważa użytkownika, który wielokrotnie próbował uruchamiać potencjalnie niebezpieczny kod w konsoli PHP, zawierający próby wywołania zablokowanych funkcji systemowych lub nieskończone pętle. Po przeanalizowaniu historii aktywności użytkownika, administrator podejmuje decyzję o zablokowaniu konta.

Krok 4: Wykonanie blokady

Administrator znajduje odpowiedniego użytkownika na liście i klika przycisk *Zablokuj* w kolumnie akcji. System wyświetla okno dialogowe z prośbą o potwierdzenie:



Rysunek 4.36: Stan konta użytkownika przed blokadą

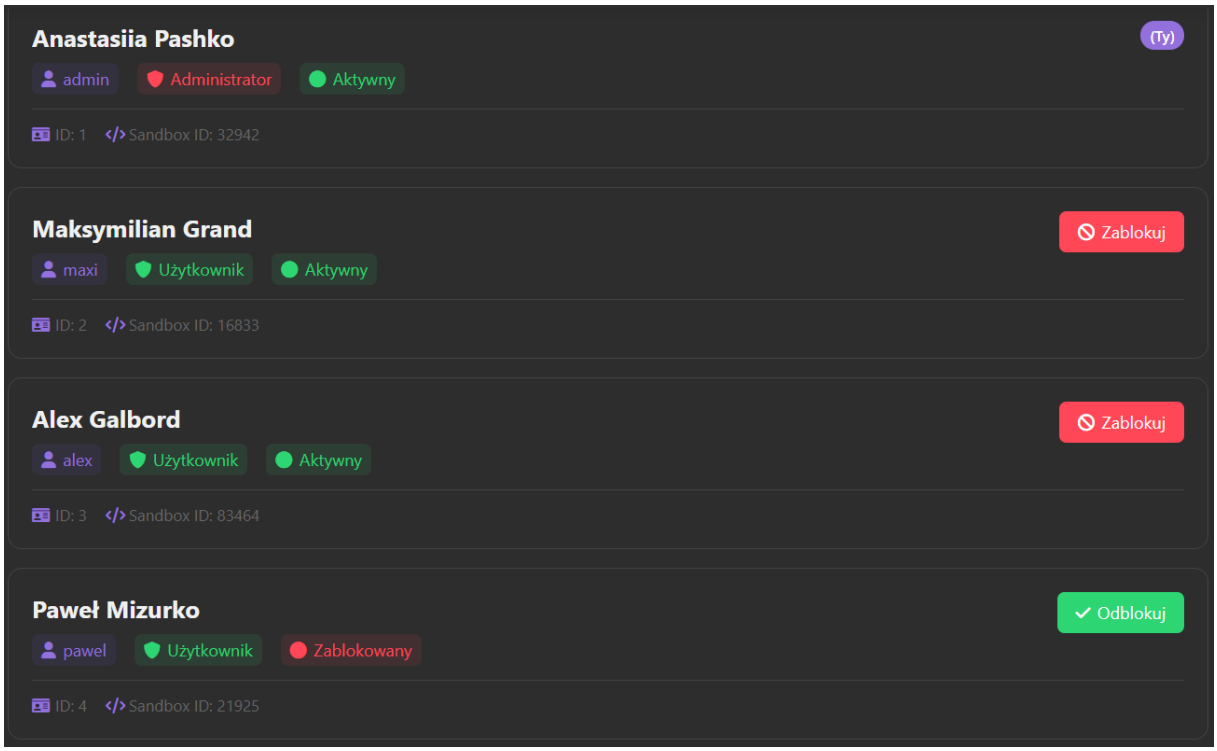
> **Czy na pewno chcesz zablokować użytkownika?**
> Zablokowany użytkownik nie będzie mógł się zalogować do systemu aż do odblokowania przez administratora.

Administrator potwierdza operację, klikając *Ok*.

Krok 5: Weryfikacja zmiany statusu

Po potwierdzeniu, system aktualizuje status użytkownika na liście w bazie danych - zmienia się z 1 na 0, co odpowiednio oznacza ze statusu *Aktywny* na *Zablokowany*. W kolumnie akcji przycisk *Zablokuj* zmienia się na *Odblokuj*. Administrator otrzymuje potwierdzenie wykonania operacji w postaci komunikatu:

> **Sukces:** Użytkownik został zablokowany.



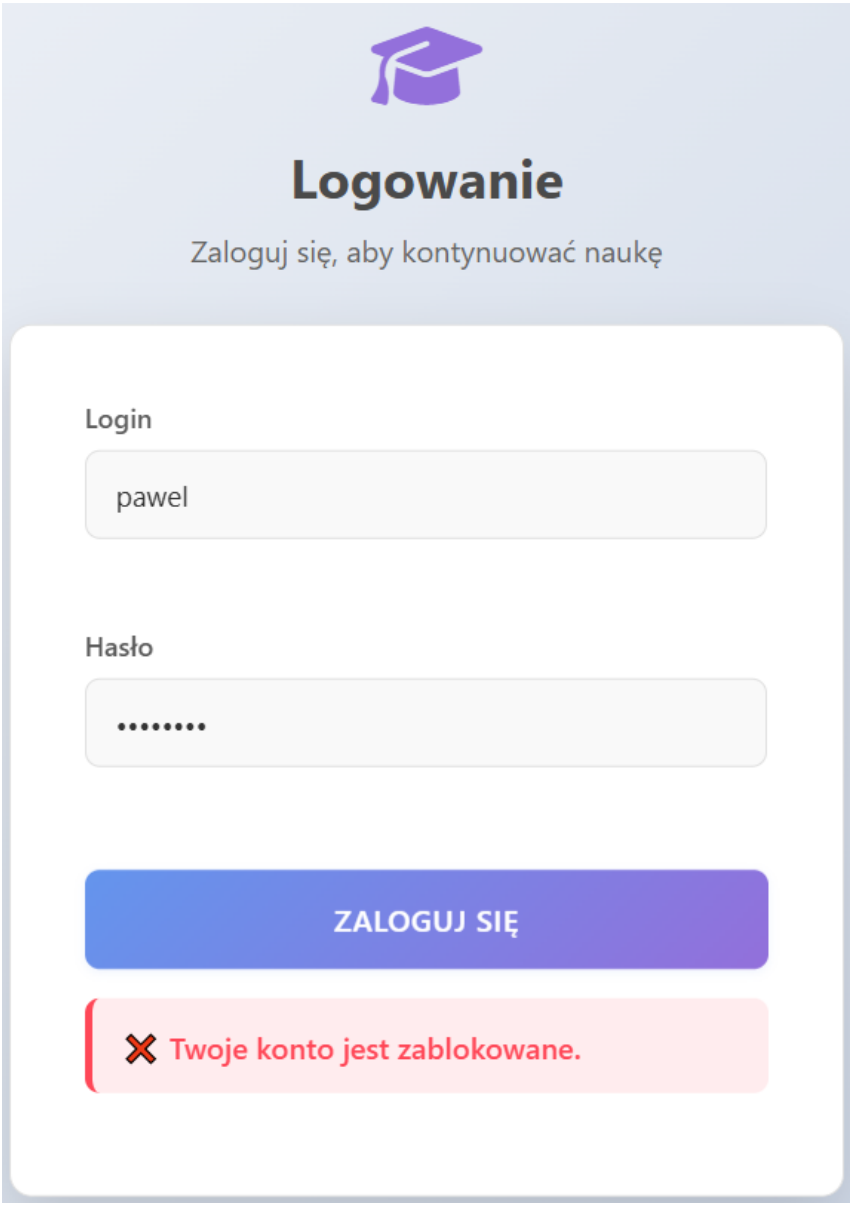
Rysunek 4.37: Lista użytkowników ze zaktualizowanym statusem

Krok 6: Próba logowania zablokowanego użytkownika

Zablokowany użytkownik próbuje zalogować się do systemu, używając swojego loginu i hasła. Zamiast przekierowania na dashboard, system wyświetla komunikat:

Dostęp zablokowany: *Twoje konto jest zablokowane.*

Użytkownik nie ma dostępu do żadnych funkcjonalności platformy aż do odblokowania konta.



Rysunek 4.38: Komunikat o zablokowanym koncie przy próbie logowania

Krok 7: Odblokowanie użytkownika

Jeśli użytkownik wyjaśnił sytuację lub błąd został naprawiony, administrator może odblokować konto. W panelu zarządzania użytkownikami, administrator znajduz zablo-

kowanego użytkownika i klika przycisk *Odblokuj*. Po potwierdzeniu, status użytkownika zmienia się z powrotem na *Aktywny*, a użytkownik odzyskuje pełny dostęp do platformy.

4.7.5 Podsumowanie scenariuszy

Przedstawione scenariusze pokazują kompletny cykl użytkowania platformy EduPHP:

- Od nauki poprzez interaktywne rozwiązywanie zadań (Scenariusz 1)
- Przez zarządzanie treścią edukacyjną (Scenariusz 2)
- Aż do rozwiązywania problemów poprzez zintegrowany system wsparcia (Scenariusz 3)
- Aż po zarządzanie bezpieczeństwem i dostępem użytkowników (Scenariusz 4).

Platforma zapewnia spójne środowisko pracy zarówno dla uczących się programowania użytkowników, jak i dla administratorów zarządzających treścią oraz wspierających proces edukacyjny. Wszystkie funkcjonalności są zintegrowane w jednolity interfejs, co ułatwia nawigację i korzystanie z systemu.

Rozdział 5

Specyfikacja wewnętrzna

Niniejszy rozdział przedstawia wewnętrzną specyfikację techniczną opracowanego systemu, koncentrując się na architekturze, strukturze danych oraz szczegółach implementacyjnych platformy edukacyjnej opracowanej w niniejszej pracy. W kolejnych sekcjach szczegółowo omówiono kluczowe aspekty systemu, w tym jego architekturę, organizację baz danych, główne komponenty i moduły, przegląd klas, implementowane algorytmy oraz zastosowane wzorce projektowe. Celem rozdziału jest przedstawienie kompleksowego opisu technicznego, umożliwiającego zrozumienie wewnętrznej struktury systemu, zasad jego działania oraz zastosowanych rozwiązań inżynierskich.

5.1 Idea systemu EduPHP

System EduPHP został zaprojektowany jako kompleksowa platforma edukacyjna dedykowana nauce programowania w języku PHP, z naciskiem na praktyczne aspekty tworzenia aplikacji webowych. Główną motywacją do stworzenia tej platformy były wnioski wynikające z analizy istniejących rozwiązań internetowych, które, według autorki, nie spełniają wszystkich wymagań niezbędnych do komfortowego i efektywnego uczenia się. Kluczowym elementem systemu jest implementacja zaawansowanego mechanizmu sandbox, umożliwiającego bezpieczne wykonywanie kodu PHP napisanego przez użytkowników, przy jednoczesnym zapewnieniu pełnej izolacji danych pomiędzy różnymi uczestnikami platformy.

