УДК 004.414.3

РАЗРАБОТКА ИГРОВОГО ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ ЯЗЫКУ ПРОГРАММИРОВАНИЯ JAVA С ИСПОЛНЕНИЕМ КОДА В РЕАЛЬНОМ ВРЕМЕНИ

Л. Р. Нуруллина[1], Д. Д. Ильясов[2], А. И. Хайруллин[3], Р. Р. Мирхусаинов[4], М. Р. Сидиков[5], М. М. Абрамский[6], А. Р. Ахметшин[7]

[1-7] *Высшая школа информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета*

[1]sadenliia@gmail.com, [2]damirilyasov1997@gmail.com, [3]az.khayrull@gmail.com, [4]mirhusainow@gmail.com, [5]sidikov.marsel@gmail.com, [6]ma@it.kfu.ru, [7]raakhme@gmail.com

Аннотация

Разработан прототип приложения, обучающего в игровой форме базовому синтаксису языка Java. Рассмотрены вопросы связи между реализацией игрового процесса и обучающими упражнениями, в которых должен исполняться программный код. Приведены геймплей и архитектура клиентской и серверной частей.

***Ключевые слова**: язык программирования java, обучение программированию, игровые обучающие приложения, клиент-серверные приложения, фреймворк Spring*

ВВЕДЕНИЕ

Большинство современных профессиональных разработчиков в свое время обучалось программированию со значительной долей самостоятельной работы [1]. Игровой подход к обучению может существенно облегчить обучение синтаксису языков программирования [2]. Существует несколько успешных проектов для разных языков программирования, которые используют его. Один из ярких примеров – RubyWarrior [3] – онлайн-игра, позволяющая управлять героем с помощью написания программного кода. С ее помощью можно сравнительно быстро обучиться базовым конструкциям и синтаксису языка программирования Ruby.

По данным ресурса StackOverflow.com на 2018 год, Java входит в топ-5 популярных языков программирования [1]. Существует несколько игровых онлайн-приложений для обучения синтаксису Java, самые популярные среди них – JavaRush [4] и CodinGame [5]. На JavaRush материал дается в рамках некоторой истории из игрового сюжета, с которой связывается задание. Но выполнение задания не вызывает никакой интерактивной реакции в пользовательском интерфейсе, а также пользователю дается для прочтения большой объем материала. У CodinGame есть определенный сюжет, связанные с ним задания и их графическая интерпретация, но она не связана в реальном времени с логикой кода, который пишет обучающийся. В связи с этим появилась идея создать игровое онлайн-приложение, где пользователь мог бы управлять героем с помощью программного кода, тем самым изучая синтаксис и базовые конструкции языка программирования.

В настоящей работе рассмотрены основные вопросы реализации подобного приложения. В разделе 1 описана концепция игры. В разделе 2 представлены описание разработки серверной части приложения, ее архитектура. Раздел 3 представляет создание браузерного приложения. В разделе 4 обсуждены вопросы проверки правильности кода и его исполнения.

1. ГЕЙМПЛЕЙ

В основу концепции по аналогии с приложением [3] было решено положить историю о герое, который преодолевает препятствия и сражается с врагами на пути к основной цели (прохождению уровня). Управление героем осуществляется посредством написания программного кода, описывающего движения игрока. Такая модель дает возможность неограниченно создавать новые уровни, повышая их уровень сложности. Разработка игры строилась на основе следующего сюжета.

Главный персонаж работает в подразделении службы специального назначения (S.W.A.T.), его отправляют на задание, финальной целью которого является поимка опасного преступника. На начальном этапе в качестве игровых препятствий было решено создать «пики» и «врагов», которые могут наносить урон.

Пользователь может совершать четыре основных действия: передвижение, передвижение в прыжке, атака врага, отдых – для восстановления очков здоровья (*health points, HP*).

Существует возможность проверки ситуаций: наличие врага, наличие препятствия, количество HP.

Таким образом, пользователь может проверять возможные опасности на поле и в зависимости от них определять, какие действия должен сделать герой. Схема взаимодействий изображена на gameplay-диаграмме (рис. 1).

