IMPLEMENTACJA INTERAKTYWNEGO KURSU PROGRAMOWANIA W TECHNOLOGII WEBOWEJ

Jakub Swacha[1]

Uniwersytet Szczeciński
Wydział Nauk Ekonomicznych i Zarządzania
[1] e-mail: jakubs@wneiz.pl

Słowa kluczowe | informatyczne wspomaganie nauczania, nauka programowania, e-learning

Streszczenie

W artykule podjęto problem implementacji interaktywnego kursu programowania w technologii webowej. Opisano przykład udanej implementacji takiego kursu opartej na wykorzystaniu standardowych technologii webowych i zestawu gotowych komponentów, uwzględniając m.in.: wymagania stawiane interaktywnemu kursowi programowania, architekturę i technologię rozwiązania, niezbędne składniki i format zapisu treści kursu, komponenty i układ interfejsu użytkownika, sposób edycji i wykonywania kodu rozwiązania oraz zasady weryfikacji poprawności rozwiązań i generowania podpowiedzi dla uczestników kursu.

Wprowadzenie

O tym, że nauka programowania jest trudna, świadczą zarówno wyniki badań opinii prowadzone wśród studentów (Simon i in., 2009), jak i przeprowadzone na międzynarodową skalę pomiary efektów nauki, które obnażają jej powszechną niską skuteczność (McCracken i in., 2001). Wśród zidentyfikowanych barier utrudniających skuteczną naukę programowania wymienia się m.in. brak: możliwości samodzielnej praktyki oraz dostępu do niezwłocznej informacji zwrotnej (*instant feedback*) i zrozumiałych wyjaśnień (Rogerson i Scott, 2010).

Narzędziem dydaktycznym wolnym od wymienionych wyżej barier jest interaktywny kurs programowania. Prócz nauczanych treści udostępnia on interpreter nauczanego języka programowania, pozwalający na samodzielne wykonywanie przez studenta zadań bez potrzeby korzystania z innego narzędzia, a zazwyczaj także automatyczną ocenę zgłaszanych rozwiązań ćwiczeń oraz sugestie poprawek odnoszących się do błędów wykrytych w niepoprawnym rozwiązaniu. O bardzo korzystnym wpływie dostępności automatycznej oceny rozwiązań i opartej na niej niezwłocznej informacji zwrotnej na efekty uczenia się programowania świadczą wyniki badań Fernandeza Alemana (2011).

Rozwój technologii webowych stworzył możliwość realizacji interaktywnych kursów programowania działających w środowisku przeglądarki WWW. Rozwiązania takie mają oczywistą zaletę polegającą na możliwości korzystania z kursu bez potrzeby instalowania jakiegokolwiek oprogramowania poza przeglądarką WWW na komputerze uczestnika kursu, co również oznacza możliwość korzystania z kursu niezależnie od rodzaju urządzenia i systemu operacyjnego, którego używa osoba ucząca się.

Obecnie dostępnych jest już wiele interaktywnych kursów programowania opartych na takiej technologii, udostępnianych przez różne platformy internetowe, z których najpopularniejszą jest Codecademy (Colao, 2014). Analizę porównawczą tej i innych platform tego rodzaju autor niniejszego tekstu zawarł we wcześniejszym opracowaniu (Swacha, 2016b).

Mimo to istnieją ważne przesłanki, które mogą uzasadniać samodzielną implementację interaktywnego kursu programowania. Może to być np.:

- wybrany do nauki język programowania,
- język, którym posługują się uczestnicy kursu,
- dostosowanie treści kursu do zamierzonych celów nauczania, w tym:
  a) część języka mająca być objęta kursem,
  b) biblioteki i rozszerzenia mające być przedmiotem nauki,
  c) specyficzne wymagania technologiczne dotyczące formy prezentacji wyników działania programu (np. nauka programowania grafiki, dźwięku, gier interaktywnych),
- zapewnienie ścisłego wglądu osobom prowadzącym kurs w przebieg nauki,
- uniezależnienie się od zewnętrznych dostawców, np. ze względu na:
  a) koszty korzystania z ich usług,
  b) słabe połączenie internetowe ze światem,
  c) zabezpieczenie majątkowych praw autorskich do opracowanego przez siebie kursu (platformy umożliwiające publikowanie własnych kursów mogą wymagać zrzeczenia się tych praw),
  d) niepewności co do kontynuowania działania platformy na dotychczasowych zasadach i dostępności kursu w obecnej postaci w przyszłości.