Podstawową ideę systemu można zdefiniować przez trzy filary funkcjonalności:

1. *Bezpieczne i izolowane środowisko wykonawcze* – umożliwienie użytkownikom pisania, testowania i wykonywania dowolnego kodu PHP bez ryzyka naruszenia stabilności systemu głównego, uszkodzenia danych innych użytkowników lub wystąpienia konfliktów w dostępie do zasobów.

2. *Zautomatyzowana weryfikacja postępów edukacyjnych* – implementacja systemu automatycznej oceny poprawności rozwiązań programistycznych poprzez porównanie

wyników wykonania kodu użytkownika z oczekiwanymi danymi wyjściowymi zdefiniowanymi dla każdego zadania; w obecnej wersji systemu weryfikacja odbywa się samodzielnie przez użytkownika.

3. *Kompleksowe śledzenie i analiza postępów* – stworzenie zaawansowanego systemu monitorowania rozwoju kompetencji programistycznych użytkowników, obejmującego śledzenie postępów w kursach, analizę statystyk rozwiązywania zadań oraz zastosowanie algorytmicznego systemu obliczania punktacji uwzględniającego zarówno czas wykonania, jak i liczbę prób, przy czym czas ma większe znaczenie w proporcji 0.7:0.3.

Platforma rozwiązuje istotny problem edukacji programistycznej – brak bezpiecznego środowiska do eksperymentowania z kodem w kontekście pracy z bazami danych. Tradycyjne podejścia do nauki PHP często narażają początkujących programistów na ryzyko uszkodzenia środowisk deweloperskich lub konfliktów danych. Platforma rozwiązuje ten problem poprzez implementację wielowarstwowych zabezpieczeń i automatyczną izolację danych, umożliwiając skupienie się na nauce programowania bez obaw o konsekwencje błędów w kodzie.

Kluczowym aspektem systemu jest jego modularna konstrukcja, pozwalająca na łatwe rozszerzanie funkcjonalności oraz adaptację do różnych scenariuszy edukacyjnych. Platforma została zaprojektowana z uwzględnieniem potrzeb początkujących programistów, rozpoczynających swoją przygodę z PHP, zapewniając im bezpieczne i wspierające środowisko do nauki.

5.2 Architektura systemu

System został zrealizowany w oparciu o architekturę klient-serwer z rozdzieleniem odpowiedzialności pomiędzy warstwą frontend a backend. Takie podejście, zgodne z wymaganiami określonymi w rozdziale 3, umożliwia niezależny rozwój poszczególnych komponentów systemu, zapewniając jednocześnie elastyczność i skalowalność rozwiązania.

5.2.1 Architektura klient-serwer z separacją warstw

Architektura systemu EduPHP opiera się na separacji pomiędzy następującymi warstwami:

- Warstwa kliencka: aplikacja webowa wykonywana w przeglądarce użytkownika, zaimplementowana w JavaScript (ES6+) bez wykorzystania zewnętrznych frameworków.
- Warstwa serwerowa: aplikacja Spring Boot udostępniająca REST API.

- Warstwa danych: baza danych MySQL z podziałem na bazę główną i sandbox.

Taka architektura, często określana jako "Thin Client"[13], umożliwia:

- Niezależne testowanie i rozwój stron frontend i backend.
- Łatwą wymianę technologii w przyszłości.
- Lepsze zabezpieczenia – logika biznesowa po stronie serwera.
- Optymalizację wydajności poprzez minimalizację przesyłanych danych.

5.2.2 Warstwa prezentacji - Frontend

Warstwa prezentacji odpowiada za interakcję z użytkownikiem końcowym i prezentację danych zgodnie z wymaganiami użyteczności wymienionymi w rozdziale 3.

- Technologia: JavaScript (ES6+) bez frameworków frontend;
- Struktura: HTML5 dla struktury, CSS3 dla stylizacji, JavaScript dla logiki;
- Komunikacja z backend: Fetch API do wykonywania zapytań HTTP do REST API;
- Architektura frontend:
  - Moduły ES6 dla organizacji kodu;
  - Obsługa asynchronicznych operacji poprzez async/await;
  - Dynamiczne renderowanie komponentów interfejsu;
  - Walidacja formularzy po stronie klienta.
- Responsywność: Zapewnienie poprawnego działania na różnych rozdzielczościach ekranu.

5.2.3 Warstwa serwerowa - Backend

Warstwa serwerowa implementuje logikę biznesową i udostępnia interfejs programistyczny dla strony frontend.

- Technologia: Spring Boot 3.2.x z Java 21;
- Architektura API: REST zgodnie z wymaganiami z rozdziału 3;
- Struktura warstwowa backend:
  - Kontrolery (Controller Layer): Obsługa żądań HTTP, walidacja wejścia;

  - Serwisy (Service Layer): Implementacja logiki biznesowej;
  - Repozytoria (Repository Layer): Dostęp do bazy danych przez JPA;
  - Encje (Entity Layer): Mapowanie obiektowo-relacyjne;
  - DTO (Data Transfer Objects): Transfer danych między warstwami;
- Zabezpieczenia: Spring Security z BCrypt do hashowania haseł;
- Transakcyjność: Zarządzanie transakcjami przez @Transactional.

5.2.4 Warstwa danych - Data Layer

System wykorzystuje dwie oddzielne bazy danych MySQL, co zapewnia izolację i bezpieczeństwo.

- Baza główna (eduphp): Przechowuje dane systemowe:
  - dane użytkowników i profili,
  - strukturę kursów i zadań,
  - postępy użytkowników i statystyki,
  - zgłoszenia i odpowiedzi.
- Baza sandbox (eduphp_sandbox): Izolowane środowisko wykonawcze:
  - tabele ćwiczeniowe (orders, customers, products),
  - izolacja danych per użytkownik przez kolumnę user_id, którego identyfikator jest losowo generowany dla większego bezpieczeństwa;
  - Ograniczone uprawnienia użytkownika bazy.
- Technologia ORM: JPA/Hibernate dla mapowania obiektowo-relacyjnego;
- Generowanie zapytań: Spring Data JPA z metodami nazwanymi.

5.2.5 Komponent sandbox – bezpieczne środowisko wykonawcze

Komponent sandbox to najbardziej zaawansowany technicznie komponent systemu, realizujący kluczowe wymaganie bezpiecznego wykonania kodu PHP.

- Mechanizm wykonania: Izolowany proces PHP uruchamiany przez ProcessBuilder – klasę z biblioteki standardowego języka Java, umożliwiającą tworzenie i kontrolowanie zewnętrznych procesów systemowych. ProcessBuilder tworzy oddzielny proces systemowy dla interpretera PHP, co zapewnia fizyczną izolację kodu użytkownika od głównej aplikacji Java.

- Architektura sandbox:
  - Tworzenie tymczasowych plików PHP z kodem użytkownika;
  - Dodawanie automatycznych zabezpieczeń (security wrappers) - automatyczne opakowywanie kodu użytkownika w warstwę bezpieczeństwa, która wyłącza niebezpieczne funkcje i dodaje mechanizmy izolacji danych;
  - Wykonanie kodu z limitem czasowym (10 sekund) - mechanizm zabezpieczający przed zawieszeniem się kodu;
  - Przechwytywanie wyjścia i błędów - oddzielne strumienie dla standardowego wyjścia i komunikatów o błędach.
- Izolacja danych: Automatyczne modyfikowanie zapytań SQL:
  - SELECT → dodanie WHERE user_id = X LIMIT 100 - zapobiega wyciekowi danych innych użytkowników i ogranicza obciążenie bazy;
  - INSERT → automatyczne dodanie user_id - zapewnia, że dane są przypisane do właściwego użytkownika;
  - UPDATE/DELETE → dodanie warunku user_id = X - zapobiega modyfikacji lub usunięciu danych innych użytkowników.
- Bezpieczeństwo:
  - Wyłączenie niebezpiecznych funkcji PHP (eval, system, exec, passthru, shell_exec) - zapobiega wykonywaniu poleceń systemowych;
  - Ograniczenia zasobów (pamięć, czas CPU) - zabezpieczenie przed atakami typu denial-of-service (DoS), które polegają na celowym przeciążeniu systemu poprzez wykonywanie operacji zużywających duże ilości zasobów, takich jak nieskończone pętle lub rekurencje bez warunku zakończenia;
  - Izolacja systemu plików - kod użytkownika ma dostęp tylko do własnych plików tymczasowych.

5.2.6 Komunikacja między komponentami

Komunikacja w systemie odbywa się zgodnie ze wzorcem REST:

- Frontend → Backend: Żądania HTTP (GET, POST, PUT, DELETE) z danymi w formacie JSON;
- Backend → Frontend: Odpowiedzi HTTP z danymi JSON lub kodami statusu;
- Backend → Baza danych: Połączenie JDBC przez JPA/Hibernate;

- Sandbox → Baza sandbox: Połączenie MySQL z izolacją przez zabezpieczone wrappery - specjalnie wygenerowany kod PHP, który modyfikuje zapytania SQL w locie przed ich wykonaniem.

5.2.7 Uzasadnienie wyboru określonej architektury

System zachowuje fundamentalne zasady architektury warstwowej:

1. Separacja odpowiedzialności - każdy komponent ma określone zadania:
    - Frontend: Prezentacja i interakcja z użytkownikiem;
    - Backend: Logika biznesowa i przetwarzanie danych;
    - Baza danych: Przechowywanie i zarządzanie danymi.
2. Niezależność warstw - możliwość modyfikacji jednej warstwy bez wpływu na pozostałe;
3. Ponowne użycie - logika biznesowa może być wykorzystana przez różne rozwiązania klienckie (web, mobile, API);
4. Testowalność - każda warstwa może być testowana niezależnie.

Kluczowym elementem architektury jest separacja frontend (JavaScript) od backend (Spring Boot REST API) oraz podział baz danych na główną i sandbox, co zapewnia zarówno bezpieczeństwo, jak i elastyczność systemu. Wykorzystanie ProcessBuildera do uruchamiania izolowanych procesów PHP stanowi fundament bezpieczeństwa całego systemu, pozwalając na całkowite odseparowanie wykonywanego kodu użytkownika od głównej aplikacji.

