Жуков Николай Николаевич
ассистент кафедры
ФГБОУ ВПО «Российский государственный
педагогический университет им. А.И. Герцена»
г. Санкт-Петербург

РАЗРАБОТКА СИСТЕМЫ АВТОМАТИЗИРОВАННОЙ ПРОВЕРКИ ЗАДАНИЙ СТУДЕНТОВ

***Аннотация**: в статье содержится описание системы автоматизированной проверки Javascript сценариев, осуществляемой в рамках дисциплины «Программирование» для подготовки будущих инженеров по направлению «Информатика и вычислительная техника» на кафедре компьютерных технологий и электронного обучения института компьютерных наук и технологического образования РГПУ им. А.И. Герцена.*

***Ключевые слова**: JavaScript, тестирование, автоматизация, система, автоматизированная система, Phantom, JSON.*

Цель разработки данной системы – упростить проверку заданий, выполненных студентами по дисциплине «Программирование», подготовка по которой осуществляется в рамках направления «Информатика и вычислительная техника» на кафедре компьютерных технологий и электронного обучения института компьютерных наук и технологического образования РГПУ им. А.И. Герцена.

Методика её использования детально описана в предыдущей статье автора [3], а также статьях других авторов [1; 2]. В этой статье будет рассмотрена структура этой системы и кратко описаны её компоненты.

Общая структура системы представляет совокупность трех модулей: (1) модуля сбора (ввода) данных, (2) модуля хранения результатов и (3) модуля проверки задания (рис. 1). Кратко опишем каждый из них и основные технические особенности их реализации.

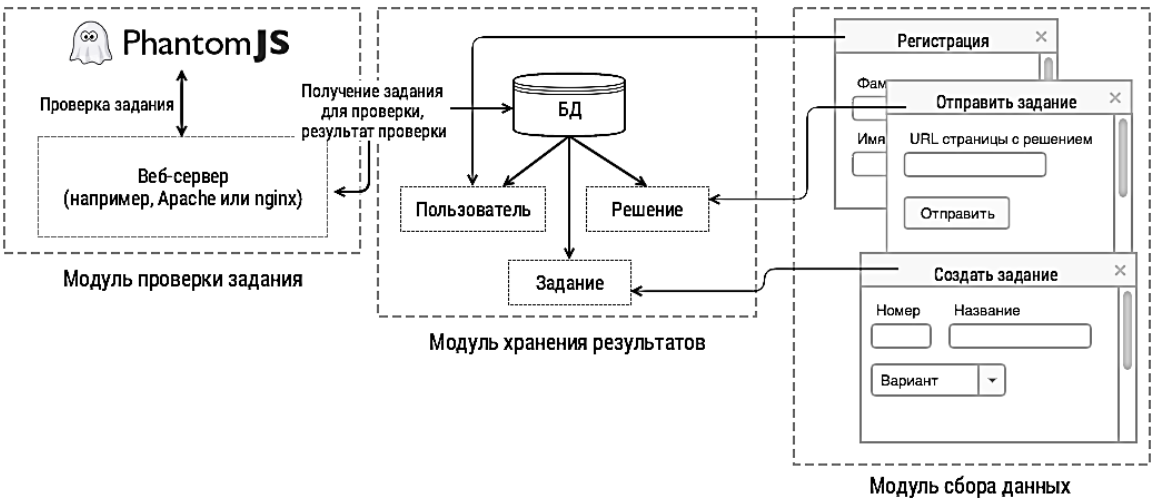

Рис. 1. Общая структура системы автоматизированной проверки заданий студентов

Модуль сбора (*ввода*) *данных* представлен на схеме набором основных веб-страниц с формами (1) регистрации нового студента в системе, (2) создания вариантов заданий преподавателем и (3) отправки задания на проверку (на схеме отображены только основные формы).

После заполнения формы регистрации в системе студенту присваивается уникальный идентификатор, который будет использоваться при выдаче ему персонального варианта задания. Как только студент завершил регистрацию он может получить задание и отправить его на проверку.

Модуль хранения результатов представляет собой базу данных, содержащую в себе информацию о созданных заданиях и их вариантах, зарегистрированных студентах, результатах выполнения ими заданий.

Приведем название и структуру наиболее важных сущностей:

– пользователь (номер студента, фамилия, имя, отчество студента, электронная почта, пароль для доступа, идентификатор варианта);

– решение задачи (идентификатор варианта, идентификатор студента, ссылка на решение, время тестирования);

– задание (номер задания, идентификатор варианта, описание задачи, сигнатура задания – пары входных и выходных наборов значений);

– результат (идентификатор студента, идентификатор задания, количество баллов).