W niniejszym opracowaniu podjęto się zademonstrowania sposobu implementacji interaktywnego kursu programowania w technologii webowej na przykładzie autorskiego interaktywnego kursu języka Python, którego aktualna wersja dostępna jest pod adresem http://uoo.univ.szczecin.pl/~jakubs/kurs (Swacha, 2016a).

1. Specyfikacja wymagań

Wobec kursu będącego przedmiotem niniejszego opracowania określono następujące wymagania:

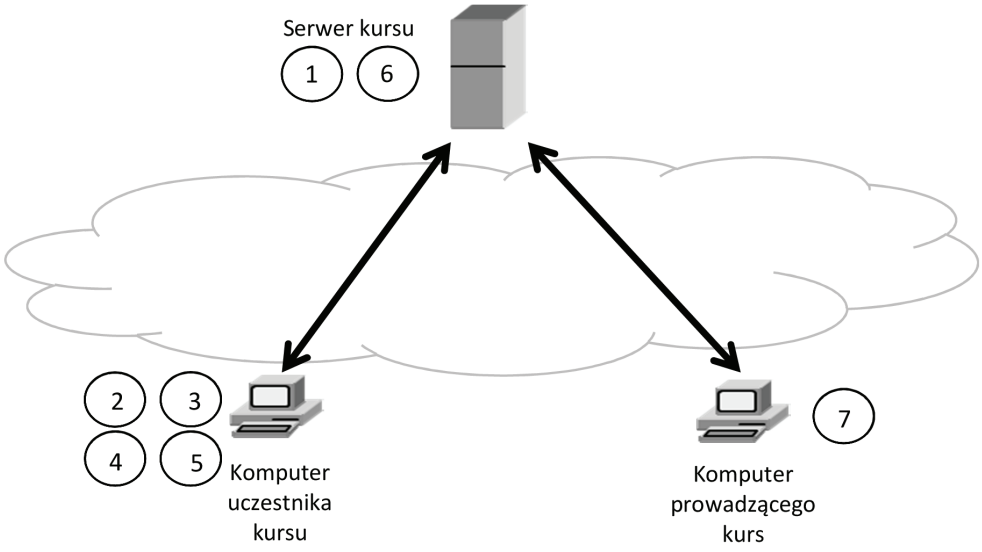
1. Odnośnie do środowiska pracy:
   1.1. Działanie w przeglądarce internetowej, bez konieczności instalacji interpretera nauczanego języka programowania na komputerze uczestnika kursu.
2. Odnośnie do funkcjonalności dla uczestnika kursu:
   2.1. Edycja kodu rozwiązania bezpośrednio w oknie przeglądarki.
   2.2. Uruchamianie kodu rozwiązania bezpośrednio z okna przeglądarki.
   2.3. Automatyczne sprawdzanie poprawności rozwiązania.
   2.4. Wyświetlanie wyniku działania kodu rozwiązania bezpośrednio w oknie przeglądarki.
   2.5. Wyświetlanie automatycznych podpowiedzi do nieprawidłowego rozwiązania.
   2.6. Trwałe zapamiętywanie postępów w nauce (kończenia kolejnych zadań) z odtwarzaniem stanu zaawansowania kursu przy kolejnej wizycie na stronie kursu.
3. Odnośnie do funkcjonalności dla prowadzącego kurs:
   3.1. Informacja o zadaniach ukończonych przez poszczególnych uczestników.
   3.2. Wgląd w treść przesłanych rozwiązań i dodatkowe informacje obejmujące: datę i godzinę przesłania rozwiązania, długość czasu edycji i liczbę wprowadzonych zmian.

Większość wymienionych wyżej wymagań wynika wprost z celu budowy kursu. Słowa wyjaśnienia wymagają jedynie pozycje: 2.6 (potrzeba poinformowania uczestnika kursu o tym, gdzie dotąd zaszedł – szczególnie uzasadniona w przypadku dłuższej przerwy w nauce) i 3.2 (umożliwienie wykrycia prostych rodzajów manipulacji – tj. czy faktycznie nadesłano poprawne rozwiązanie, czy nadesłano je w ustalonym terminie i czy sposób jego uzyskania nie budzi wątpliwości – np. bardzo krótki czas edycji, niewielka liczba wprowadzonych zmian – które wskazywałyby, że skopiowano gotowe rozwiązanie).

