Studia i Materiały Informatyki Stosowanej, Tom 18, Nr 1, 2026
str. 45-49

Aplikacja webowa wspomagająca pracę uczniów z korepetytorami

Mateusz Skajewski, Zbyszko Królikowski

Wydział Informatyki
Uniwersytet Kazimierza Wielkiego, ul. Mikołaja Kopernika 1, 85-074 Bydgoszcz
e-mail: mateusz.skajewski@student.ukw.edu.pl

Streszczenie: *W artykule przedstawiono projekt i implementację aplikacji webowej, której głównymi założeniami jest wspomaganie współpracy w relacji uczeń-korepetytor poprzez zautomatyzowanie procesu umawiania się na spotkania oraz znajdowanie uczniów bądź nauczycieli celem podjęcia współpracy.*

Słowa kluczowe: *uczeń, nauczyciel, oferty, dyspozycje, spotkania, kontakt, CodeIgniter, PHP*

Web Application Supporting Student-Tutor Collaboration

Abstract: *The paper describes the design and implementation of a web application aimed at supporting student-tutor collaboration by automating meeting scheduling and finding students or teachers for cooperation.*

Keywords: *student, teacher, offers, dispositions, meetings, contact, CodeIgniter, PHP*

1. Wstęp

Celem pracy było zaprojektowanie i stworzenie aplikacji webowej umożliwiającej umieszczanie i korzystanie z ogłoszeń korepetycji w relacji nauczyciel-uczeń. Korepetytorzy mogą zamieszczać swoje ogłoszenia, a uczniowie z nich korzystać. W realizacji projektu wykorzystano system do zarządzania bazą danych MySQL, połączenie za pomocą skryptów PHP. Projekt został stworzony w oparciu o framework CodeIgniter. Narzędziem, które wykorzystano do obsługi bazy danych w projekcie jest phpMyAdmin, a użyte języki to HTML, CSS, PHP oraz JS. Aplikacja webowa została tak zaprojektowana, że dodanie nowych funkcji oraz edytowanie istniejących jest intuicyjne i fragmentaryczne, nie wpływa na całość programu ani na inne, niezwiązane funkcjonalności.

2. Projekt aplikacji

2.1. Środowisko i narzędzia

W projekcie wykorzystano framework **CodeIgniter 4** [1], będący darmowym narzędziem do tworzenia dynamicznych aplikacji webowych w języku PHP. Framework opiera się na architekturze MVC (Model–View–Controller), co sprzyja przejrzystości i spójności kodu. Dostarcza on liczne biblioteki i wbudowane mechanizmy, m.in. ochronę przed atakami CSRF, walidację formularzy oraz obsługę różnych systemów baz danych, takich jak MySQL, PostgreSQL czy SQLite. Istotnym wsparciem w procesie tworzenia aplikacji był wbudowany pasek narzędzi do debugowania.

Środowiskiem programistycznym użytym podczas realizacji projektu był **Visual Studio Code** — edytor kodu źródłowego stworzony przez firmę Microsoft. Narzędzie to oferuje funkcje ułatwiające pracę programisty, takie jak podświetlanie składni, parowanie nawiasów czy możliwość instalacji rozszerzeń. Dzięki swojej elastyczności i popularności stanowi ono wygodne środowisko do tworzenia aplikacji webowych.

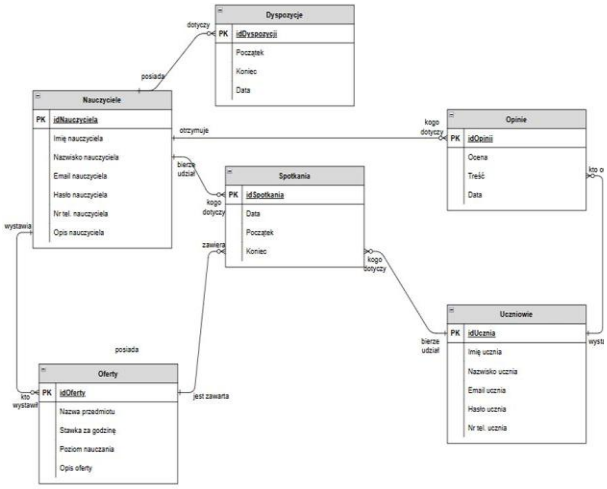
Inne wykorzystane narzędzia to **Bootstrap** – framework CSS zawierający mnóstwo użytecznych narzędzi do tworzenia interfejsu graficznego aplikacji internetowych, **Composer** [2] **–** system zarządzania pakietami dla języka PHP (menedżer zależności), **phpMyAdmin –** narzędzie do zarządzania bazą danych MySQL.