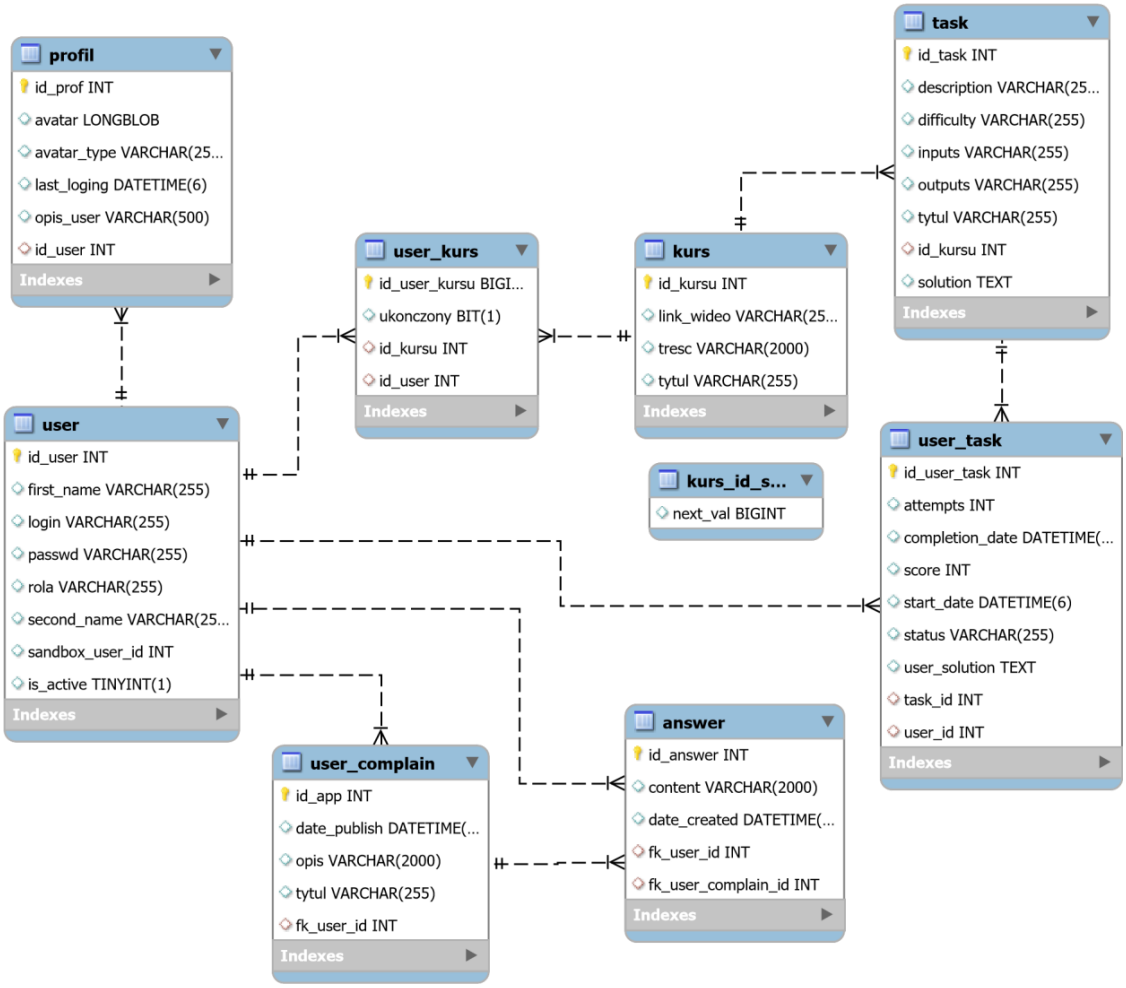
5.3 Organizacja danych i struktury bazodanowe

W niniejszej części przedstawiono architekturę danych, obejmującą zarówno główną bazę danych przechowującą informacje systemowe, jak i specjalizowane środowisko sandbox służące do bezpiecznego wykonywania kodu tworzonego przez uczestników platformy. Struktura ta została zaprojektowana z uwzględnieniem wymagań dotyczących izolacji, bezpieczeństwa i efektywności przetwarzania.

5.3.1 Diagram relacji encji głównej bazy danych

Architektura głównej bazy danych systemu została przedstawiona na rysunku 5.1, który ilustruje złożone powiązania pomiędzy encjami odpowiadającymi za różne aspekty funkcjonowania platformy edukacyjnej. Diagram ten odzwierciedla relacje umożliwiające

efektywne zarządzanie użytkownikami, kursami, zadaniami oraz kompleksowe śledzenie postępów w nauce programowania.



Rysunek 5.1: Diagram relacji encji głównej bazy danych systemu EduPHP

5.3.2 Charakterystyka głównych encji systemowych

Encja użytkownika (User)

Podstawowa jednostka systemu przechowująca informacje o uczestnikach platformy. Każdy użytkownik posiada unikalny identyfikator (idUser), login służący do uwierzytelniania oraz zaszyfrowane hasło. Encja ta zawiera również pole sandboxUserId, które stanowi losowo generowany identyfikator wykorzystywany do izolacji danych w środowisku sandbox. Flaga isActive umożliwia administratorom zarządzanie dostępem do platformy. Dodatkowo encja przechowuje podstawowe dane personalne użytkownika, w tym imię (first_name), nazwisko (second_name) oraz przypisaną rolę systemową (user lub admin).

Encja profilu użytkownika (Profil)

Rozszerzenie informacji o uczestniku platformy, zawierające dane personalizacyjne. Encja przechowuje obraz profilowy w formacie LONGBLOB wraz z informacją o typie MIME. Pole opisUser umożliwia użytkownikom dodanie krótkiej autoprezentacji, natomiast lastLoging służy do śledzenia aktywności poprzez zapis znacznika czasu ostatniego logowania.

Encja kursu edukacyjnego (Kurs)

Reprezentacja materiału dydaktycznego organizowanego w formie strukturalnego kursu. Encja zawiera tytuł kursu, treść merytoryczną ograniczoną do 2000 znaków oraz opcjonalny odnośnik do materiału wideo. Każdy kurs może zawierać wiele powiązanych zadań programistycznych.

Encja zadania programistycznego (Task)

Definicja ćwiczenia praktycznego przypisanego do konkretnego kursu. Encja zawiera szczegółowy opis wymagań, przykładowe dane wejściowe, oczekiwane wyniki oraz poziom trudności. Pole solution przechowuje wzorcowe rozwiązanie, które może być udostępniane użytkownikom jako pomoc edukacyjna, pod warunkiem rezygnacji z punktacji za dane zadanie.

Encja rozwiązania użytkownika (UserTask)

Śledzenie postępów uczestników w realizacji zadań. Encja przechowuje kod napisany przez użytkownika wraz z metrykami dotyczącymi efektywności nauki, w tym liczbą podejść oraz wynikiem w skali 0-10. Status zadania może przyjmować wartości: *nierozpoczęty*, *w trakcie* lub *ukończony*, co umożliwia precyzyjne monitorowanie postępów i generowanie szczegółowych statystyk.

Encja zapisu na kurs (UserKurs)

Realizacja relacji wiele-do-wielu pomiędzy użytkownikami a kursami. Encja zawiera flagę ukończony, która umożliwia śledzenie ukończenia materiału przez uczestnika. Struktura ta pozwala na elastyczne zarządzanie dostępem do kursów oraz personalizację ścieżki edukacyjnej.

Encja zgłoszenia technicznego (UserComplain)

Mechanizm umożliwiający uczestnikom platformy zgłaszanie problemów i pytań. Encja przechowuje temat zgłoszenia oraz szczegółowy opis ograniczony do 2000 znaków. Pole

datePublish zawiera znacznik czasu utworzenia zgłoszenia, umożliwiając porządkowanie według chronologii.

Encja odpowiedzi na zgłoszenie (Answer)

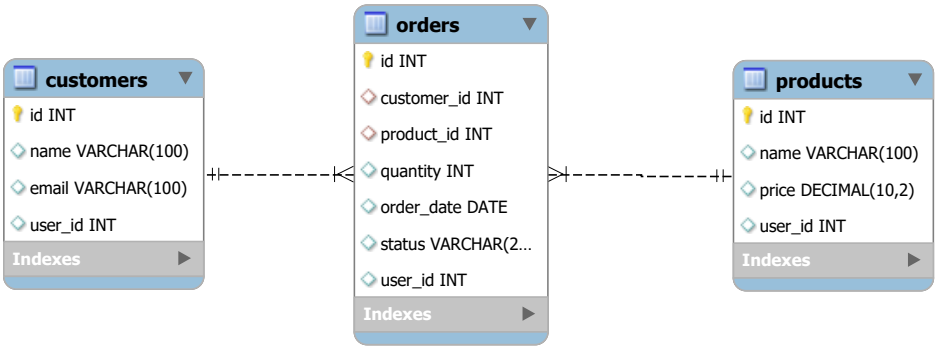
Służy do zapisywania przebiegu dyskusji wokół zgłoszonych tematów. Encja zawiera treść odpowiedzi oraz znacznik czasu jej utworzenia. Struktura ta umożliwia wielowątkową wymianę informacji pomiędzy uczestnikami platformy.

Sekwencja numerowania kursów (kurs_id_seq)

Specjalna struktura bazodanowa służąca do generowania unikalnych identyfikatorów dla nowych kursów edukacyjnych. Sekwencja zapewnia atomowe i niepowtarzalne przydzielanie wartości identyfikatorów, eliminując potencjalne konflikty podczas równoczesnego tworzenia wielu kursów przez różnych administratorów.

5.3.3 Diagram struktury bazy danych środowiska sandbox

Architektura środowiska sandbox została przedstawiona na rysunku 5.2. Baza ta została zaprojektowana jako odseparowane środowisko wykonawcze, w którym uczestnicy platformy mogą bezpiecznie eksperymentować z kodem PHP, współpracującym z bazami danych.



Rysunek 5.2: Diagram struktury bazy danych sandbox

5.3.4 Charakterystyka środowiska sandbox

Środowisko sandbox zaprojektowane z myślą o maksymalnym bezpieczeństwie i izolacji danych pomiędzy różnymi użytkownikami. Jego struktura obejmuje następujące tabele:

Tabela zamówień (orders)

Przykładowa tabela służąca do ćwiczeń związanych z operacjami na danych transakcyjnych. Zawiera informacje o produktach, ilościach, cenach oraz datach realizacji zamówień.

Każdy rekord jest automatycznie przypisywany do konkretnego użytkownika poprzez kolumnę user_id.

Tabela klientów (customers)

Struktura przeznaczona do ćwiczeń z zarządzaniem danymi klientów. Zawiera podstawowe informacje kontaktowe oraz jest izolowana dla określonego użytkownika za pomocą kolumny user_id.

Tabela produktów (products)

Encja umożliwiająca ćwiczenia z zarządzaniem katalogiem produktów. Struktura ta służy do nauki operacji CRUD w kontekście praktycznych scenariuszy biznesowych.

5.3.5 Zasady bezpieczeństwa dostępu

System implementuje rozbudowany model kontroli dostępu do zasobów bazodanowych, różnicujący uprawnienia pomiędzy główną bazą danych a środowiskiem sandbox. Użytkownik bazy sandbox (eduphp_sandbox) posiada ograniczone uprawnienia, obejmujące jedynie operacje SELECT, INSERT, UPDATE i DELETE na tabelach środowiska wykonawczego. Wszystkie uprawnienia administracyjne (DROP, CREATE, ALTER) zostały jawnie odwołane, co eliminuje ryzyko przypadkowej lub celowej modyfikacji struktury bazy.