2. Architektura i technologia rozwiązania

Kluczową decyzją dotyczącą architektury interaktywnego kursu programowania w technologii webowej jest umieszczenie interpretera nauczanego języka programowania i skryptów odpowiedzialnych za automatyczną ocenę rozwiązania po stronie serwera lub klienta (przeglądarki). Oba rozwiązania mają swoje wady i zalety (zob. tab. 2 w: Swacha, 2017b), z których jako główną zaletę pierwszego można wskazać uniemożliwienie manipulacji wynikiem oceny, a jako główne zalety drugiego: radykalne obniżenie wymagań co do niezbędnych zasobów po stronie serwera, brak zagrożenia dla bezpieczeństwa serwera z powodu wykonywania na nim niesprawdzonego kodu oraz brak opóźnienia mogącego wynikać z obciążenia serwera pomiędzy przesłaniem rozwiązania a otrzymaniem wyników oceny. W opisywanej implementacji wybrano to drugie rozwiązanie (zob. rys. 1), ponieważ przedmiotem implementacji jest kurs podstaw

programowania, gdzie ryzyko manipulacji jest niewielkie (uczestnicy zainteresowani są przyswojeniem przekazywanej wiedzy, a nie jak najszybszym ukończeniem kursu), a jej praktyczne konsekwencje zerowe (osiągnięcie efektów kształcenia będzie bowiem weryfikowane innymi metodami).



Rysunek 1. Podział zadań pomiędzy warstwą serwerową i kliencką kursu

Legenda: 1) przechowanie treści zadań, 2) wyświetlenie treści zadania, 3) edycja kodu rozwiązania, 4) wykonanie kodu rozwiązania i ocena poprawności, 5) wyświetlenie wyniku wykonania i oceny rozwiązania, 6) przechowanie rozwiązań poszczególnych uczestników kursu, 7) wyświetlenie informacji o wykonanych zadaniach i rozwiązań poszczególnych uczestników kursu.

Źródło: opracowanie własne.

Implementację przeprowadzono z wykorzystaniem standardowych technologii webowych:
- HTML i CSS do nadania struktury i stylizacji interfejsu użytkownika,
- JavaScript z biblioteką jQuery do przygotowania skryptów działających po stronie przeglądarki (dynamiczne elementy interfejsu użytkownika, interpretacja i sprawdzenie poprawności rozwiązania, przesłanie rozwiązania na serwer),
- PHP do przygotowania skryptów działających po stronie serwera (kontrola logowania, rejestracja ukończonych zadań, wgląd prowadzącego w treść przesłanych rozwiązań).

3. Treść kursu: zakres danych, ich format i sposób wczytywania

Mimo iż implementacja dotyczyła pojedynczego kursu (podstaw programowania w języku Python, przygotowanego w języku polskim), przyjęto, że rozwiązanie może zostać w przyszłości zaadaptowane na potrzeby innych kursów. Dlatego z kodu programu wyodrębniono metadane dotyczące samego kursu, których listę przedstawia tabela 1.

Tabela 1. Zestawienie danych opisujących kurs

Nazwa pola	Zawartość
title	tytuł kursu
language	język uczestników kursu
programminglanguage	język programowania będący przedmiotem kursu
author	autor kursu
email	adres poczty elektronicznej autora kursu
version	wersja kursu
created	data opracowania pierwszej wersji kursu
edited	data ostatniej modyfikacji kursu
lessons	lista tytułów lekcji zawartych w kursie

Źródło: opracowanie własne.

Każdą z lekcji kursu umieszczono w osobnym pliku, zawierającym listę składających się na nią ćwiczeń, każde zapisane w postaci zestawu pól zawierających definiujące je informacje (patrz tab. 2).