2.2. Diagram związków encji [3]

Jednym z fundamentów dobrego projektu informatycznego jest przemyślany i przejrzysty diagram związków encji – ERD (ang. *Entity Relation Diagram*). Jego głównym zadaniem jest wizualizacja struktury bazy danych. Dobry diagram ERD powinien być zrozumiały dla osoby również spoza branży informatycznej. Diagram związków encji pokazuje zależności między elementami bazy danych. Składa się z encji, atrybutów oraz związków. W kontekście diagramu ERD należy wspomnieć o mocy powiązania, którą rozumiemy

jako maksymalną liczbę instancji jednej encji, które mogą być powiązane z instancją innej encji. Istnieją trzy najważniejsze typy powiązań:

- jeden do jednego;
- jeden do wielu;
- wiele do wielu.



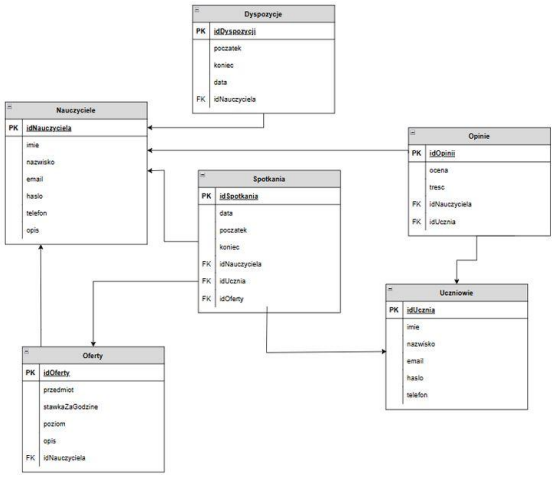
Rys. 2. 1. Diagram związków encji

2.3 Schemat relacyjnej bazy danych [4]

Następnym nieodłącznym elementem projektu aplikacji webowej jest schemat relacyjnej bazy danych. Opiera się on na relacjach, inaczej tabelach, w których przechowywane są dane. Każda tabela zawiera klucz główny – PK (ang. Primary Key), będący unikalnym identyfikatorem relacji. Tabele mogą zawierać klucz obcy – FK (ang. Foreign Key), który odnosi się do klucza głównego innej tabeli, tworząc związki między tabelami.

Na poniższym rysunku przedstawiono schemat relacyjnej bazy danych, który powstał w wyniku transformacji modelu związków encji do postaci schematu relacyjnej bazy danych wraz z zachowaniem następujących reguł:

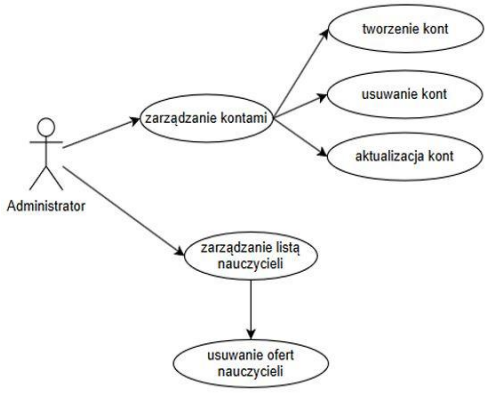
- odzwierciedlenie encji w relację; nazwa encji (l. pojedyncza) jest odwzorowywana w nazwę relacji (l. mnoga);
- odzwierciedlenie atrybutu encji w atrybut relacji; nazwa atrybutu encji jest odwzorowywana w nazwę atrybutów relacji;
- odzwierciedlenie typu danych atrybutu encji w typ danych atrybutu relacji;
- transformacja identyfikatora encji w klucz podstawowych relacji;
- opcjonalność lub obowiązkowość atrybutów encji informuje, czy klucz obcy relacji jest typu null czy not null;
- transformacja ograniczeń integralnościowych do odpowiadających im ograniczeń integralnościowych relacji.