5.3.6 Strategie izolacji danych pomiędzy użytkownikami

Kluczowym aspektem architektury bazodanowej systemu jest zapewnienie pełnej izolacji danych pomiędzy różnymi uczestnikami platformy. Osiągnięto to poprzez:

1. Fizyczną separację baz: Oddzielenie bazy głównej od środowiska sandbox eliminuje ryzyko przypadkowego uszkodzenia danych systemowych przez kod użytkownika.
2. Automatyczne wstawianie identyfikatorów: System automatycznie dodaje kolumnę user_id do każdego zapytania INSERT wykonywanego w środowisku sandbox, co zapewnia poprawne przypisanie danych do właściwego użytkownika.
3. Dynamiczną modyfikację zapytań: Wszystkie operacje SELECT, UPDATE i DELETE są automatycznie rozszerzane o warunki filtrujące dane według identyfikatora użytkownika razem z domyślnymi danymi bazy, co uniemożliwia dostęp do informacji innych uczestników.
4. Ograniczenia na ilość rekordów: Zapytania SELECT są automatycznie uzupełniane o klauzulę LIMIT 100, co zapobiega przypadkowemu lub celowemu przeciążeniu systemu poprzez odpytywanie dużych zbiorów danych.

5.3.7 Mechanizmy integralności i spójności danych

System implementuje szereg mechanizmów zapewniających poprawność i kompletność przechowywanych informacji:

- Ograniczenia kluczy obcych: Relacje pomiędzy tabelami są egzekwowane na poziomie bazy danych, co zapobiega powstawaniu nieprawidłowych powiązań.
- Unikalne ograniczenia: Kluczowe pola identyfikacyjne (login, sandboxUserId) posiadają unikalne ograniczenia, eliminując możliwość powstania duplikatów.
- Wartości domyślne: Pola takie jak rola czy isActive posiadają sensowne wartości domyślne, zapewniając poprawność danych nawet przy niepełnej inicjalizacji.
- Ograniczenia długości: Określone maksymalne długości pól tekstowych zapobiegają nadmiernemu zużyciu przestrzeni dyskowej i potencjalnym problemom z wydajnością.

5.4 Komponenty, moduły, biblioteki oraz przegląd istotnych klas

Platformę EduPHP zorganizowano jako zbiór współpracujących ze sobą komponentów, z których każdy realizuje określone zadania funkcjonalne. W niniejszej części przedstawiono strukturę projektu, kluczowe moduły systemowe, wykorzystane biblioteki zewnętrzne oraz najważniejsze klasy implementujące podstawową logikę aplikacji.

5.4.1 Struktura organizacyjna projektu

Projekt podzielono na logicznie wydzielone moduły, co ułatwia zarządzanie kodem źródłowym, testowanie poszczególnych komponentów oraz późniejszą rozbudowę systemu. Główna struktura katalogów odzwierciedla architekturę warstwową zastosowaną w implementacji.

Warstwa prezentacji (katalog Controller)

Moduł ten zawiera klasy odpowiedzialne za obsługę żądań HTTP i komunikację z klientem zewnętrznym. Kontrolery implementują punkty końcowe REST API, przetwarzają dane wejściowe i koordynują przepływ informacji pomiędzy warstwą prezentacji a logiką biznesową.

Warstwa logiki biznesowej (katalog Service)

Moduł zawierający implementację zasad działania systemu. Serwisy encapsulują złożoną logikę przetwarzania, zapewniając separację między obsługą żądań HTTP a operacjami na danych. Każdy serwis odpowiada za określony obszar funkcjonalny platformy.

Warstwa dostępu do danych (katalog Repository)

Dany moduł zapewnia abstrakcję dostępu do źródeł danych. Repozytoria definiują interfejsy operacji na bazach danych, wykorzystując mechanizmy dostarczane przez Spring Data JPA do generowania zapytań SQL i zarządzania transakcjami.

Warstwa modelu domenowego (katalog Model)

Moduł zawierający definicje encji odwzorowujących struktury tabel bazodanowych. Klasy modelu implementują mapowanie obiektowo-relacyjne i definiują relacje pomiędzy różnymi bytami systemu.

Warstwa obiektów transferowych (katalog DTO)

Moduł przeznaczony do definiowania struktur danych wykorzystywanych podczas komunikacji pomiędzy warstwami systemu. Obiekty DTO optymalizują transfer informacji poprzez ograniczenie ilości przesyłanych danych.

Warstwa konfiguracyjna (katalog Config)

Moduł odpowiedzialny za definiowanie globalnych ustawień systemowych oraz konfigurację mechanizmów bezpieczeństwa i komunikacji międzykomponentowej. Zawiera klasy konfiguracyjne Springa, które inicjalizują niezbędne komponenty frameworka i określają zasady działania całej aplikacji.

5.4.2 Kluczowe komponenty funkcjonalne

Komponent zarządzania użytkownikami

Moduł odpowiedzialny za rejestrację, uwierzytelnianie i administrację kontami uczestników platformy. Implementuje mechanizmy walidacji danych, hashowania haseł oraz zarządzania sesjami użytkowników.

Komponent zarządzania treścią edukacyjną

Moduł umożliwiający tworzenie, modyfikację i prezentację materiałów dydaktycznych. Zarządza strukturą kursów, zadaniami praktycznymi oraz zasobami multimedialnymi wykorzystywanymi w procesie nauczania.

Komponent środowiska wykonawczego sandbox

Najbardziej zaawansowany technicznie moduł systemu, odpowiedzialny za bezpieczne wykonanie kodu PHP tworzonego przez użytkowników. Implementuje mechanizmy izolacji, monitorowania zasobów oraz automatycznej walidacji rozwiązań.

Komponent śledzenia postępów

Moduł gromadzący i analizujący dane dotyczące aktywności uczestników platformy. Śledzi postępy w realizacji zadań, oblicza statystyki efektywności nauki oraz generuje raporty i wizualizacje wyników.

Komponent komunikacji i zgłoszeń

Moduł umożliwiający wymianę informacji pomiędzy użytkownikami platformy. Implementuje system zgłaszania problemów, dyskusji wokół zgłoszeń oraz moderacji treści.

5.4.3 Wykorzystane biblioteki i zależności zewnętrzne

System został zbudowany z wykorzystaniem sprawdzonych bibliotek open-source, wymienionych w tabeli 5.1, które dostarczają gotowe rozwiązania dla typowych problemów programistycznych, przyspieszając rozwój aplikacji i zwiększając jej niezawodność.

Tabela 5.1: Główne zależności projektu

Biblioteka	**Wersja**	**Przeznaczenie**
Spring Boot Starter Web	3.5.4	Podstawowy framework aplikacji webowych i REST API
Spring Boot Starter Data JPA	3.5.4	Obsługa dostępu do baz danych przez JPA/Hibernate
Spring Boot Starter Security	3.5.4	Mechanizmy bezpieczeństwa, autoryzacji i hashowania BCrypt
Spring Boot DevTools	3.5.4	Narzędzia deweloperskie do hot-reload podczas programowania
MySQL Connector/J	8.0.x	Sterownik łączący z bazą danych MySQL
Lombok	1.18.x	Automatyczne generowanie getterów, setterów i konstruktorów
SpringDoc OpenAPI UI	2.7.0	Automatyczna dokumentacja API (Swagger UI)
Spring Boot Configuration Processor	3.5.4	Przetwarzanie konfiguracji właściwości aplikacji

5.4.4 Przegląd najważniejszych klas systemowych

Klasa PHPExecutorService

Centralny komponent odpowiedzialny za bezpieczne wykonanie kodu PHP w izolowanym środowisku sandbox. Klasa implementuje zaawansowane mechanizmy zabezpieczeń, w tym timeout wykonania, ograniczenia zasobów oraz automatyczną izolację danych w zapytaniach SQL.

- Główne metody:
  - executePHPCode() - wykonuje kod PHP z pełnymi zabezpieczeniami;
  - saveUserSolution() - zapisuje rozwiązanie użytkownika do pliku;
  - getLastUserSolution() - pobiera ostatnie rozwiązanie użytkownika;
  - addPHPSecurityWrappers() - opakowuje kod użytkownika w warstwę bezpieczeństwa.
- Kluczowe funkcjonalności:
  - Tworzenie izolowanych procesów PHP za pomocą ProcessBuilder.
  - Automatyczna modyfikacja zapytań SQL z izolacją per użytkownik.
  - Przechwytywanie i analiza wyników wykonania kodu.
  - Zarządzanie plikami tymczasowymi i ich automatyczne usuwanie.

Klasa SandboxUserService

Komponent odpowiedzialny za generowanie i zarządzanie unikalnymi identyfikatorami użytkowników w środowisku sandbox. Implementuje algorytm gwarantujący unikalność identyfikatorów przy zachowaniu losowości ich wartości.

- Główna metoda: getOrCreateSandboxUserId() - zwraca istniejący lub generuje nowy identyfikator;
- Algorytm generowania: Losowe wartości z zakresu 999-99999 z weryfikacją unikalności;
- Maksymalna liczba prób: 50 prób generowania unikalnego identyfikatora.

Klasa UserService

Kompleksowy serwis zarządzający wszystkimi aspektami związanymi z kontami użytkowników. Implementuje zaawansowane mechanizmy walidacji danych wejściowych z ochroną przed typowymi atakami sieciowymi.

- Kluczowe metody:
  - zapiszRejestracje() - przetwarza proces rejestracji nowego użytkownika;
  - sprawdzLogin() - weryfikuje dane logowania i zarządza sesją;
  - deleteUser() - realizuje transakcyjne usuwanie konta użytkownika.
- Mechanizmy bezpieczeństwa:
  - wielopoziomowa walidacja danych wejściowych;
  - ochrona przed SQL Injection, XSS i buffer overflow;
  - hashowanie haseł z wykorzystaniem algorytmu BCrypt.

Klasa UserTaskService

Serwis śledzący postępy użytkowników w realizacji zadań programistycznych. Implementuje algorytm obliczania punktacji uwzględniający zarówno czas wykonania, jak i liczbę prób.

- Główne metody:
  - calculateScore() - oblicza wynik w skali 0-10 na podstawie czasu i prób
  - startTask() - inicjuje nowe zadanie dla użytkownika;
  - completeTask() - finalizuje zadanie z obliczeniem punktacji;
  - getUserTaskStatusDTO() - zwraca aktualny status zadania.
- Algorytm punktacji: Wagi 0.7 dla czasu i 0.3 dla liczby prób.
- Automatyczne zarządzanie stanem: NOT_STARTED → IN_PROGRESS → COMPLETED.

Klasa KursService

Serwis zarządzający strukturą kursów edukacyjnych i śledzeniem postępów użytkowników w ich realizacji. Implementuje mechanizmy obliczania stanu zaawansowania w kursach oraz ogólnych statystyk postępów.

- Kluczowe metody:
  - getProgressUkonczenia() - oblicza postęp w konkretnym kursie;
  - getOverallProgress() - oblicza ogólny postęp użytkownika;
  - getCompletedCoursesCount() - zwraca liczbę ukończonych kursów;

  - deleteKurs() - realizuje kaskadowe usuwanie kursu z wszystkich powiązanych danych.

- Mechanizmy obliczania:

  - procentowe określenie postępu w kursie;
  - średni postęp uwzględniający wszystkie kursy użytkownika;
  - zliczanie ukończonych kursów z flagą ukończony.