Tabela 2. Zestawienie danych opisujących lekcję w ramach kursu

Nazwa pola	Zawartość
title	nazwa zadania
intro	treść dydaktyczna przygotowująca do wykonania zadania
task	opis zadania programistycznego
initcode	stan początkowy edytora kodu
inputHas	lista wzorców tekstowych, które muszą zostać znalezione w kodzie źródłowym rozwiązania, by uznać je za prawidłowe
inputHasHint	lista podpowiedzi do wyświetlenia w przypadku nieodnalezienia poszczególnych wzorców z listy *inputHas*
inputHasNot	lista wzorców tekstowych, które nie mogą zostać znalezione w kodzie źródłowym rozwiązania, by uznać je za prawidłowe
inputHasNotHint	lista podpowiedzi do wyświetlenia w przypadku odnalezienia poszczególnych wzorców z listy *inputHasNot*
errorHas	lista wzorców tekstowych, które odnalezione w treści komunikatu o błędzie wykonania spowodują wyświetlenie podpowiedzi
errorHasHint	lista podpowiedzi do wyświetlenia w przypadku odnalezienia poszczególnych wzorców z listy *errorHas*
outputHas	– lista wzorców tekstowych, które muszą zostać znalezione w wyniku wykonania rozwiązania, by uznać je za prawidłowe

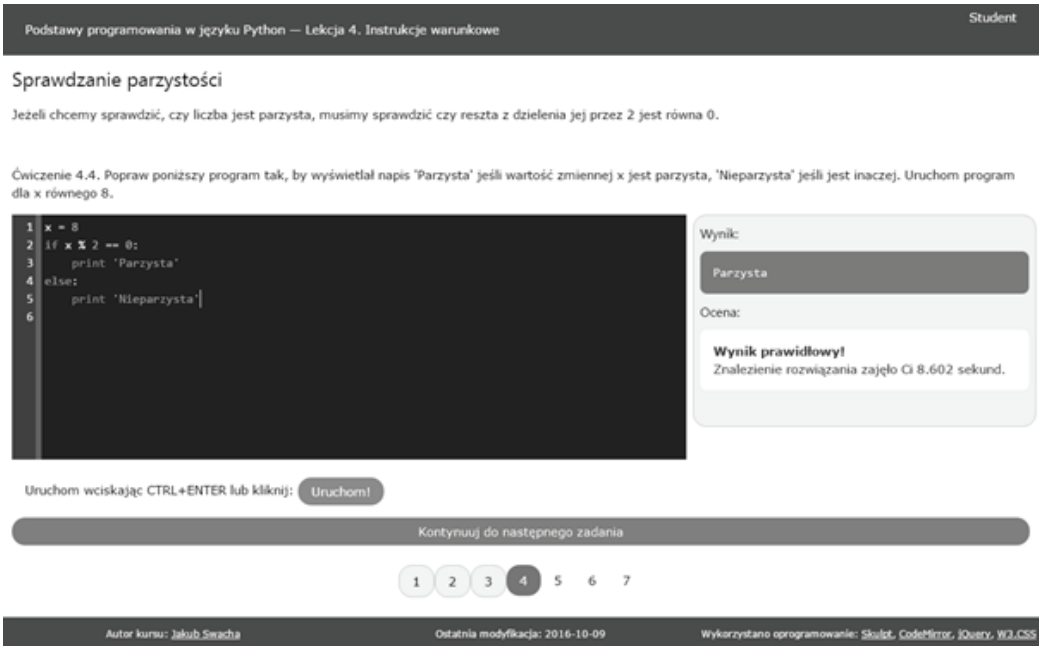
Źródło: opracowanie własne.

Jako format danych kursu wybrano JSON (Bray, 2014), co pozwoliło na zaimplementowanie prostego sposobu zmiany aktualnej lekcji: pliki danych mają formę poprawnego kodu JavaScript, dzięki czemu możliwe jest ich ładowanie poprzez wstrzykiwanie do obiektowego modelu dokumentu wyświetlanej strony kursu elementu HTML *Script*, odnoszącego się do pliku danej lekcji.

4. Interfejs użytkownika kursu

Rysunek 2 przedstawia zrzut ekranu podczas rozwiązywania jednego z zadań kursu. Rozpoznać na nim można (patrząc od lewego górnego rogu):

- nagłówek kursu i pole wyboru lekcji (pasek na samej górze),
- przycisk logowania do kursu (w prawym górnym rogu),
- nazwę, wprowadzenie i treść zadania (białe pole poniżej nagłówka kursu),
- pole edycji rozwiązania (duży ciemny prostokąt po lewej stronie),
- panel prezentacji wyników (jasnoszary zaokrąglony prostokąt z widocznymi polami: ciemnym „wynik" i jasnym „ocena"; pole „podpowiedzi" jest ukryte ze względu na prawidłowy rezultat),
- przycisk uruchamiania wprowadzonego kodu,
- przycisk przejścia do kolejnego zadania,
- przyciski nawigacyjne pokazujące ukończone i nieukończone zadania i pozwalające na przemieszczanie się pomiędzy nimi,
- pasek informacyjny (na samym dole).