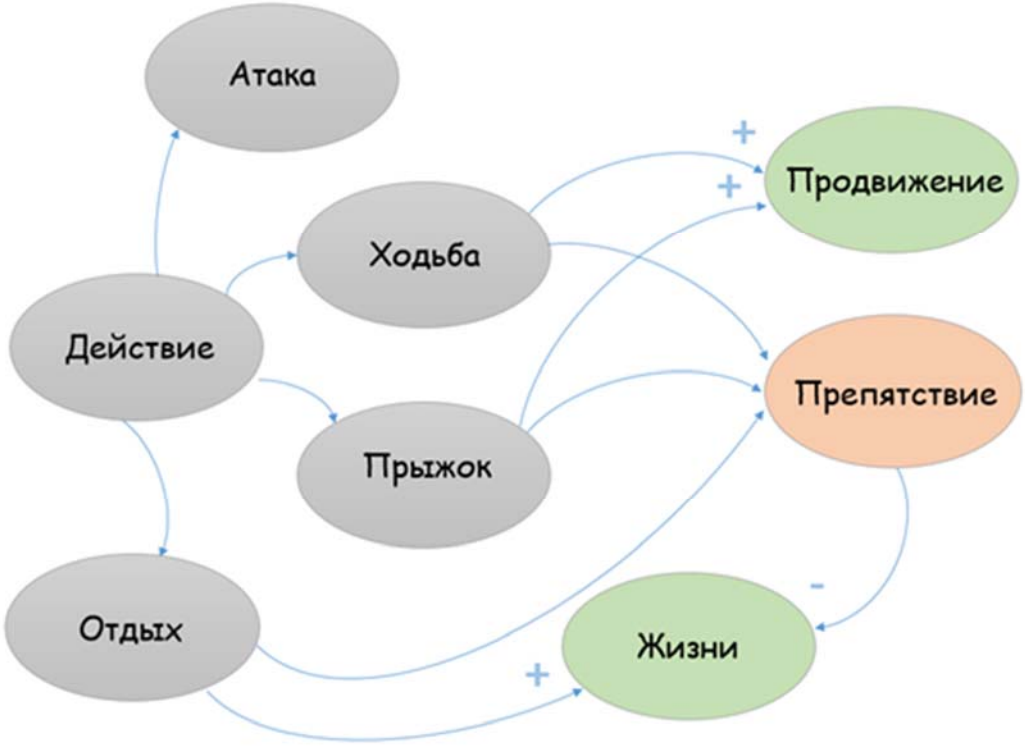

Рис. 1. Gameplay-диаграмма

Если герой в движении попадает на препятствие, то теряет очки HP, иначе – продвигается вперед. Если герой отдыхает – HP увеличиваются. Но если при этом он был рядом с врагом, то одновременно его HP уменьшатся.

Игра поделена на уровни. С каждым уровнем увеличиваются сложность, а также количество навыков, которые нужно применить. На первом уровне доступна всего одна возможность, то есть один метод – метод ходьбы. Таким образом, пользователь должен понять, как вызываются методы. На втором уровне появляется метод для прыжка. Далее добавляются препятствия. С их появлением реализуется метод проверки их наличия. При появлении методов для определения ситуации программисту необходимо также научиться писать условия.

Поле с расставленными на нем препятствиями, а также его размерность будут генерироваться на каждом уровне случайным образом, но с учетом сложности

уровня. Такой подход дает пользователю возможность проходить игру заново, даже если она уже была пройдена им.

В игре предусмотрена возможность создавать аккаунт для сохранения результата и создания рейтинговых таблиц, поскольку соревновательный аспект может добавить игроку мотивации для улучшения собственных результатов.

2. АРХИТЕКТУРА СЕРВЕРНОЙ ЧАСТИ

За основу при разработке серверной части был взят фреймворк Spring Boot [6]. Сервер имеет контроллер с одним главным методом, который принимает на вход код, написанный пользователем (игроком) в виде строки (тип String). Данный код обрабатывается слоем сервисов, затем возвращается ответ. С помощью перечисленной в ответе информации клиентское приложение отображает визуализацию действий персонажа и результаты прохождения уровня.

Так может выглядеть модель ответа сервера:

```
{
    “actions” : [
        { “action” : “MOVE_FORWARD” }
        { “action” : “MOVE_FORWARD” }
    ],
    stageCompleted: true
}
```

Из ответа сервера клиентское приложение должно сделать вывод, что персонаж два раза сделал шаг вперёд и прошёл уровень.