Klasa GlobalExceptionHandler

Centralny komponent obsługi wyjątków w aplikacji. Zapewnia spójne i przyjazne użytkownikowi komunikaty o błędach, jednocześnie logując szczegóły techniczne dla celów diagnostycznych.

- Obsługiwane wyjątki:

  - IllegalArgumentException - błędy walidacji danych wejściowych;
  - DataIntegrityViolationException - naruszenia integralności danych;
  - MethodArgumentNotValidException - błędy walidacji DTO;
  - Exception - ogólne błędy serwera.

- Struktura odpowiedzi: Spójny format JSON ze wskazaniem typu błędu i komunikatem.

- Logowanie: Pełne informacje diagnostyczne zapisywane w logach systemowych.

5.4.5 Architektura komunikacji między komponentami

Komunikacja pomiędzy poszczególnymi komponentami systemu odbywa się zgodnie z ustalonymi wzorcami projektowymi, co zapewnia luźne powiązania i wysoką kohezję poszczególnych modułów.

- Kontrolery - Serwisy: Wywołania metod z przekazywaniem obiektów DTO;
- Serwisy - Repozytoria: Operacje na danych poprzez interfejsy JPA;
- Frontend - Backend: Komunikacja REST API z wykorzystaniem JSON;
- Sandbox - Baza danych: Zabezpieczone połączenia z automatyczną izolacją.

5.4.6 Wzorce projektowe zastosowane w implementacji

System EduPHP wykorzystuje szereg sprawdzonych wzorców projektowych, które zwiększają jego modularność, testowalność i łatwość utrzymania. Poniżej przedstawiono kluczowe wzorce wraz z przykładami ich implementacji w kodzie źródłowym platformy.

Wzorzec Warstw (Layered Architecture)

Separacja odpowiedzialności pomiędzy prezentacją, logiką biznesową i dostępem do danych. Architektura warstwowa jest fundamentem całego systemu, co widać w strukturze pakietów przedstawwionej na listingu 5.1:

Listing 5.1: Struktura warstwowa systemu

```
// Warstwa Prezentacji (Controller)
package com.polsl.EduPHP.Controller;
// np. UserController, KursController, PHPExecutionController

// Warstwa Logiki Biznesowej (Service)
package com.polsl.EduPHP.Service;
// np. UserService, PHPExecutorService, TaskService

// Warstwa Dostępu do Danych (Repository)
package com.polsl.EduPHP.Repository;
// np. UserRepository, TaskRepository, UserTaskRepository

// Warstwa Modelu (Entity)
package com.polsl.EduPHP.model;
// np. User, Task, Kurs, UserTask

// Warstwa DTO (Data Transfer Object)
package com.polsl.EduPHP.DTO;
// np. UserRegisterDTO, TaskDTO, UserTaskDTO
```

Każda warstwa ma jasno określone obowiązki i komunikuje się tylko z sąsiednimi warstwami, co minimalizuje zależności i ułatwia testowanie.

Wzorzec Serwis (Service Pattern)

Encapsulacja złożonej logiki biznesowej w dedykowanych komponentach. Serwisy grupują powiązane funkcjonalności i ukrywają złożoność przed warstwą prezentacjizostało przedstawiono na listingu 5.2

Listing 5.2: Przykład serwisu z logiką biznesową

```
@Service
@Transactional
public class UserService {

    public User zapiszRejestracje(UserRegisterDTO rejestracja) {
        // WALIDACJA BEZPIECZEŃSTWA
        validateUserInput(rejestracja);

        // WALIDACJA BIZNESOWA
        Optional<User> existingUser = userRepository.findByLogin(
            rejestracja.getLogin());
        if (existingUser.isPresent()) {
            throw new IllegalArgumentException(
                "Login ' " + rejestracja.getLogin() + "' jest już
                   zajęty");
        }

        //      MAPOWANIE DTO     ENCJA Z HASHOWANIEM HASŁA
        User user = new User();
        user.setFirstName(rejestracja.getFirstName());
        // ... więcej logiki biznesowej

        return userRepository.save(user);
    }

    // Pozostała logika biznesowa związana z użytkownikami
    public Map<String, Object> sprawdzLogin(UserLogowanieDTO
       logowanie) { ... }
    public void deleteUser(Integer userId) { ... }
}
```

Wzorzec Repozytorium (Repository Pattern)

Abstrakcja dostępu do źródeł danych. Repozytoria zapewniają jednolity interfejs do operacji na danych, niezależny od konkretnej implementacji bazy danych, przykłaad przedstawiono na listingu 5.3

Listing 5.3: Przykłady repozytoriów

```
1 // Interfejs repozytorium dla użytkowników
2 @Repository
3 public interface UserRepository extends CrudRepository<User,
    Integer> {
4     Optional<User> findByLogin(String login);
5
6     @Query("SELECT␣u.sandboxUserId␣FROM␣User␣u␣WHERE␣u.idUser␣=␣:
        userId")
7     Optional<Integer> findSandboxUserIdByUserId(@Param("userId")
        Integer userId);
8
9     Optional<User> findBySandboxUserId(Integer sandboxUserId);
10 }
```

Wzorzec DTO (Data Transfer Object)

Optymalizacja transferu danych pomiędzy warstwami. Obiekty DTO przenoszą tylko niezbędne dane, poprawiając wydajność i bezpieczeństwo, przykład którego jest zaprezentowanych na listingu 5.4

Listing 5.4: Przykłady klas DTO

```
1 // DTO dla rejestracji użytkownika z walidacją
2 @Data
3 public class UserRegisterDTO {
4     @NotBlank(message = "Imię␣jest␣wymagane")
5     @Size(min = 2, max = 50, message = "Imię␣musi␣mieć␣2-50␣znakó
        w")
6     private String firstName;
7
8     @NotBlank(message = "Nazwisko␣jest␣wymagane")
9     @Size(min = 2, max = 50, message = "Nazwisko␣musi␣mieć␣2-50␣
        znaków")
10    private String secondName;
11
12    @NotBlank(message = "Login␣jest␣wymagany")
```

```
13      @Size(min = 3, max = 30, message = "Login musi mieć 3-30 znak
            ów")
14      private String login;
15
16      @NotBlank(message = "Hasło jest wymagane")
17      @Size(min = 6, message = "Hasło musi mieć co najmniej 6 znakó
            w")
18      private String passwd;
19
20      private String rola = "user";
21  }
```

Podsumowanie zastosowanych wzorców

Wykorzystanie powyższych wzorców projektowych w systemie EduPHP przynosi następujące korzyści:

- Modularność: Każdy komponent ma jasno określone obowiązki;
- Testowalność: Izolacja warstw ułatwia pisanie testów jednostkowych;
- Zmienność: pozwala na łatwą wymianę algorytmów;
- Bezpieczeństwo: zapewnia kontrolę nad tworzeniem procesów;
- Wydajność: Wzorzec DTO minimalizuje transfer danych;
- Łatwość utrzymania: Separacja odpowiedzialności ułatwia modyfikacje.

Zastosowanie tych wzorców tworzy solidne fundamenty architektoniczne, które umożliwiają dalszy rozwój platformy przy zachowaniu wysokiej jakości kodu i łatwości jego utrzymania.

5.5 Przegląd ważniejszych algorytmów

W niniejszej części przedstawiono kluczowe algorytmy zaimplementowane w systemie EduPHP. Zaprezentowano zarówno pseudokody ilustrujące zasadę działania poszczególnych mechanizmów, jak również wybrane fragmenty rzeczywistego kodu źródłowego demonstrujące praktyczną implementację omawianych rozwiązań.

5.5.1 Algorytm automatycznej izolacji zapytań SQL

Algorytm 1 demonstruje mechanizm automatycznej modyfikacji zapytań SQL w celu zapewnienia izolacji danych pomiędzy różnymi użytkownikami platformy. Przy projektowaniu struktury zadań i przykładów operacji na bazach danych czerpano z uznanych przez społeczności programistów materiałów edukacyjnych, w szczególności z książki [30], która dostarcza praktycznych przykładów i sprawdzonych metod nauczania programowania w PHP z wykorzystaniem MySQL.

Algorithm 1 Automatyczna izolacja zapytań SQL per użytkownik

```
1: function DODAJIZOLACJĘUŻYTKOWNIKA(zapytanie, userId)
2:     zapytanieWielkie ← naWielkieLitery(przytnij(zapytanie))
3:     if zapytanieWielkie zaczyna się od "SELECT" then
4:         return modyfikujZapytanieSELECT(zapytanie, userId)
5:     else if zapytanieWielkie zaczyna się od "INSERT" then
6:         return modyfikujZapytanieINSERT(zapytanie, userId)
7:     else if zapytanieWielkie zaczyna się od "UPDATE" lub
   zapytanieWielkie zaczyna się od "DELETE" then
8:         return modyfikujZapytanieAktualizujące(zapytanie, userId)
9:     else
10:        return zapytanie                ▷ Zachowaj oryginalne zapytanie
11:    end if
12: end function
```

Przykładowe transformacje zapytań SQL

Implementacja automatycznej izolacji realizuje następujące transformacje, przykładowe transformacje przedstawione są na listingu 5.5:

Listing 5.5: Przykłady transformacji zapytań SQL

```
1  -- Oryginalne zapytanie użytkownika:
2  SELECT * FROM orders WHERE quantity > 10
3
4  -- Po automatycznej transformacji:
5  SELECT * FROM orders
6  WHERE (user_id = 12345 OR user_id IS NULL)
7    AND quantity > 10
8  LIMIT 100
9
10 -- Oryginalne INSERT:
11 INSERT INTO orders (product_name, quantity)
12 VALUES ('Produkt␣A', 5)
13
14 -- Po automatycznej transformacji:
```

```
INSERT INTO orders (product_name, quantity, user_id)
VALUES ('Produkt A', 5, 12345)

-- Oryginalne UPDATE:
UPDATE orders SET quantity = 0 WHERE id = 100

-- Po automatycznej transformacji:
UPDATE orders SET quantity = 0
WHERE id = 100 AND (user_id = 12345 OR user_id IS NULL)
```

5.5.2 Algorytm generowania unikalnego sandboxUserId

Algorytm 2 prezentuje procedurę generowania unikalnych identyfikatorów dla użytkowników w środowisku sandbox.

Algorithm 2 Generowanie unikalnego identyfikatora sandbox

```
function wygenerujUnikalneIdSandbox
    MIN_ID ← 999                                   ▷ Minimalna wartość ID
    MAX_ID ← 99999                                 ▷ Maksymalna wartość ID
    MAKS_PRÓB ← 50                  ▷ Maksymalna liczba prób generowania
    próby ← 0
    while próby < MAKS_PRÓB do
        losoweId ← losowaLiczbaCałkowita(MIN_ID, MAX_ID)
        if ¬czyIdIstniejeWBazie(losoweId) then
            return losoweId                        ▷ Znaleziono unikalne ID
        end if
        próby ← próby + 1
    end while
    throw nowy WyjątekRuntime("Nie udało się wygenerować unikalnego ID")
end function
```

5.5.3 Podsumowanie algorytmów

Przedstawione algorytmy stanowią podstawę działania systemu EduPHP, zapewniając bezpieczeństwo wykonania kodu użytkowników. Zastosowanie zaawansowanych mechanizmów izolacji danych oraz przydzielenia unikalnego identyfikatora dla każdego użytkownika w izolowanej bazie danych czyni platformę odporną na typowe zagrożenia bezpieczeństwa charakterystyczne dla środowisk edukacyjnych.

