Development of the iKariera Work Portal

Filip Boros, Olga Chovancova

***Abstract*—**The aim of the thesis is the additional development of work portal iKariéra.cz built on Grails technology. The portal should serve students and graduates to find work positions. The application supports language mutations to Slovak and English languages. For the project was done a required upgrade from older to a later version because there is a major difference in both versions. There have been implemented new features for the portal, which were accustomed to requirements of IAESTE Slovakia. Application with the final solution of this work has been deployed to the server. Results of this thesis advanced project in its development and brought it closer to its final completion.

***Keywords*—** Gradle, Grails, Groovy, web application.

I. Introduction

The iKariera work portal will primarily serve students and graduates to find suitable work positions. It should be able to compete with other popular portals with a similar aim. That is why iKariera needs to be easy and comfortable to use and to be built on modern and reliable technologies. The original version of the project was at development phase, which was not yet suitable for deployment. It was built on older versions of technologies. It could not only mean potential security risks, but also lack of performance and stability compared to actual trend. Therefore, the first task was about analysis and upgrading of base technologies of the project. Next work was devoted to the implementation of new features for the portal according to the ideas of IAESTE Slovakia (The International Association for the Exchange of Students for Technical Experience). The project was then deployed to the accessible server for final testing and demonstration of new technologies and features.

II. Used technologies

Portal iKariera is complex web application built on framework Grails. The original version of the project used Grails 2.5.4. The goal was to make the upgrade to new version 3.1.9, which is a complete rework of previous versions. Grails uses and combines advantages of many technologies. It is necessary to mention the most important ones to grasp what Grails is about.

Groovy is an object-oriented programming language for the Java platform. It is very similar to Java programming language. The majority of syntax is similar, but the Groovy aims to simplify specific approaches for Java programmers. Groovy augments and enhances existing Java tools, adds new capabilities to existing Java classes, simplifies tests and many more [1]. Grails uses Groovy as its base programming language.

Gant is a tool for scripting Ant tasks using Groovy. It is used in building processes by Groovy and Java programs. Gant forms integrated a base system for building projects in Grails applications of 1.x and 2.x versions.

Gradle is an advanced tool for automatization of the project building. It combines the best from various build tools and adds its innovations on top of them [2]. Gradle was added for 3.x Grails versions where it replaced Gant. It is used for processes like a compilation, running tests and compression of application. Running Grails command runs competent Gradle command. Introduction of Gradle meant migration configuration files in Grails application into the new *build.gradle* file [3]. This file is also used for the definition of plugins.

F. Boros, University of Zilina, Faculty of Management Science and Informatics, Zilina, Slovakia (e-mail: boros.filip@gmail.com).
O. Chovancova, University of Zilina, Faculty of Management Science and Informatics, Zilina, Slovakia (e-mail: olga@chovancova.sk).

Spring is an open source framework for the development of applications build on Java. It offers complex support for application structure and allows developers to focus more on the development of application itself. Spring consists of many modules which provide different services. It reaches a certain level of abstraction and simplification of code because of that [4]. Grails builds on this framework and uses many of its useful features in processing data and their validation, in configurations, controller logic and many more.

Spring Boot is a tool used for simplification of running Spring-based applications, which otherwise require a lot of various complex configurations. Spring Boot does most of these configurations automatically which saves a lot of time and effort [5]. It became new base upon which are built 3.x versions of Grails.

Grails is web application framework, which contains all aspects of modern web development. It is based on MVC (Model View Controller) architecture. It uses features and technologies from various Java frameworks and combines them with innovations of development with dynamic programming languages. It offers the stability of technologies and simplifies configurations and running applications. It has access to all advantages and features of Java language because it runs on JVM (Java Virtual Machine). At the same time, it is written in Groovy, which allows access to all Groovy libraries and tools [6], [7]. Hibernate is used for the creation of data models and communication with the database. Spring framework is a base controller for Grails application. SiteMesh is a template framework for creating views. Grails binds all of these into one reliable technology.

