Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Jakub Szafrański
nr albumu: 259064
Praca inżynierska
na kierunku informatyka stosowana

System automatycznego sprawdzania skryptów bashowych

Opiekun pracy dyplomowej
dr hab. Jacek Kobus
Instytut Fizyki

Toruń 2016

Pracę przyjmuję i akceptuję

.....
data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

.....
data i podpis pracownika dziekanatu

Uniwersytet Mikołaja Kopernika zastrzega sobie prawo własności niniejszej pracy magisterskiej w celu udostępniania dla potrzeb działalności naukowo-badawczej lub dydaktycznej

Spis treści

1 Wstęp **3**

2 Architektura systemu sprawdzającego **6**
2.1 Środowiska uruchomieniowe 7
2.2 Centralny punkt zarządzania 9
2.2.1 Definicje repozytoriów, egzaminów, zadań i scenariuszy 9

3 Implementacja **10**
3.1 Struktura plików 10
3.2 Wzorzec projektowy interfejsu WWW 12

4 Instalacja i konfiguracja **14**
4.1 Wymagania systemowe 14
4.2 Instalacja 14
4.2.1 Cykliczne sprawdzanie 16
4.3 Podstawowa konfiguracja 17
4.3.1 Sekcja „database” 17
4.3.2 Sekcja „debug” 18
4.3.3 Sekcja „email_error_reporting” 19
4.3.4 Sekcja „vars_environment” 21
4.3.5 Przykładowy plik konfiguracyjny 21
4.4 Baza danych 22
4.5 Uruchamianie systemu sprawdzającego 23
4.5.1 Instalacja serwera aplikacji Phusion Passenger 24
4.5.2 Uruchamianie 25

5 Konfiguracja systemu **27**
5.1 Pierwsze konto 27
5.2 Logowanie do interfejsu WWW 27
5.3 Logowanie do interfejsu administracyjnego 29
5.4 Dodawanie grupy środowisk uruchomieniowych 30
5.5 Dodawanie środowiska uruchomieniowego 32
5.6 Dodawanie repozytorium 34
5.7 Dodawanie egzaminu 37
5.8 Dodawanie zadań 39
5.9 Dodawanie scenariuszy sprawdzających 40

6 Strona dla studentów **45**
6.1 Strona główna 45

6.2 Strona z wynikami egzaminu 47
6.2.1 Wyniki pojedynczego użytkownika 48
6.2.2 Lista wyników wszystkich studentów 49
6.2.3 Tabela z wynikami wszystkich studentów 50

7 Podsumowanie **52**

Bibliografia **52**

Spis rysunków **54**

Rozdział 1

Wstęp

Studenci kierunku *Informatyka Stosowana* w trakcie swoich studiów muszą zdobyć podstawową wiedzę z zakresu działania dostępnych na rynku systemów operacyjnych, ze szczególnym naciskiem na systemy z rodziny GNU/Linux. W części teoretycznej przedmiotu studenci zdobywają wiedzę na temat podstaw działania współczesnych systemów operacyjnych, zaś w części laboratoryjnej ćwiczą korzystanie z podstawowych narzędzi, które są dostępne w większości dystrybucji systemu GNU/Linux. M.in. studenci muszą opanować podstawy języka Bash i dlatego w ramach ćwiczeń muszą napisać (i zaliczyć) szereg mniej lub bardziej skomplikowanych skryptów w tym języku.

Używany obecnie system do oceny nadsyłanych przez studentów rozwiązań, to zbiór skryptów napisanych w językach Bash i Perl (wykorzystujący dodatkowo system Sphinx do generowania strony z wynikami), które weryfikują poprawność działania programów przesłanych przez studentów, oceniając je w pewnej, zdefiniowanej skali punktowej. Studenci na początku zajęć dostają dostęp do przygotowanego wcześniej repozytorium, do którego mogą przesyłać napisane przez siebie skrypty. System sprawdzający monitoruje wspomniane repozytorium, a następnie periodycznie dokonuje sprawdzenia skryptów, które zmieniły się od ostatniego czasu. Po sprawdzeniu wystawiana jest propozycja oceny, którą student może podejrzeć pod wskazanym adresem www. Oceniane jest między innymi standardowe wyjście oprogramowania napisanego przez studenta, jego dokumentacja lub zmiany które zaszły w systemie po jego wykonaniu (np. pliki, które zostały zmodyfikowane w trakcie jego działania).

Używany obecnie system posiada jednak kilka istotnych wad:

- kod jest mało modularny, trudno rozwijać oprogramowanie o dodatkowe możliwości lub poprawiać ewentualne błędy,
- skrypty oceniane są przy wykorzystaniu zwykłego konta, z którego korzysta także system sprawdzający i system generujący stronę z wynikami,
- kontrola nad środowiskiem, w którym wykonywane są skrypty nie jest zunifikowana. Definiując zadanie nie można w łatwy sposób stworzyć „idealnego" środowiska (dotyczy to zarówno zmiennych środowiskowych jak i plików/zasobów dostępnych dla oprogramowania napisanego przez studenta).