Rozdział 6

Weryfikacja i walidacja

Niniejszy rozdział prezentuje proces weryfikacji i walidacji, którego celem było potwierdzenie spełnienia zdefiniowanych wymagań funkcjonalnych i niefunkcjonalnych oraz zapewnienie jakości i niezawodności wdrożonego rozwiązania. Opisano przyjętą metodykę testowania, zorganizowane eksperymenty, konkretne przypadki testowe, a także wykryte i usunięte błędy.

6.1 Przyjęta metodyka testowania

Weryfikację platformy przeprowadzono zgodnie ze strukturalnym modelem V, dostosowanym do charakteru aplikacji webowej oraz specyfiki projektu. Model ten zapewnił systematyczne powiązanie czynności testowych z odpowiednimi etapami rozwoju systemu, gwarantując kompleksową weryfikację na każdym poziomie abstrakcji.

6.1.1 Model V w kontekście projektu

Proces weryfikacji odzwierciedlał lewą i prawą gałąź modelu V [17], koncentrując się na praktycznej walidacji kolejnych komponentów systemu.

- Testowanie wymagań (Requirements Testing): Każdą funkcjonalność opisaną w rozdziale 3 zweryfikowano pod kątem kompletnej i poprawnej implementacji. Przykładem jest walidacja mechanizmu sandbox pod kątem spełnienia wszystkich założeń bezpieczeństwa.
- Testowanie architektury i projektowania (Architecture & Design Testing): Przeprowadzono testy integracyjne sprawdzające poprawność komunikacji między kluczowymi komponentami opisanymi w rozdziale 5.2, takimi jak frontend z backendem REST API oraz backend z bazami danych, czyli główną i sandbox.
- Testowanie systemowe (System Testing): Platformę poddano kompleksowym testom jako całość, z perspektywy użytkownika końcowego. Sprawdzono wszystkie ścieżki

użycia, w tym rejestrację, logowanie, zarządzanie kursami, rozwiązywanie zadań, działanie systemu zgłoszeń oraz zarządzanie użytkownikami.

- Testowanie akceptacyjne (Acceptance Testing): Finalna walidacja potwierdziła, że system realizuje swój podstawowy cel – umożliwia bezpieczną i efektywną naukę programowania w PHP poprzez interaktywną platformę edukacyjną.

6.1.2 Organizacja procesu testowego i podejście BDD

Ze względu na charakter projektu, w którym nacisk położono na implementację kluczowych mechanizmów bezpieczeństwa i funkcjonalności, zrezygnowano z pisania automatycznych testów jednostkowych na rzecz testów manualnych i integracyjnych, prowadzonych w Behavior-Driven Development [17], rozwój oparty na zachowaniu. Podejście to skoncentrowało się na weryfikacji zachowań systemu zdefiniowanych w postaci konkretnych scenariuszy użytkownika.

Proces testowy obejmował trzy etapy:

1. Odkrywanie (ang. discovery): Analiza wymagań funkcjonalnych z rozdziału 3 w celu zidentyfikowania kluczowych scenariuszy użycia i krytycznych ścieżek w systemie.

2. Formułowanie (ang. formulation): Przekształcenie wymagań w zestaw konkretnych, wykonywalnych scenariuszy testowych, opisujących akcje użytkownika i oczekiwane reakcje systemu. Scenariusze te stały się podstawą dla przypadków testowych opisanych w sekcji 6.2.

3. Ręczna egzekucja (ang. manual execution): Systematyczne przeprowadzanie zdefiniowanych scenariuszy w rzeczywistym środowisku aplikacji z wykorzystaniem narzędzi wspomagających opisanych w sekcji 6.4.

Takie podejście pozwoliło na efektywną weryfikację złożonych interakcji, szczególnie w modułach takich jak bezpieczeństwo w bazie danych sandbox, gdzie automatyzacja na wczesnym etapie byłaby utrudniona.

6.2 Przypadki testowe

W ramach procesu weryfikacji zdefiniowano i wykonano szereg przypadków testowych, skupiających się na krytycznych ścieżkach funkcjonalnych oraz wymaganiach niefunkcjonalnych, w szczególności dotyczących bezpieczeństwa. Testy miały charakter niepełny wobec wszystkich możliwych kombinacji danych, lecz były kompletne w zakresie weryfikacji założonych wymagań.

6.2.1 Testy funkcjonalne – scenariusze użytkownika

Główne testy funkcjonalne koncentrowały się na podstawowych operacjach dostępnych dla dwóch ról użytkowników: administratora (`ADMIN`) i zwykłego użytkownika (`USER`). Wyniki testów zestawiono w tabeli 6.1.

Tabela 6.1: Przypadki testowe funkcjonalne systemu EduPHP

ID	Scenariusz	Warunki wstępne	Oczekiwany rezultat	Status
T1	Rejestracja nowego użytkownika	System uruchomiony, baza danych gotowa.	Konto zostaje utworzone, użytkownik jest przekierowany do formularza logowania, a po uwierzytelnieniu na dashboard.	SUKCES
T2	Utworzenie kursu przez administratora	Użytkownik zalogowany z uprawnieniami `ADMIN`.	Nowy kurs jest zapisywany w bazie danych i pojawia się na liście kursów po odświeżeniu strony.	SUKCES
T3	Utworzenie zadania programistycznego	Użytkownik `ADMIN`, istnieje przynajmniej jeden kurs.	Zadanie zostaje przypisane do kursu i jest widoczne na liście zadań.	SUKCES
T4	Zgłoszenie problemu przez użytkownika	Dowolny zalogowany użytkownik (`USER` lub `ADMIN`).	Zgłoszenie trafia do ogólnej listy i jest widoczne w panelu zgłoszeń.	SUKCES
T5	Blokowanie użytkownika przez administratora	Użytkownik `ADMIN`, istnieje konto użytkownika do zablokowania.	Zablokowany użytkownik nie może się zalogować; otrzymuje komunikat o konieczności kontaktu z administratorem.	SUKCES
T6	Wykonanie kodu PHP w sandbox	Zalogowany użytkownik, aktywne zadanie.	Kod wykonuje się, a wynik (lub błąd) jest prezentowany w konsoli. Próba użycia niebezpiecznych funkcji kończy się błędem.	SUKCES

Przeprowadzenie powyższych testów potwierdziło, że wszystkie kluczowe moduły systemu (zarządzanie użytkownikami, treścią, sandbox, zgłoszeniami) działają zgodnie z za-

łożeniami i integrują się w spójną całość.

6.2.2 Testy niefunkcjonalne – bezpieczeństwo i wydajność

W danym podpunkcie przeprowadzono szereg eksperymentów mających na celu weryfikację wymagań niefunkcjonalnych.

Testy bezpieczeństwa sandbox

Środowisko wykonawcze PHP, opisane w sekcji 5.5, zostało poddane intensywnym testom mającym na celu potwierdzenie skuteczności implementowanych zabezpieczeń.

- Ochrona przed Denial-of-Service (DoS): Wysłano kod zawierający nieskończoną pętlę (`while(true){}`). Oczekiwano, że proces zostanie przerwany po 10 sekundach zdefiniowanych w parametrze `php.execution.timeout`. Test zakończył się powodzeniem – wykonywanie kodu zostało automatycznie zatrzymane.
- Izolacja danych użytkowników: Dla dwóch różnych kont użytkowników wykonano te same operacje SQL na tabelach w bazie sandbox. Potwierdzono, że zapytania `SELECT` zwracają tylko dane przypisane do danego użytkownika, dzięki automatycznemu dodaniu warunku `WHERE user_id = X`, a operacje `INSERT` automatycznie wstawiają właściwą wartość `user_id`.
- Blokada niebezpiecznych funkcji: Próby wywołania funkcji systemowych takich jak `system()`, `exec()`, `eval()` zakończyły się błędem wykonania, potwierdzając działanie parametru `disable_functions` w wrapperze bezpieczeństwa.

Testy wydajnościowe

W środowisku deweloperskim przeprowadzono podstawową walidację responsywności systemu.

- Czas ładowania stron: Kluczowe strony interfejsu użytkownika ładowały się w czasie poniżej 3 sekund.
- Czas wykonania kodu: Proste skrypty PHP wykonywały się natychmiast, a bardziej złożone, lecz poprawnie napisane, mieściły się w granicach ustalonego limitu 10 sekund.

6.3 Wykryte i usunięte błędy

Proces rozwoju i testowania platformy EduPHP wiązał się z identyfikacją i usuwaniem licznych usterek. Poniżej opisano najważniejsze błędy, które zostały wykryte i naprawione, co znacząco podniosło stabilność i bezpieczeństwo systemu.

6.3.1 Krytyczne problemy bezpieczeństwa i funkcjonalności

W tabeli 6.2 zestawiono kluczowe błędy wraz z analizą przyczyn i podjętymi działaniami naprawczymi.

Tabela 6.2: Wykryte i usunięte błędy krytyczne

Problem	**Prawdopodobna przyczyna**	**Wdrożone rozwiązanie**	**Efekt**
Nieskończona pętla w kodzie PHP powodowała permanentne zawieszenie wątku wykonawczego.	Brak mechanizmu kontroli czasu wykonania zewnętrznego procesu PHP.	Implementacja timeout'u w klasie `PHPExecutorService` przy użyciu `Process.waitFor()` oraz wymuszonego zniszczenia procesu po przekroczeniu limitu (10s).	Kod jest automatycznie przerywany, co chroni przed atakami DoS.
Potencjalny atak DoS poprzez masowe/złożone zapytania SQL z poziomu sandbox.	Brak ograniczeń na poziomie wrappera dla zapytań typu `SELECT`.	Rozszerzenie automatycznej transformacji zapytań (Algorytm 1) o dodanie klauzuli `LIMIT 100` do każdego zapytania `SELECT`.	Ograniczenie możliwości przeciążenia bazy danych przez pojedyncze zapytanie.
Możliwość wykonania niebezpiecznych poleceń systemowych z poziomu kodu użytkownika.	Niepełna konfiguracja środowiska PHP uruchamianego przez `ProcessBuilder`.	Rozszerzenie listy `disable_functions` w wrapperze bezpieczeństwa o `system, exec, passthru, shell_exec, popen, proc_open, eval`.	Zablokowanie bezpośredniego dostępu do powłoki systemowej.
Brak izolacji danych między użytkownikami w bazie sandbox podczas ręcznego testowania.	Zapytania SQL generowane przez użytkowników nie były automatycznie filtrowane.	Wdrożenie pełnego mechanizmu automatycznej modyfikacji zapytań (dla `SELECT, INSERT, UPDATE, DELETE`) z użyciem losowego `sandboxUserId` oraz domyślnych danych, jak opisano w sekcji 5.3.	Każdy użytkownik operuje wyłącznie na swoim, wydzielonym podzbiorze danych testowych.