Rysunek 2. Przykładowa strona kursu

Źródło: opracowanie własne.

Do stylizacji wizualnej interfejsu użytkownika wykorzystano framework W3.CSS (W3Schools, 2015) ze względu na jego niewielki rozmiar i fakt, że zawiera wszystkie kompo-

nenty niezbędne do implementacji strony kursu (m.in. responsywne panele, rozwijane menu, okna dialogowe).

5. Edycja kodu rozwiązania

Na wrażenia uczestników kursu nie powinien wpływać negatywnie edytor służący do wprowadzania kodu źródłowego stanowiącego rozwiązanie ćwiczenia. Pożądane cechy pola edycji obejmują przede wszystkim: numerowanie linii, kolorowanie składni dostosowane do języka będącego przedmiotem nauki oraz wspomaganie edycji (np. automatyczne wcięcia instrukcji blokowych czy automatyczne domykanie nawiasów). Wyklucza to użycie standardowego pola edycji tekstu, które takich udogodnień nie posiada.

Wszystkie je zapewnia natomiast CodeMirror (Haverbeke, 2011), napisany w JavaScript i działający w środowisku przeglądarki WWW edytor kodu źródłowego dostosowany do składni wielu języków programowania (w tym języka Python). Dlatego w opisywanym kursie wykorzystano pole edycji oparte na tym rozwiązaniu.

W momencie załadowania zadania, treść podana w przypisanym mu polu *initcode* wpisywana jest do pola edycji. W momencie gdy uczestnik kursu decyduje się na uruchomienie swojego rozwiązania, treść pola edycji przekazywana jest do interpretera.

6. Wykonywanie kodu rozwiązania

Wymóg działania w przeglądarce internetowej, bez konieczności instalacji interpretera nauczanego języka programowania na komputerze uczestnika kursu, oznacza konieczność dołączenia interpretera do samego kursu. Z kolei podjęta decyzja architektoniczna – umieszczenie interpretera nauczanego języka programowania i skryptów odpowiedzialnych za automatyczną ocenę rozwiązania po stronie przeglądarki – wymusza, by miał on postać programu działającego w środowisku skryptowym przeglądarki.

Samodzielne uzyskanie interpretera spełniającego ten ostatni wymóg jest realne dzięki dostępności kompilatorów generujących na wyjściu kod JavaScript, takich jak np. Emscripten (Zakai, 2011). Niemniej w przypadku języka Python, będącego przedmiotem nauczania w opisywanym kursie, nie było potrzeby robienia tego ze względu na dostępność gotowych interpreterów tego języka – w opisywanym kursie wykorzystano oprogramowanie Skulpt (Graham, 2011).

7. Weryfikacja poprawności rozwiązań i generowanie podpowiedzi

Po wykonaniu przez interpreter kodu stanowiącego rozwiązanie zadania, następuje weryfikacja jego poprawności. W pierwszej kolejności sprawdzany jest wynik działania: zostanie uznany za prawidłowy tylko wtedy, gdy zawiera wszystkie wzorce tekstowe zdefiniowane w polu *outputHas*. Następnie, niezależnie od prawidłowości rozwiązania, sprawdzane jest występowanie wzorców (zapisanych jako wyrażenia regularne) w treści rozwiązania. Służą one do wygenerowania podpowiedzi, a także do uznania rozwiązania, mimo prawidłowego wyniku, za