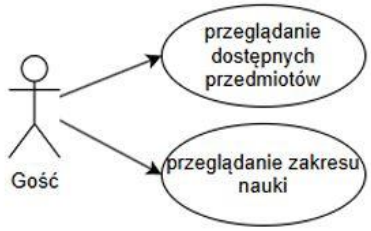
Rys. 2. 2. Schemat relacyjnej bazy danych

2.4 Diagram przypadków użycia

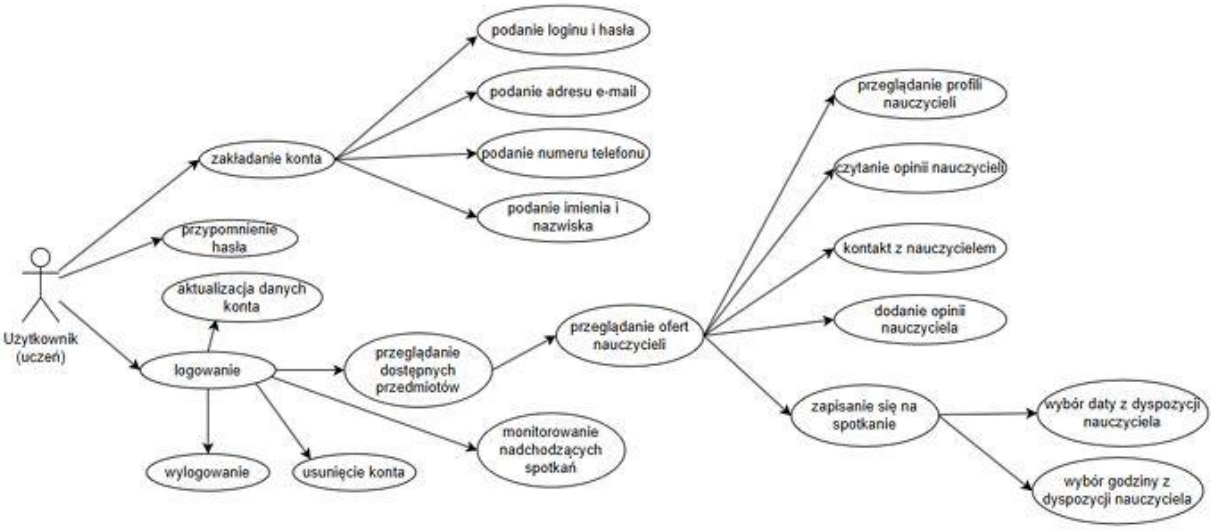
Diagram przypadków użycia (DPU) jest diagramem behawioralnym w zunifikowanym języku modelowania (ang. *Unified Modeling Language*, *UML*). Opisuje wymagania funkcjonalne programu, może zostać wykorzystany w celu prostego przedstawienia działania systemu [5]. Fundamentem diagramów przypadków użycia jest zidentyfikowanie różnych użytkowników systemu – aktorów [6]. Poniżej diagram przypadków użycia dla 4 aktorów: Administrator (osoba uprawniona do zarządzania systemem), Gość (niezalogowana osoba odwiedzająca stronę), Użytkownik z podziałem na Ucznia i Nauczyciela.