6.3.2 Szczegółowa analiza wybranego błędu – izolacja danych

Najbardziej złożonym i interesującym problemem, którego rozwiązanie miało fundamentalne znaczenie dla bezpieczeństwa platformy, był początkowy brak pełnej izolacji danych w środowisku sandbox.

Opis problemu

Podczas równoległego testowania kont dwóch użytkowników zaobserwowano, że wykonując zapytanie `SELECT * FROM orders`, każdy z nich mógł zobaczyć dane wprowadzone przez drugiego. Podobnie, nieodpowiednio skonstruowane zapytanie `UPDATE` lub `DELETE` mogłoby wpłynąć na rekordy innego użytkownika. Było to sprzeczne z podstawowym założeniem bezpiecznego środowiska edukacyjnego.

Analiza przyczyn

Przyczyna leżała w bezpośrednim przekazywaniu kodu użytkownika do interpretera PHP bez żadnej ingerencji w zapytania SQL. Mechanizm izolacji wymagał nie tylko osobnych połączeń do bazy, ale przede wszystkim automatycznej i niewidocznej dla użytkownika modyfikacji każdego wysyłanego zapytania.

Proces debugowania i implementacji rozwiązania

Rozwiązanie wdrożono w postaci bezpiecznego wrappera PHP, który jest dołączany do kodu użytkownika przed wykonaniem. Kluczowe kroki to:

1. Wygenerowanie i przypisanie każdemu użytkownikowi unikalnego, losowego identyfikatora (patrz Algorytm 2).

2. Napisanie funkcji PHP textttmodifyQuery(), która przechwytuje wywołania do bazy danych za pomocą `mysqli::query()`.

3. Implementacja logiki (Algorytm 1), która dynamicznie analizuje i przekształca zapytania:

   - Do `SELECT` dodaje warunek `WHERE user_id = ? OR user_id IS NULL` oraz `LIMIT 100`.
   - Do `INSERT` automatycznie dodaje kolumnę `user_id` z odpowiednią wartością.
   - Do `UPDATE` i `DELETE` dodaje zabezpieczający warunek `WHERE user_id = ?`.

Weryfikacja

Po implementacji przeprowadzono test z dwoma użytkownikami, którzy tworzyli i odczytywali dane w tych samych tabelach. Potwierdzono, że dane każdego z nich są całkowicie odseparowane, a próba wykonania nieautoryzowanego zapytania kończy się brakiem wyników lub błędem, jednocześnie zachowując możliwość pracy na domyślnych danych szkoleniowych, gdzie wartości dla identyfikatorów użytkowników są równe *null*, user_id IS NULL.

6.4 Narzędzia wspomagające proces testowania

W trakcie weryfikacji wykorzystano szereg narzędzi deweloperskich i testujących , które umożliwiły efektywną kontrolę poszczególnych warstw aplikacji, narzędzia te opisano krótko poniżej.

- **Postman**: Podstawowe narzędzie do testowania punktów końcowych REST API backend Spring Boot. Umożliwiło wysyłanie żądań HTTP (GET, POST, PUT, DELETE) z różnymi parametrami i ciałami w formacie JSON, weryfikację kodów odpowiedzi oraz struktury zwracanych danych. Przy jego pomocy utworzono także pierwsze konto administratora.

- **Narzędzia deweloperskie przeglądarki (Opera DevTools)**: Niezbędne do debugowania strony frontend. Użyto konsoli JavaScript do śledzenia błędów skryptów, zakładki `Network` do monitorowania żądań HTTP i odpowiedzi ze strony backend oraz inspektora do weryfikacji poprawności renderowania interfejsu.

- **MySQL Workbench**: Użyte do bezpośredniego sprawdzania stanu baz danych, głównej `eduphp` i `eduphp_sandbox`, po wykonaniu akcji przez aplikację. Pozwalało to na potwierdzenie, czy dane są poprawnie zapisywane, modyfikowane i izolowane.

- **IntelliJ IDEA Debugger**: Zintegrowane środowisko programistyczne służyło do debugowania kodu Java warstwy backend, śledzenia przepływu wykonywania oraz inspekcji zmiennych w kluczowych serwisach, takich jak `PHPExecutorService` czy `UserService`.

6.5 Wyniki walidacji systemu

Końcowym etapem procesu weryfikacji była całościowa walidacja systemu pod kątem spełnienia wymagań określonych na początku projektu.

6.5.1 Spełnienie wymagań funkcjonalnych

Platforma EduPHP w pełni realizuje wszystkie kluczowe wymagania funkcjonalne zdefiniowane w rozdziale 3:

- Zarządzanie treścią edukacyjną: moduł kursów i zadań pozwala administratorowi na tworzenie, edycję i usuwanie materiałów. Użytkownicy mogą przeglądać kursy, zapisywać się na nie i śledzić postępy.
- Interaktywny system zadań: zaimplementowane środowisko sandbox umożliwia pisanie, wykonywanie i testowanie kodu PHP w bezpieczny sposób, z natychmiastową informacją zwrotną. Algorytm oceny, patrz punkt 5.5, śledzi postępy.
- System uwierzytelniania i autoryzacji: Działa poprawnie, rozróżniając role `USER` i `ADMIN`. Hasła są bezpiecznie hashowane przy użyciu BCrypt.
- System zgłoszeń i komunikacji: Umożliwia użytkownikom tworzenie zgłoszeń, a administratorom – zarządzanie nimi i odpowiadanie.
- Panel administracyjny: Dostępny tylko dla roli `ADMIN`, zawiera funkcje zarządzania użytkownikami, kursami i zadaniami.

6.5.2 Spełnienie wymagań niefunkcjonalnych

System spełnia również założenia niefunkcjonalne:

- Wydajność: Interfejs użytkownika ładuje się w akceptowalnym czasie, a wykonanie kodu w sandbox jest ograniczone w czasie, co zapobiega przeciążeniu.
- Bezpieczeństwo: Wdrożono wielowarstwowe zabezpieczenia: hashowanie haseł, izolację procesów PHP, timeout wykonania, blokadę niebezpiecznych funkcji oraz automatyczną izolację danych SQL. System jest odporny na podstawowe ataki, takie jak DoS poprzez nieskończone pętle czy próby wykonania poleceń systemowych.
- Użyteczność: Interfejs jest responsywny, intuicyjny i dostępny w języku polskim.
- Architektura: Zachowano czystą separację frontend (JavaScript), backend (Spring Boot REST API) oraz warstwy danych (MySQL), co ułatwia utrzymanie i rozwój.

6.5.3 Ograniczenia systemu

W trakcie walidacji zidentyfikowano także świadomie zaakceptowane ograniczenia, które wyznaczają kierunki potencjalnej przyszłej rozbudowy:

- Ograniczona automatyzacja testów: Ze względu na priorytet implementacji kluczowych mechanizmów bezpieczeństwa i funkcjonalności sandbox, proces testowania skupił się na manualnych scenariuszach BDD. W pełni produkcyjnym wdrożeniu system powinien zostać uzupełniony o kompleksowe testy automatyczne dla backend, takich jak JUnit, Spring Boot Test oraz frontend.
- Ograniczona skalowalność w kontekście sandbox: Obecne rozwiązanie, oparte o ProcessBuilder, jest wystarczające dla celów edukacyjnych, ale przy bardzo dużej liczbie równoczesnych użytkowników może wymagać optymalizacji (np. kolejkowanie zadań, konteneryzacja).
- Konfiguracja ręczna: Proces instalacji i konfiguracji opisany w rozdziale 4 wymaga ręcznej interwencji (stworzenie admina poprzez Postman, konfiguracja aplikacji). W przyszłości można go zautomatyzować za pomocą narzędzi takich jak Docker.

6.6 Podsumowanie procesu weryfikacji i walidacji

Przeprowadzony proces weryfikacji i walidacji systemu EduPHP, oparty na modelu V i podejściu BDD, pozwolił na potwierdzenie osiągnięcia założonych celów projektowych. Dzięki systematycznym testom manualnym i integracyjnym potwierdzono, że platforma:

1. funkcjonalnie realizuje swój główny cel – służy jako efektywne i bezpieczne narzędzie do nauki programowania w PHP poprzez interaktywne zadania;
2. jest stabilna i niezawodna – kluczowe ścieżki użytkownika działają poprawnie, a system odpowiednio reaguje na poprawne i niepoprawne dane wejściowe;
3. zapewnia bezpieczeństwo – wdrożone mechanizmy sandbox skutecznie izolują kod użytkownika, ograniczają zasoby i chronią integralność danych, co zostało udowodnione poprzez ponowne testy wcześniej wykrytych i naprawionych błędów;
4. spełnia zdefiniowane wymagania – zarówno funkcjonalne, jak i niefunkcjonalne, co dokumentują wyniki przeprowadzonych przypadków testowych.

Rozdział 7

Podsumowanie i wnioski

Celem referowanego projektu inżynierskiego było zaprojektowanie i zaimplementowanie platformy edukacyjnej, umożliwiającej realizację praktycznych ćwiczeń dotyczących programowania w języku PHP, z wykorzystaniem dostępu do relacyjnej bazy danych, przy jednoczesnej minimalizacji nakładu pracy związanego z konfiguracją środowiska programistycznego. Opracowana aplikacja spełnia wszystkie przyjęte założenia i wymagania. W ramach pracy wdrożono mechanizm bezpiecznej rejestracji i uwierzytelniania z wykorzystaniem algorytmu szyfrowania haseł BCrypt. Interfejs użytkownika został zaprojektowany jako przejrzysty i responsywny (dostosowujący się do różnych rozmiarów ekranów). Centralnym i najbardziej wymagającym technicznie modułem systemu jest odizolowane środowisko wykonawcze typu sandbox, pozwalające na bezpieczne uruchamianie skryptów napisanych przez uczestników kursów dostępnych na platformie. Kluczowym wyzwaniem w jego implementacji było opracowanie mechanizmu automatycznej separacji danych poszczególnych użytkowników w bazie danych. Problem ten został rozwiązany poprzez wdrożenie systemu dynamicznego przekształcania zapytań SQL, który transparentnie dodaje filtrujący warunek identyfikatora użytkownika do wszystkich operacji na danych. Dodatkowo mechanizm sandbox blokuje potencjalnie niebezpieczne operacje. W ramach pracy zaimplementowano także pozostałe komponenty, takie jak forum zgłoszeń, moduł statystyk z graficzną wizualizacją danych oraz algorytm przyznawania ocen uwzględniający czas wykonania zadania.