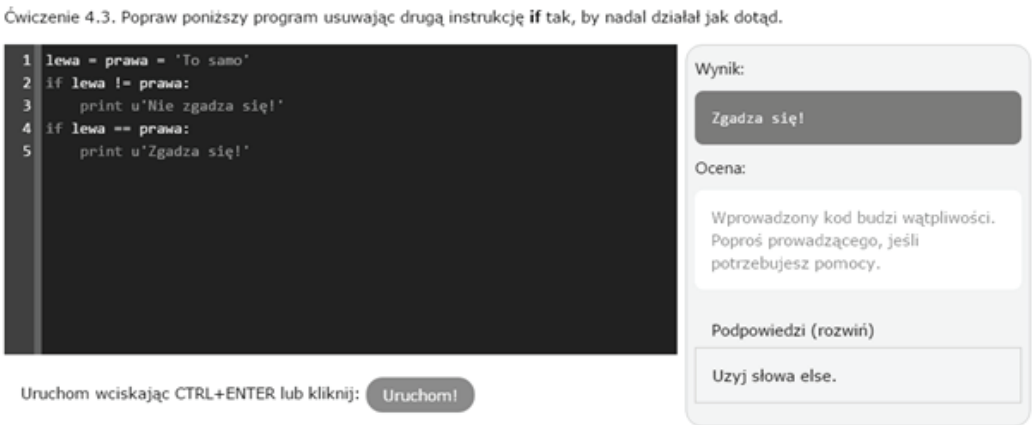
niezgodne z zasadami. Za zaliczone uważa się tylko zadania, dla których wprowadzono rozwiązania zarazem prawidłowe i zgodne z zasadami.

Wzorce z grupy *inputHas* nadają się najlepiej do wykrycia braku w rozwiązaniu prawidłowych wartości wejściowych i elementów języka, których użycie było wymienione explicite w opisie zadania. Obmyślając wzorce dla tej grupy należy zachować ostrożność, ponieważ ten sam poprawny rezultat można zwykle uzyskać na różne sposoby, w tym także nie zawierające elementów, które mogłyby wydawać się oczywiste twórcy kursu.

Wzorce z grupy *inputHasNot* nadają się najlepiej do wykrycia zamieszczenia w rozwiązaniu elementów języka, których użycie było zabronione explicite w opisie zadania oraz typowych błędów zasługujących na wygenerowanie podpowiedzi.

Wzorce z grupy *errorHas* pozwalają na wygenerowanie podpowiedzi stanowiących dodatkowy komentarz do komunikatu o błędzie wykonania, co jest szczególnie przydatne, gdy dany błąd wykonania łatwo twórcy kursu (a trudno uczestnikowi) przypisać do błędu logicznego specyficznego rodzaju.

Wygenerowane podpowiedzi są ukryte przed wzrokiem uczestnika kursu. Dopiero kliknięcie na przypisanym im przycisku powoduje ich wyświetlenie (patrz rys. 3).



Rysunek 3. Przykład niezaliczenia zadania mimo poprawnego wyniku i podpowiedzi

Źródło: opracowanie własne.

8. Rejestracja postępów nauki

Wprowadzenie poprawnego (i uznanego za zgodne z zasadami) rozwiązania powoduje oznaczenie zadania jako zaliczone. W przypadku zalogowania się uczestnika kursu (z kursu można bowiem korzystać także bez logowania) informacja o stanie ukończenia zadań w poszczególnych lekcjach przechowywana jest na serwerze, co ułatwia kontynuowanie nauki po dłuższej przerwie. Ponadto każde rozwiązanie zaliczające zadanie wprowadzone przez zalogowanego uczestnika jest przesyłane na serwer, wraz z dodatkowymi informacjami obejmującymi

datę i godzinę przesłania rozwiązania, długość czasu edycji i liczbę wprowadzonych zmian, co pozwala na późniejsze sprawdzenie jakości i sposobu uzyskania (czas, liczba edycji) rozwiązania przez prowadzącego kurs.

Podsumowanie

Postęp technologii webowej stworzył techniczną możliwość implementacji interaktywnych kursów programowania wymagających do działania jedynie przeglądarki internetowej. Mimo dostępności w internecie gotowych kursów dla różnych języków programowania, w praktyce mogą wystąpić przesłanki uzasadniające samodzielną implementację takiego kursu. Właśnie opisanie sposobu przeprowadzenia takiej implementacji (Swacha, 2016a) jest przedmiotem niniejszego opracowania. Z pewnością zaproponowane tu rozwiązania mogą posłużyć za wzór dla kolejnych implementacji tego typu, nawet dotyczących innych języków programowania i zakresów merytorycznych.