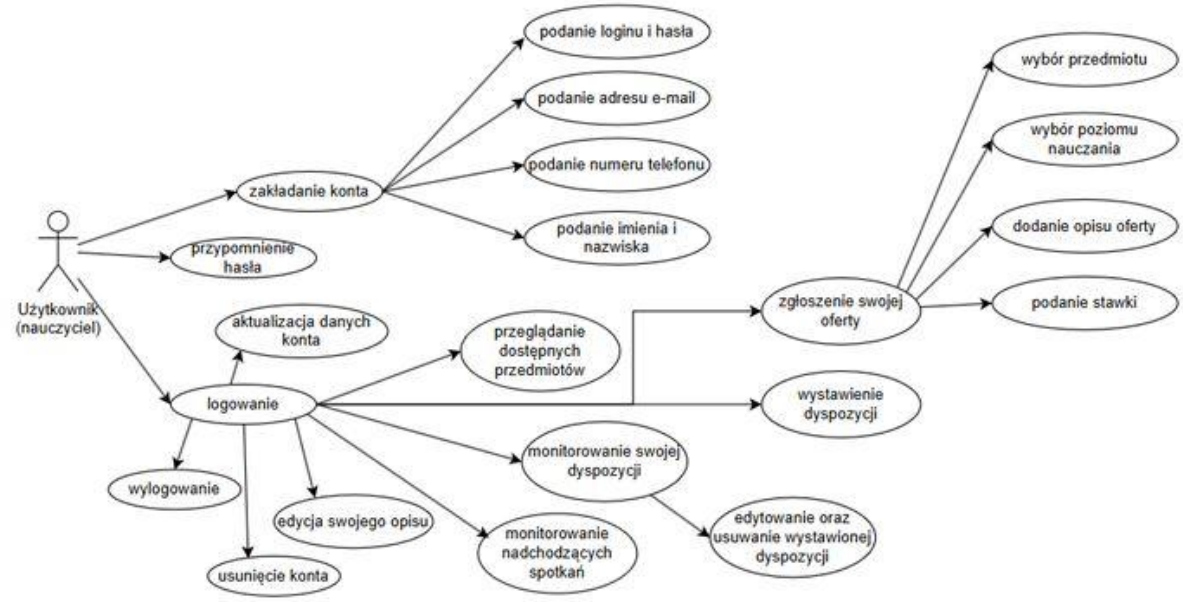
Rys. 2. 3. DPU dla Administratora



Rys. 2. 4. DPU dla Gościa



Rys. 2. 5. DPU dla użytkownika zalogowanego z uprawnieniami ucznia



Rys. 2. 6. DPU dla użytkownika zalogowanego z uprawnieniami nauczyciela

3. Implementacja

3.1. Wymagania funkcjonalne

Wymagania funkcjonalne odzwierciedlają zamierzone zachowanie systemu lub tego, co system zrobi. Zachowanie to można wyrazić jako usługi,

zadania lub funkcji, którą system ma wykonać [7]. Są to konkretne opisy definiujące funkcje wykonywane przez oprogramowanie systemu.

Każdy scenariusz podzielony jest na sekcje:

- przypadek użycia – nazwa przypadku użycia;
- aktorzy – użytkownik i elementy systemu biorący udział podczas wykonywania konkretnego przypadku użycia;
- wymagania – warunki, które muszą zostać spełnione, by umożliwić wykonanie scenariusza;
  - scenariusz główny – przebieg zdarzeń przedstawiający optymalną realizację opisanej funkcjonalności;
  - scenariusz poboczny – alternatywny przebieg zdarzeń, gdy wystąpią odstępstwa od założonego scenariusza głównego.

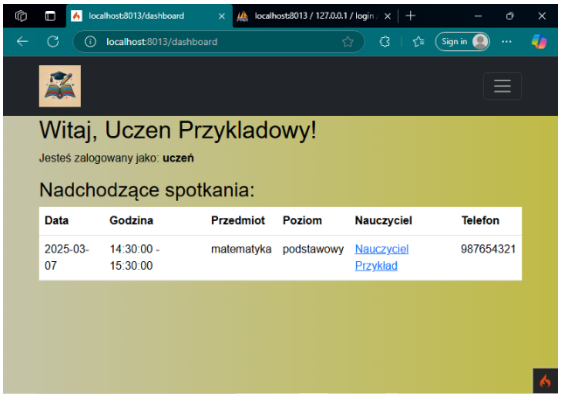
Funkcjonalności, które zostały zaimplementowane w aplikacji obejmują m.in. rejestrację i logowanie użytkownika, aktualizację profilu lub hasła, przeglądanie i korzystanie z ofert nauczycieli, umawianie się na spotkanie, wystawianie ocen nauczycielom, dodawanie swoich ofert (jako nauczyciel).

3.2 Wymagania pozafunkcjonalne

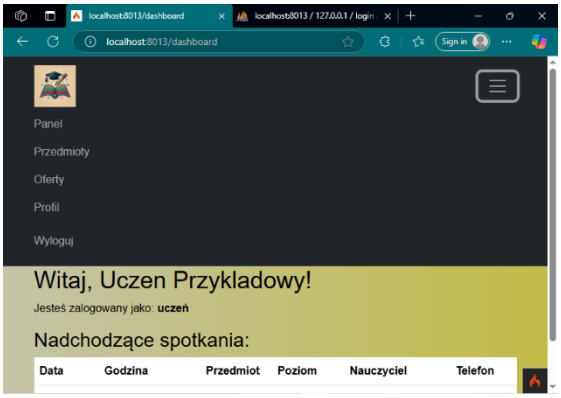
Wymagania pozafunkcjonalne nie są bezpośrednio związane z funkcjonalnością systemu, ale skupiają się na takich rzeczach, jak na przykład: jakość i prostota korzystania z aplikacji, dostosowanie do różnych urządzeń, podatność na błędy lub weryfikacja użytkowników. Obejmują m.in. weryfikację użytkowników, wystawianych ogłoszeń i ocen oraz system powiadomień. Te systemy nie zostały zaimplementowane w projekcie, lecz są przykładem do dalszego rozwoju aplikacji.