Niezależnie od spełnienia założeń i wymagań, opracowana platforma może być dalej rozbudowywana i doskonalona. Jednym z możliwych kierunków rozwoju aplikacji jest implementacja systemu automatycznej weryfikacji rozwiązań, który samodzielnie porównywałby rezultat działania kodu użytkownika z wzorcową odpowiedzią zdefiniowaną przez autora zadania. Z technicznego punktu widzenia znaczącym usprawnieniem byłaby pełna konteneryzacja wszystkich komponentów aplikacji za pomocą oprogramowania Docker, wraz z automatyczną instalacją i konfiguracją uruchamianą jedną komendą. Kolejnym potencjalnym kierunkiem rozwoju aplikacji byłoby hostowanie platformy na zewnętrznym serwerze, co umożliwiłoby dostęp do niej szerszej grupie użytkowników poza środowiskiem

lokalnym. Wymagałoby to także dopracowania obsługi zapytań SQL w mechanizmie sandbox, ponieważ obecna implementacja nie interpretuje poprawnie wszystkich konstrukcji języka SQL, np. aliasów, które stanowią jego podstawowe elementy.

Realizacja projektu wiązała się z koniecznością rozwiązania szeregu problemów technicznych. Najbardziej złożonym zadaniem okazało się zapewnienie kompletnej separacji informacji przechowywanych przez różnych użytkowników w bazie danych sandboxa. Opracowanie mechanizmu dynamicznego przekształcania zapytań SQL, który w sposób transparentny dodaje filtrujący warunek dotyczący identyfikatora użytkownika, wymagało wielu iteracji testów i poprawek. Równie istotnym problemem było wdrożenie niezawodnego limitu czasu wykonania skryptu PHP oraz stworzenie skutecznej warstwy zabezpieczającej, która uniemożliwia wywołanie funkcji zagrażających stabilności systemu.

Architektura systemu, oparta na oddzielonych warstwach i komunikacji przez API REST, została zaprojektowana z myślą o elastyczności i łatwości przyszłych modyfikacji. Dzięki temu platforma EduPHP stanowi nie tylko funkcjonalne narzędzie edukacyjne, ale także solidną podstawę do dalszych prac rozwojowych i adaptacji do nowych wymagań.

Bibliografia

[1] David Sklar Adam Trachtenberg. *PHP: receptury*. O'Really, 2012. ISBN: 978-0-576-10101-5.

[2] Lynn Beighley. *Head First SQL*. O'Reilly Media, 2015. ISBN: 978-05-968-0085-7.

[3] Marjorie Sayer-Anshu Aggarwal Sailu Reddy David Gourley Brian Totty. *HTTP The Definitive Guide*. O'Reilly Media, 2009. ISBN: 978-14-493-7958-2.

[4] DivNotes. *Team, C. (no date) The Complete History of Computer Programming Languages.* 2025. URL: `https://divnotes.com/blog/history-of-computer-programming-languages` (term. wiz. 12. 01. 2026).

[5] tłumaczenie: Marek Pêtlicki Faroult Robson. *SQL: sztuka programowania*. Helion, 2012. ISBN: 978-83-246-0895-9.

[6] Peter Gasston. *The Book of CSS3*. No Starch Press, 2011. ISBN: 978-15-932-7286-9.

[7] Mark Heckler. *Spring Boot: Up and Running*. O'Reilly Media, 2021. ISBN: 978-14-920-7693-3.

[8] Anil Hemrajani. *Agile Java Development with Spring, Hibernate and Eclipse*. Pearson Education, 2006. ISBN: 978-01-327-1490-7.

[9] Hoffman. *Web Application Security*. O'Reilly Media, Incorporated, 2024. ISBN: 978-10-981-4393-0.

[10] Agile Institute. *Iteracyjne i przyrostowe tworzenie oprogramowania*. 2022. URL: `https://agileinstitute.pl/index.php/2022/06/04/iteracyjne-i-przyrostowe-tworzenie-oprogramowania/` (term. wiz. 15. 12. 2025).

[11] Derek J. Balling Jeremy D. Zawodny. *High Performance MySQL Optimization, Backups, Replication, Load Balancing More*. O'Reilly Media, 2008. ISBN: 978-05-965-5516-0.

[12] Ian Robinson Jim Webber Savas Parastatidis. *REST in Practice. Hypermedia and Systems Architecture*. O'Reilly, 2010. ISBN: 978-14-493-9692-3.

[13] Mark S. Merkow Joseph T. Sinclair. *Thin Clients Clearly Explained*. Elsevier Science, 2000, ISBN = 978-01-264-5535-9.

[14] Mike Cannon-Brookes Patrick A. Lightbody Joseph Walnes Ara Abrahamian. *Java Open Source Programming*. Wiley, 2004. ISBN: 978-07-645-5834-4.

[15] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 2004. ISBN: 978-03-211-9770-2.

[16] Weidig Lachowski. *Java: podejscie funkcyjne: rozszerzanie obiektowego kodu Javy o zasady programowania funkcyjnego*. Gliwice: Helion, 2024. ISBN: 978-83-289-0651-8.

[17] Marcin Leśniczek. *Behavior-Driven Development – złote, ale czy skromne?* 2021. URL: `https://angular.love/pl/behavior-driven-development-zlote-ale-czy-skromne` (term. wiz. 18. 12. 2025).

[18] Loukman. *SQL Injection*. kologo loukman, 2017. ISBN: 978-39-614-2498-6.

[19] Marcin Dulak Lukasz Dynowski. *Learning API Styles*. O'Reilly Media, 2025. ISBN: 978-10-981-5395-3.

[20] Adam McDaniel. *HTML5*. Wiley, 2011. ISBN: 978-04-709-5222-1.

[21] Michael McLaughlin. *MySQL Workbench: Data Modeling Development*. McGraw Hill LLC, 2013. ISBN: 978-00-717-918-9.

[22] Somnath Musib. *Spring Boot in Practice*. Manning, 2022. ISBN: 978-16-172-9881-3.

[23] Jason T. Roff. *UML: A Beginner's Guide*. McGraw-Hill Education, 2003. ISBN: 978-00-722-2460-3.

[24] Rustcode. *Rustcode (no date) The evolution of Programming Languages: Timeline of innovation from 1940s to today.* 2025. URL: `https://www.rustcodeweb.com/2025/02/evolution-of-programming-languages-timeline.html` (term. wiz. 12. 01. 2026).

[25] ScienceNewsToday. *E. of (2025) The evolution of programming languages, Science News Today.* 2025. URL: `https://www.sciencenewstoday.org/the-evolution-of-programming-languages` (term. wiz. 12. 01. 2026).

[26] Robert Hansen-Anton Rager Petko D. Petkov Seth Fogie Jeremiah Grossman. *XSS Attacks*. O'Reilly, 2011. ISBN: 978-00-805-5340-5.

[27] Kyle Simpson. *You Don't Know JS: ES6 Beyond*. O'Reilly Media, 2015. ISBN: 978-14-919-0526-5.

[28] Laurentiu Spilca. *Spring Security in Action*. Manning, 2020. ISBN: 978-16-383-5074-3.

[29] Catalin Tudose. *Java Persistence with Spring Data and Hibernate*. Manning, 2023. ISBN: 978-16-383-5185-6.

[30] Kevin Yank. *PHP MySQL: novice to ninja*. Melbourne, Australia: SitePoint, 2012. ISBN: 978-0-9872478-1-0.

Dodatki

Spis skrótów i symboli

ACID zestaw właściwości transakcji bazodanowych: Atomowość (Atomicity), Spójność (Consistency), Izolacja (Isolation), Trwałość (Durability)

API Interfejs Programowania Aplikacji (ang. *Application Programming Interface*)

BCrypt algorytm kryptograficzny służący do bezpiecznego hashowania haseł

BDD Rozwój Sterowany Zachowaniami (ang. *Behavior-Driven Development*)

CORS Współdzielenie Zasobów Między Źródłami (ang. *Cross-Origin Resource Sharing*)

CPU Centralna Jednostka Obliczeniowa (ang. *Central Processing Unit*)

CRUD podstawowe operacje na danych: Tworzenie, Odczytywanie, Aktualizacja, Usuwanie (ang. *Create, Read, Update, Delete*)

CSS3 Kaskadowe Arkusze Stylów, wersja 3 (ang. *Cascading Style Sheets*)

DML Język Manipulacji Danymi (ang. *Data Manipulation Language*)

DTO Obiekt Transferu Danych (ang. *Data Transfer Object*)

DoS Odmowa Usługi (ang. *Denial of Service*)

ES6+ standard języka JavaScript (ECMAScript), wersja 6 i nowsze

HTML5 Hipertekstowy Język Znaczników, wersja 5 (ang. *HyperText Markup Language*)

HTTP Protokół Transferu Hipertekstu (ang. *Hypertext Transfer Protocol*)

IDE Zintegrowane Środowisko Programistyczne (ang. *Integrated Development Environment*)

IoC Odwrócenie Kontroli (ang. *Inversion of Control*)

JDK Zestaw Narzędzi Deweloperskich Java (ang. *Java Development Kit*)

JPA Architektura Trwałości Java (ang. *Java Persistence API*)

JWT Token Webowy JSON (ang. *JSON Web Token*)

KiB kibibajt (jednostka pamięci, 1024 bajty)

MB megabajt (jednostka pamięci)

MVC Model-Widok-Kontroler (ang. *Model–View–Controller*)

ORM Mapowanie Obiektowo-Relacyjne (ang. *Object-Relational Mapping*)

PHP pierwotnie: Personal Home Page, obecnie: rekurencyjny akronim PHP: Hypertext Preprocessor (ang. *PHP: Hypertext Preprocessor*)

RAM Pamięć O Dostępie Swobodnym (ang. *Random Access Memory*)

REST Reprezentacyjne Przekazywanie Stanu (ang. *Representational State Transfer*)

RWD Responsywne Projektowanie Stron Internetowych (ang. *Responsive Web Design*)

RUP Ujednolicony Proces Racjonalny (ang. *Rational Unified Process*)

SQL Strukturalny Język Zapytań (ang. *Structured Query Language*)

UI Interfejs Użytkownika (ang. *User Interface*)

UML Ujednolicony Język Modelowania (ang. *Unified Modeling Language*)

URI Jednolity Identyfikator Zasobu (ang. *Uniform Resource Identifier*)

URL Jednolity Lokator Zasobu (ang. *Uniform Resource Locator*)

XSS Skrypty Krzyżowe (ang. *Cross-Site Scripting*)

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- **Źródła programu:**
  - Pliki frontend: `https://github.com/anastasiiapashko/EduPHP/tree/main/EduPHP_Frontend`
  - Pliki backend: `https://github.com/anastasiiapashko/EduPHP/tree/main/EduPHP_Backend`
  - Interpreter PHP: `https://github.com/anastasiiapashko/EduPHP/tree/main/php`
- **Dane testowe:**
  - Pliki skryptowe: `https://github.com/anastasiiapashko/EduPHP/tree/main/skrypty`
- **Film pokazujący działanie opracowanego oprogramowania:** `https://drive.google.com/file/d/1GHfnn0DDt6nFB4QryDZxfEtADdhh62kU/view?usp=sharing`

Spis rysunków

Spis tabel