Opisane tu rozwiązanie będzie także przedmiotem dalszych prac samego autora. Po pierwsze trwają starania nad wprowadzeniem do niego części koncepcji gamifikacyjnych zaproponowanych oryginalnie dla zgamifikowanej platformy e-learningowej (Swacha, Baszuro, 2013), która jak dotąd jeszcze nie powstała. Drugi kierunek prowadzonych badań to ewaluacja kursu przez studentów, szczególnie interesująca na tle innych dostępnych rozwiązań służących temu samemu celowi (Swacha, 2017a).

Literatura

Bray, T. (red.) (2014). The JavaScript Object Notation (JSON) Data Interchange Format. RFC 7159, IETF.

Colao, J.J. (2014). With 24 Million Students, Codecademy Is Bigger Than You Thought, "Forbes" Pobrane z: http://www.forbes.com/sites/jjcolao/2014/04/23/with-24-million-students-codecademy-is-bigger-than-you-thought (16.10.2016).

Fernandez Aleman, J.L. (2011). Automated assessment in a programming tools course, *IEEE Transactions on Education*, *54*(4), 576–581.

Graham, S. (2011). *Skulpt*. Pobrane z: http://www.skulpt.org (16.10.2016).

Haverbeke, M. (2011). *CodeMirror*. Pobrane z: http://codemirror.net (data16.10.2016).

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.-D., Laxer, C., Thomas, L., Utting, I., Wilusz, T. (2001). A multinational, multi-institutional study of assessment of programming skills of first-year CS students, ACM SIGCSE Bulletin, 33(4), 125–140. DOI: 10.1145/572133.572137.

Rogerson, C., Scott, E. (2010). The Fear Factor: How It Affects Students Learning to Program in a Tertiary Environment, *Journal of Information Technology Education*, *9*, 147–171.

Simon, B., Hanks, B., McCauley, R., Morrison, B., Murphy, L., Zander, C. (2009). For Me, Programming is... W: *Proceedings of the Fifth International Workshop on Computing Education Research Workshop* (s. 105–116). New York: ACM. DOI: 10.1145/1584322.1584335.

Swacha, J. (2016a). *Interaktywny kurs języka Python*. Pobrane z: http://uoo.univ.szczecin.pl/~jakubs/kurs (16.10.2016).

Swacha, J. (2016b). Webowe środowiska do nauki programowania. W: A.B. Kwiatkowska i M.M. Sysło (red.), *Informatyka w edukacji. Kształcenie informatyczne i programowanie dla wszystkich uczniów* (s. 109–118). Toruń: Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika.

Swacha, J. (2017a). *Interaktywny kurs języka Python: realizacja i ocena studentów* (w przygotowaniu).

Swacha, J. (2017b). Scripting Environments of Gamified Learning Management Systems for Programming Education. W: R.A. Peixoto de Queirós i M. Teixeira Pinto (red.), *Gamification-Based E-Learning Strategies for Computer Programming Education* (s. 278–294), Hershey: Information Science Reference. DOI: 10.4018/978-1-5225-1034-5.ch013.

Swacha, J., Baszuro, P. (2013). Gamification-based e-learning platform for computer programming education. W: N. Reynolds, M. Webb (red.), *Learning while we are connected. Vol. 1: Research papers* (s. 122–130). Toruń: Wydawnictwo Naukowe UMK.

W3Schools (2015). W3.CSS. Pobrane z: http://www.w3schools.com/w3css (16.10.2016).

Zakai, A. (2011). Emscripten: an LLVM-to-JavaScript compiler. W: *Proceedings of the ACM International Conference Companion on Object Oriented Programming Systems Languages and Applications Companion* (s. 301–312). New York: ACM. DOI: 10.1145/2048147.2048224.

IMPLEMENTING AN INTERACTIVE PROGRAMMING COURSE USING WEB TECHNOLOGY

KEYWORDS

IT-supported education, computer programming education, e-learning

ABSTRACT

The paper discusses the problem of implementing an interactive programming course using web technology. An example of a successful implementation of such a course using standard web technology and ready-made components has been described, including such aspects as: the requirements for an interactive programming course, the architecture and technology of the solution, the necessary components and the format of the course content, the components and layout of the user interface, the solutions chosen for code editing and execution, and the rules for verifying the correctness of exercise solutions and generating hints for course participants.