Список всех классов-сервисов серверного приложения, через которые проходит пользовательский код, в том числе те, в которых он проверяется на корректность и исполняемость:

- CompileService – сервис, отвечающий за исполнение клиентского кода;
- ActionService – сервис, в котором содержатся все методы, доступные пользователю:
    - walk() – идти вперёд;
    - attack() – атаковать клетку перед собой;
    - jump() – перепрыгнуть одну клетку;
    - rest() – отдыхать (восполняется здоровье);
    - health() – метод возвращает текущее состояние здоровья;

- isEnemyAhead() – возвращает true, если перед персонажем враг, иначе false;
- isSpikesAhead() – возвращает true, если перед персонажем пики, иначе false;

- BanValidateService – сервис, проверяющий пришедший от пользователя код на предмет возможности его компиляции и наличие вредоносных участков.

Для обеспечения безопасности данных пользователей используется Spring Security [7] в сочетании с JWT[2]. Обработка кода игроков, авторизация, регистрация осуществляются на одном сервере. Подробнее об обработке пользовательского кода рассказано в разделе 4.

3. РАЗРАБОТКА БРАУЗЕРНОЙ ВЕРСИИ

Для разработки браузерной версии игры было решено использовать подход одностраничного веб-приложения. Одностраничные приложения (Single-Page Applications, сокр. SPA) – это веб-приложения, состоящие из одного единственного html-документа, содержимое которого динамически изменяется при взаимодействии с сервером под влиянием действий пользователя.

Задачи обновления данных на странице, предоставления переходов между представлениями, называемых роутингом, а также управления состоянием приложения полностью выполняет браузерная часть приложения. Работа пользователя с такими приложениями напоминает работу в настольных программах – действия пользователя не перезагружают интерфейс окна, а содержимое поступает к пользователю посредством его взаимодействия с приложением.

Для разработки одностраничного приложения выбран следующий список технологий:

- **ReactJS** – JavaScript библиотека для создания пользовательских интерфейсов, разработанная компанией Facebook. Расширяя ее возможности с помощью других JavaScript библиотек, таких, как React-Router для организации навигации внутри одностраничного приложения, а также Redux для хранения глобального состояния приложения, ReactJS становится полноценным инструментом для разработки одностраничных приложений [8];

[2] JWT – JSON Web Token

- **TypeScript** – скриптовый язык программирования, компилируемый в JavaScript. Язык расширяет синтаксис JavaScript статической типизацией, поддержкой полноценных классов и подключения модулей [9];
- **Webpack** – сборщик и оптимизатор модулей JavaScript. Он отслеживает зависимости каждого JavaScript скрипта, строя граф зависимостей, и собирает их в один выходной файл, одновременно решая задачу оптимизации и минимизации результатов сборки [10];
- **SASS** – язык на основе CSS, расширяющий возможности работы с файлами стилей. Добавляет в CSS поддержку переменных, функций, циклов, а также наследования и вложенных конструкций [11].

Создание одностраничного приложения в виде игры потребовало применения подходов, обеспечивающих высокий уровень масштабируемости и изменчивости:

- **Методология БЭМ (Блок, Элемент, Модель)** – методология, которая задает правила по именованию CSS-классов. Основана на трех определениях: блок, элемент, модификатор. **Блоком** называется самостоятельный элемент, являющийся независимым от других, который встраивается в страницы. **Элементом** называется составная неделимая часть блока, существование которой в отрыве от него невозможно. **Модификатором** называется сущность, которая определяет внешний вид, состояние и поведение блока или элемента. Примером CSS-класса, определенного с использованием БЭМ, является .header__logo-red, который указывает на логотип шапки сайта красного цвета. Применение данной методологии гарантирует разработчику изолированность стилей элементов на странице;
- **Компонентный подход** – подход, при котором приложение делится на самостоятельные модули, которые можно переиспользовать в различных частях приложения. Примером компонента в игре является компонент игровой карты (рис. 2), который также состоит из компонентов объектов карты, таких, как Главный герой, Шипы и Враг.

Рис. 2. Компонент «Игровая карта»

4. ВАЛИДАЦИЯ И ИСПОЛНЕНИЕ КОДА НА СЕРВЕРЕ

Перед отправкой кода на компиляцию необходимо провести процесс валидации, в котором можно выделить две основные задачи: выявление синтаксических ошибок и проверка пользовательского кода на безопасность.

Для выявления синтаксических ошибок в программном коде пользователя выбор был сделан в пользу готового решения – javax.tools.Diagnostic [12].