3.3 Strona internetowa

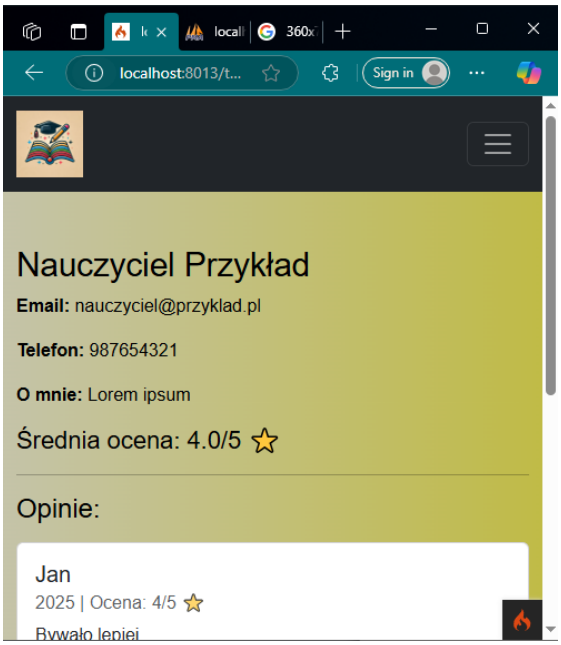
Aplikacja powstała przy użyciu wykorzystanych narzędzi, w zgodzie z opisem poszczególnych etapów projektu. Interfejs, który widzi użytkownik, jest prosty i intuicyjny. Aplikacja z założenia miała być łatwa w obsłudze dla każdego. Oto kilka zrzutów ekranu z aplikacji.



Rys. 3. 1. Dashboard zalogowanego ucznia



Rys. 3. 2. Pasek nawigacyjny, umożliwiający korzystanie ze strony na urządzeniach mobilnych



Rys. 3. 3. Profil nauczyciela

4. Podsumowanie

W ramach niniejszej pracy zaprojektowano aplikację webową. Zrealizowano cele założone na początku pracy nad projektem, a aplikacja spełnia postawione wymagania funkcjonalne. Treści przedstawione są w sposób prosty i przejrzysty. Użytkowanie aplikacji jest intuicyjne. Celem projektu było stworzenie aplikacji umożliwiającej znajdywanie uczniów i korepetytorów, a także wdrożenie systemu, który pozwoli w pewnym stopniu zautomatyzować proces umawiania spotkać. Nauczyciele w aplikacji mogą dodać opis siebie, wystawić dyspozycję oraz utworzyć ofertę. Uczniowie mają możliwość przeglądać oferty nauczycieli, umawiać się z nimi na spotkania i oceniać korepetytorów. Aplikacja jest w pełni funkcjonalna, co nie oznacza, że nie może zostać rozbudowana. Pierwszorzędnym elementem jest zwiększenie bezpieczeństwa użytkowników. Jedną z propozycji jest weryfikacja nauczycieli i dodawanych ofert, co jednocześnie ograniczy spam oraz ryzyko wykorzystywania aplikacji niezgodnie z przeznaczeniem. Innym elementem rozbudowy projektu może być implementacja wewnętrznego komunikatora, który pozwoliłby całkowicie wyeliminować przymus komunikacji za pomocą telefonu lub poczty elektronicznej.

Bibliografia

1. Oficjalna strona CodeIgniter: https://www.codeigniter.com/user_guide/
2. Oficjalna strona Composer: https://getcomposer.org/
3. D. Olczyk: Modelowanie strukturalne – definicje, notacja, techniki i narzędzia. WWSI, 2010, s87-98.
4. W. Dąbrowski i inni: ITA-101 Bazy danych. Warszawa, 2009.
5. R. Fauzan et al.: A different approach on automated use case diagram semantic assessment. INASS, 2020, s1-3.
6. R. Malan, D. Bredemeyer: Functional Requirements and Use Cases, 2001.
7. Ibidem.