Celem niniejszej pracy było opracowanie i budowa systemu oceniającego (o nazwie *Grader*), który byłby pozbawiony wymienionych wyżej wad. W systemie *Grader* zostały

wprowadzone dwie podstawowe „warstwy" działania – oprogramowanie dzieli się na moduł główny, koordynujący uruchamianie zadań studentów, zbierający wyniki i dokonujący oceny oraz moduł sprawdzający, którego rolą jest jedynie uruchamianie zadań nadesłanych przez studentów i dostarczenie do centralnego punktu systemu informacji o wyniku działania ich zadań. Dzięki takiemu podejściu system charakteryzuje się następującymi zaletami:

- kod źródłowy jest logicznie podzielony na moduły i jest czytelny, co ułatwia jego analizę, usuwanie usterek i ewentualny rozwój,
- do sprawdzania skryptów wykorzystywany został protokół SSH; daje to możliwość sprawdzania skryptów na zdalnych serwerach (na dowolnym koncie uniksowym), a użytkownik nie jest w stanie wpłynąć na system sprawdzający poprzez nadesłanie odpowiednio spreparowanego rozwiązania,
- ułatwienie zarządzania egzaminami i zadaniami poprzez zastosowanie interfejsu graficznego obsługiwanego za pośrednictwem przeglądarki internetowej.

Aplikacja została podzielona na mniejsze moduły odpowiedzialne za pojedyncze, konkretne funkcjonalności. Jeśli nie było to możliwe do osiągnięcia, to moduły obejmują ściśle ze sobą powiązane funkcjonalności.

Zastosowane w nowej wersji oprogramowania wzorce projektowe umożliwiły oddzielenie logiki biznesowej (implementacji procesów biznesowych, która odpowiada za funkcjonalną część aplikacji, modelując „prawdziwe" zachowania) od logiki aplikacji (szczegółów technicznej implementacji, odpowiedzialnych za koordynowanie interakcji pomiędzy warstwą biznesową a warstwą prezentacji), oraz uwydatniły granice odpowiedzialności pomiędzy poszczególnymi modułami, co z kolei umożliwiło zastosowanie takich rozwiązań jak testy jednostkowe[1].

Aplikacja została napisana według najnowszych (w chwili pisania pracy) standardów programowania. Oprogramowanie zostało napisane przy wykorzystaniu języka Python[1] w wersji 3.4. Aplikacja wykorzystuje zalety szkieletu aplikacji Django[2].

Niniejsza praca zawiera opis architektury całego systemu uzupełniony dokumentację użytkownika i administratora, która zawiera wszystkie informacje niezbędne do skonfigurowania i korzystania z systemu. Kod źródłowy aplikacji jest udokumentowany wewnętrznie w taki sposób, by nawet osoba, która pierwszy raz go czyta mogła w łatwy sposób znaleźć interesujący ją fragment, zrozumieć jego działanie i móc wprowadzić modyfikacje, które nie przyczynią się do negatywnego działania całego systemu.

Dokumentacja kodu została oparta o *Python docstrings*[3], dzięki czemu można ją wygenerować za pomocą narzędzia Sphinx. Osoba która edytuje kod będzie miała do niej dostęp z poziomu środowiska deweloperskiego, którego używa (wiele IDE, takich jak np. *Pycharm*[4] potrafi dokumentację zawartą w tym formacie wczytać, przetworzyć i przedstawić programiście w wygodny sposób, a nawet przy jej pomocy sprawdzać, czy dany moduł lub funkcja jest używany prawidłowo).

Układ niniejszej pracy jest następujący. W rozdziale 2. omówiona została architektura nowego systemu sprawdzającego – pokrótce opisany jest każdy składnik systemu, jego

[1] Uruchamianie testów jednostkowych i integracyjnych w aplikacji korzystającej ze szkieletu Django zostało opisane w dokumentacji tego szkieletu: `https://docs.djangoproject.com/en/1.11/topics/testing/overview/#running-tests`

przeznaczenie oraz implementacja. Rozdział 3. poświęcony jest analizie poszczególnych kroków koniecznych do przeprowadzenia instalacji, wstępnej konfiguracji i uruchomienia jednostki centralnej systemu. W rozdziale 4. opisana jest konfiguracja systemu przez jego administratora – zawiera instrukcję, którą należy wykonać w celu zdefiniowania każdego egzaminu oraz sposobu jego oceny. Rozdział 5. opisuje stronę WWW, którą widzą studenci sprawdzający, jak zostały ocenione ich skrypty.

Do pracy dołączona jest płyta CD zawierająca tekst pracy oraz kod źródłowy całego programu. *Grader* udostępniony jest na licencji MIT[5].