Метод для проверки синтаксиса находится в сервисе BanValidateServiceImpl и называется checkSyntax, параметр у которого один – String code (пользовательский код). Так как мы используем класс DiagnosticCollector[13], необходимо записать пользовательский код в специально подготовленный пустой файл и позже скомпилировать его с помощью JavaCompiler. После отработки метода getTask() в экземпляре класса DiagnosticCollector будет определен список Diagnostic – объектов, содержащих в себе информацию о синтаксических ошибках и предупреждениях компилятора. Из каждого экземпляра Diagnostic мы достаем:

- Kind (тип):
  - ERROR (ошибка);
  - WARNING (предупреждение);
  - MANDATORY_WARNING (обязательное предупреждение);
  - NOTE (примечание);
  - OTHER (другое);
- Местонахождение ошибки (включая номер строки и столбца);
- Сообщение.

Если ошибки будут обнаружены, информация о них передается в объект класса Validation с флагом isValid = false. Если ошибок нет, можно перейти к следующему этапу – проверке безопасности.

Возможность пользователя запускать практически любой Java-код на сервере влечет за собой большие риски. Чтобы эффективно противодействовать вредоносному коду, необходимо проанализировать направления возможной атаки:

- **Spring** – данный вектор атаки можно не учитывать, так как код компилируется в процессе выполнения программы, Spring не имеет доступа к скомпилированному классу;
- **throw** – с помощью этого ключевого слова пользователь может сгенерировать RuntimeException, которое сервер не обработает;
- **Рефлексия** – с помощью данного средства можно, например, подобрав имена классов, получить доступ к методам и полям;
- **File** и другие классы для работы с файлами – могут быть использованы для доступа к файловой системе машины, на которой запущен сервис;
- **Runtime** – может дать злоумышленнику доступ к командной строке (если это Windows) одной строчкой кода: Runtime.getRuntime().exec(“cmd”).

Оценив риски и проанализировав возможные векторы атаки, мы создали словарь запрещенных конструкций, которые пользователь не сможет использовать в своем программном коде. Сам процесс проверки кода на содержание запрещенных конструкций (далее – ЗП) реализует метод checkSecurity. В нем с помощью рефлексии и регулярных выражений подготавливается список ЗП и определяется их наличие. Если таковые были обнаружены, создается объект класса Validation с флагом isValid = false и сообщением об использованной ЗП, чтобы пользователь понимал, почему сервис отказался компилировать его код.

При отсутствии ошибок введенный код передается на выполнение. Так как он должен состоять только из операторов и определенных методов, заранее был подготовлен класс, из которого к ним есть доступ.

Данный класс содержит информацию о количестве доступных итераций и указатель на то, было ли совершено какое-либо действие ранее. На выходе мы получаем объект ***GameResult***, который содержит в себе результат исполнения поль-

зовательского кода или же сообщение об ошибке. Принцип работы основной процедуры класса заключается в циклическом выполнении введенного пользователем кода до тех пор, пока не закончатся доступные итерации или уровень не будет пройден.

ЗАКЛЮЧЕНИЕ

Разработан прототип веб-приложения, обучающего базовому синтаксису языка программирования Java. Несмотря на то, что изначально работа носила характер адаптации идеи приложения [3] на язык Java, в рамках проделанной работы была разработана архитектура клиент-серверного обучающего приложения с исполнением кода на Java-технологиях. Данные архитектуру и прототип можно использовать для разработки других подобных приложений, в том числе для обучения языкам программирования, исполняемым на Java Virtual Machine.

СПИСОК ЛИТЕРАТУРЫ

1. Developer Survey Results 2018. URL: https://insights.stackoverflow.com/survey/2018/
2. *Chang C.C., Liang C., Chou P. N., and Lin G. Y.* Is game-based learning better in flow experience and various types of cognitive load than non-game-based learning? Perspective from multimedia and media richness // Computers in Human Behavior. 2017.
3. Официальный сайт RubyWarrior. URL: https://www.bloc.io/ruby-warrior
4. Официальный сайт JavaRush. URL: https://javarush.ru/
5. Официальный сайт CodinGame. URL: https://www.codingame.com/
6. Официальный сайт Spring Boot. URL: https://spring.io/projects/spring-boot
7. Официальный сайт Spring Security. URL: https://spring.io/projects/spring-security
8. Официальный сайт ReactJS. URL: https://reactjs.org/
9. Официальный сайт TypeScript. URL: https://www.typescriptlang.org/
10. Официальный сайт WebPack. URL: https://webpack.js.org/
11. Официальный сайт SASS. URL: https://sass-lang.com/