Для хранения данных может использоваться как реляционная СУБД (например, MySQL), так и документно-ориентированная (например, MongoDB). Сущности также могут представлять собой JSON-файлы, что позволит упростить развертывание системы и позволит использовать её без использования сети интернет.

Приведем пример (сущности «Задание») для задачи построения ряда Фибоначчи в формате JSON (рис. 2).


```
{
    "taskid": "1",
    "variantid": "1",
    "description": "Построить ряд Фибоначчи с возможностью задавать количество элементов ряда",
    "inputs": [{
        "n": "10"
    }],
    "outputs": [{
        "answ": [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
    }],
    "boardurl": "http://kodaktor.ru/g/bb1f8b5_a5156"
}
```


Рис. 2. Описание сущности задания в JSON

Модуль проверки задания требуется для организации процедуры сравнения результата выполнения задачи студентом с ответом преподавателя. В его основе – Phantom.js, представляющий WebKit-движок без графического интерфейса (headless), в котором реализована поддержка Javascript API и множества современных веб-стандартов таких как:

– DOM (Document Object Model), CSS;

– Canvas, SVG;

– JSON.

Phantom.js имеет открытый исходный код и может быть запущен на всех платформах (Linux, Windows, MacOS) через командную строку.

В представленном ниже фрагменте (рис. 3) программного кода (расположенного также по адресу: clck.ru/9gW3F) демонстрируется открытие веб-страницы с помощью Phantom.js, что является основным действием для организации

сравнения результата, полученного студентом с правильным ответом, предоставляемым преподавателем.

```
page.open(board, function(status) {
        if (status !== 'success') {
                console.log("Unable to open URL " + boardRequest);
                        phantom.exit();
        }
        else {
                var result = page.evaluate(function(arg) {
                        if (arg == document.title)
                                return true;
                        else
                                return false;
                }, corrAnsw);
        }
}
```

Рис. 3. Фрагмент кода для проверки задания студента

При выполнении этого сценария с помощью Phantom.js в памяти будет создан новый объект («*page*») – образ страницы (DOM) с содержимым, размещенным по URL-адресу (переменная *board*) и внутри метода *evaluate* может быть выполнен javascript-код в контексте содержимого этого объекта. Необходимо подчеркнуть, что код выполняется в специальном режиме – «песочнице» так, что веб-страница не имеет доступа к Phantom.js. Приведенный выше фрагмент кода позволяет осуществлять сравнение результата, полученного студентом (и размещенного внутри тега *title*) и правильного ответа, предоставленного преподавателем. Как только запрос на проверку правильности выполнения задания будет получен системой, она, используя идентификатор варианта, выполнит сравнение результатов и в случае, если ни для одной из сигнатур задания не было возвращено ошибки, в таблицу результатов соответствующего задания будет записан балл, заранее определенный преподавателем. В обратном случае, в базу данных будет внесена информация об этой попытке.

Дальнейшая работа по усовершенствованию системы может заключаться в модификации модуля сбора данных и создании отдельного модуля, который позволит взаимодействовать с LMS (например, Moodle) посредством SCORM или Tin Can API (например, выставлять в них баллы). Модуль сбора данных может

быть дополнен функцией проверки ввода текста (typing verification), при котором сравнивается как пользователь вводил текст при регистрации и как он вводит текст при отправке задания. Кроме этого, при проверке заданий может проверяться схожесть проверяемой работы с уже находящимися в модуле хранения результатами на предмет использования заимствованного программного кода.

Список литературы

1. Аксютин П.А. Опыт построения среды электронного обучения и ее использование для преподавания дисциплины «Информационные технологии» // Электронное обучение в вузе и в школе: Материалы сетевой международной научно-практической конференции. – СПб.: Астерион, 2014. – С. 28–31.

2. Государев И.Б. Электронное обучение веб-программированию на основе технологий тестирования JavaScript-сценариев [Текст] / И.Б. Государев // Педагогический опыт: теория, методика, практика: Материалы III Междунар. науч.-практ. конф. (Чебоксары, 31 июля 2015 г.) / Редкол.: О.Н. Широков [и др.]. – Чебоксары: ЦНС «Интерактив плюс», 2015. – С. 71–73.

3. Жуков Н.Н. Подготовка будущих инженеров к тестированию программного обеспечения [Текст] / Н.Н. Жуков // Научное и образовательное пространство: перспективы развития: Материалы Междунар. науч.-практ. конф. (Чебоксары, 29 нояб. 2015 г.) / Редкол.: О.Н. Широков [и др.]. – Чебоксары: ЦНС «Интерактив плюс», 2015. – [Электронный ресурс]. – Режим доступа: https://interactive-plus.ru/discussion_platform.php?requestid=14478