12. Официальная документация java.tools.Diagnostic. URL: https://docs.oracle.com/javase/6/docs/api/index.html?javax/tools/Diagnostic.html

13. Официальная документация java.tools.DiagnosticCollector. URL: https:// docs.oracle.com/javase/7/docs/api/javax/tools/DiagnosticCollector.html

DEVELOPMENT OF APPLICATION FOR GAME-BASED LEARNING OF JAVA LANGUAGE WITH REALTIME CODE RUNNING

L. R. Nurullina[1], D. D. Iliyasov[2], A. I. Khairullin[3], R. R. Mirkhusainov[4], M. R. Sidikov[5], M. M. Abramskiy[6], A. R. Akhmetshin[7]

[1-7] *Higher School of Information Technologies and Intelligent Systems of Kazan (Volga Region) Federal University*

[1]sadenliia@gmail.com, [2]damirilyasov1997@gmail.com, [3]az.khayrull@gmail.com, [4]mirhusainow@gmail.com, [5]sidikov.marsel@gmail.com, [6]ma@it.kfu.ru, [7]raakhme@gmail.com

Abstract

This paper describes the development of the application prototype for learning Java language syntax. There are raised questions about the connection between gaming process implementation and learning exercises with the code running parts, and given the gameplay, architecture of client and server parts of the application.

Keywords*: java programming language, programming learning, game-based learning applications, client-server applications, Spring Framework*

REFERENCES

1. Developer Survey Results 2018. URL: https://insights.stackoverflow.com/survey/2018/

2. *Chang C.C., Liang C., Chou P. N., and Lin G. Y.* Is game-based learning better in flow experience and various types of cognitive load than non-game-based learning? Perspective from multimedia and media richness // Computers in Human Behavior. 2017.

3. RubyWarrior official page. URL: https://www.bloc.io/ruby-warrior
4. JavaRush official page. URL: https://javarush.ru/
5. CodinGame official page. URL: https://www.codingame.com/
6. Spring Boot official page. URL: https://spring.io/projects/spring-boot
7. Spring Security official page. URL: https://spring.io/projects/spring-security
8. ReactJS official page. URL: https://reactjs.org/
9. TypeScript official page. URL: https://www.typescriptlang.org/
10. WebPack official page. URL: https://webpack.js.org/
11. SASS official page. URL: https://sass-lang.com/
12. Official documentation of java.tools.Diagnostic. URL: https://docs. oracle.com/javase/6/docs/api/index.html?javax/tools/Diagnostic.html
13. Official documentation of java.tools.DiagnosticCollector. URL: https:// docs.oracle.com/javase/7/docs/api/javax/tools/DiagnosticCollector.html

СВЕДЕНИЯ ОБ АВТОРАХ

НУРУЛЛИНА Лия Радиковна – студентка Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Liia Radikovna NURULLINA – student of Higher School of ITIS KFU.

email: sadenliia@gmail.com

ИЛЬЯСОВ Дамир Дмитриевич – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Damir Dmitrievich IlYASOV – student of Higher School of ITIS KFU.

email: damirilyasov1997@gmail.com

ХАЙРУЛЛИН Азат Ильдарович – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Azat Ildarovich KHAYRULLIN – student of Higher School of ITIS KFU.

email: az.khayrull@gmail.com

МИРХУСАИНОВ Руслан Радикович – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Ruslan Radikovich MIRKHUSAINOV – student of Higher School of ITIS KFU.

email: mirhusainow@gmail.com

СИДИКОВ Марсель Рафаэлевич – руководитель лаборатории Java Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Marsel Rafaelevich SIDIKOV – head of Java Lab of ITIS KFU.

email: sidikov.marsel@gmail.com

АБРАМСКИЙ Михаил Михайлович – старший преподаватель кафедры программной инженерии Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Mikhail Mikhailovich ABRAMSKIY – senior lecturer at Department of Software Engineering of ITIS KFU

email: ma@it.kfu.ru

АХМЕТШИН Азат Ринатович – студент Высшей школы информационных технологи и интеллектуальных систем Казанского (Приволжского) федерального университета (ИТИС КФУ).

Azat Rinatovich AKHMETSHIN – student of Higher School of ITIS KFU.

email: raakhme@gmail.com

Материал поступил в редакцию 28 июля 2018 года