III. Upgrade of the project

Upgrading from Grails 2.5.4 to Grails 3.1.9 means huge leap in the technological level of the project. It is because Grails 3.x came with most radical changes in Grails development history so far. Versions 3.x are a complete rework of previous versions on a base level. Basic principles remained the same, but the core was rewritten entirely on Spring Boot module and build on Gradle. Many files were removed, changed or added, like new configuration files in format YAML (YAML Ain't Markup Language). These significant changes mean many incompatibilities between both versions [9].

In general, there is need to stick to a sequence of precise steps to achieve fluent and successful upgrade of the project. Main steps are:

- Copy source codes including test files to new Grails 3.x project
- Put plugins definitions into new file *build.gradle*
- Migrate configuration files into new folders
- Remove all unnecessary files

There was also need to upgrade a few plugins into new versions. Plugin *Resources 1.2.7* used for managing static files (JavaScript, CSS) had no upgrade for new Grails version. Therefore it was replaced with plugin *Asset-Pipeline-Grails 2.8.2.*

Another significant change is related to a new feature in Grails 3.x – Interceptors. It is a replacement for filters from earlier versions. They were used to apply specific logic across multiple controllers. In iKariera project were filters used to define default language if none was detected by the user. In upgrade process of iKariera were filters rewritten by new Interceptors which also allowed to use new annotation *@GrailsCompileStatic [8].* This annotation written over method provides static compilations which offer better performance and response of the application.

IV. Implementation of new features

Language mutations to Czech and English languages were present in the original project but were incomplete in some cases and with occasional translation mistakes. Both of these languages were fixed, and there was added completely new translation to the Slovak language. Framework Grails uses for language mutations built-in plugin *i18n.* Translations are stored and defined in separated files. Each for one respective language. They contain expressions with their assigned keys, which are used to call them in the application. Language is switched by changing *Locale* object. Grails provides an easy way to do so by passing a value for parameter *lang* in URL (Uniform Resource Locator). Some pages in the original version of the application were crashing if the user switched languages in them. It was caused by missing URL mappings for these pages. This problem was fixed in this development phase.

Generating CV is important feature implemented for student user account. Generating takes information from the user profile and generates modern CV. The user is shown percentage which reflects completion of his profile. For generating, purposes were in student profile added options for new information such as computer skills, short personal info, and date of birth.

For every new option was added domain model and adjusted competent controllers and views. CVs were implemented in a way that supports translation to currently chosen language in the application. Generating also properly solves the problem with unlisted information from the user profile and keeps the nice final look at CV. For implementing this feature is used iText PDF library. It provides easy creation of a final document by adding phrases and other elements into a document object. The positioning of elements in a document is done by tables. There are created three different templates for generating CV. The user can choose between them in the new view.

New login options are another added feature. At first, it was planned to create a new login with Facebook and Google. However, a recent change in Facebook restrictions requires the application to run on HTTPS (hypertext transfer protocol secure) protocol. The iKariera project in its development phase still runs only on HTTP, so it was decided by IAESTE Slovakia that this feature will be implemented in later development process. Login in with Google remained as the goal for this phase. Log in to an external provider such as Google uses users existing account registered in provider service. After the user clicks on login button, the application sends a request to Google authorization server which will request the user to log in to his Google account.

After successful login, Google service sends special token with user information to the application. Details of this communication can be viewed in Fig. 1. Sharing this discrete information between provider and application uses some of the security protocols. In this case, is used protocol OAuth2. For implementing communication through this protocol is used Grails plugin Spring Security OAuth2.

Some methods of this plugin were rewritten for usage of correct redirects and to customize the way in which is build user account after successful login from Google. If the user logs in with Google and there is found no linked account in an application with Google account, the user is requested to create or link new account in iKariera. This way the user has two separate accounts, one in Google and one in iKariera. This will probably be reduced to just one Google account in future development. However, for now, it is sufficient this way, because it does not interfere with the way how portal account logic works.

New work offers search to filter the last implemented feature. Existing filters in original project are using full-text search provided by SQL (Structured Query Language) to search for single word attributes. This approach is not sufficient for larger text searches in many records. The new filter is searching in job offer description, requirements and other sections for a query defined by the user. For this more complex full-text search approach is used Elasticsearch technology. It is open source tool for full-text search and analysis. It is built on Apache Lucene

which gives it high performance and almost instant search results. It is one of the most popular search engines which offers various features if combined with other tools.

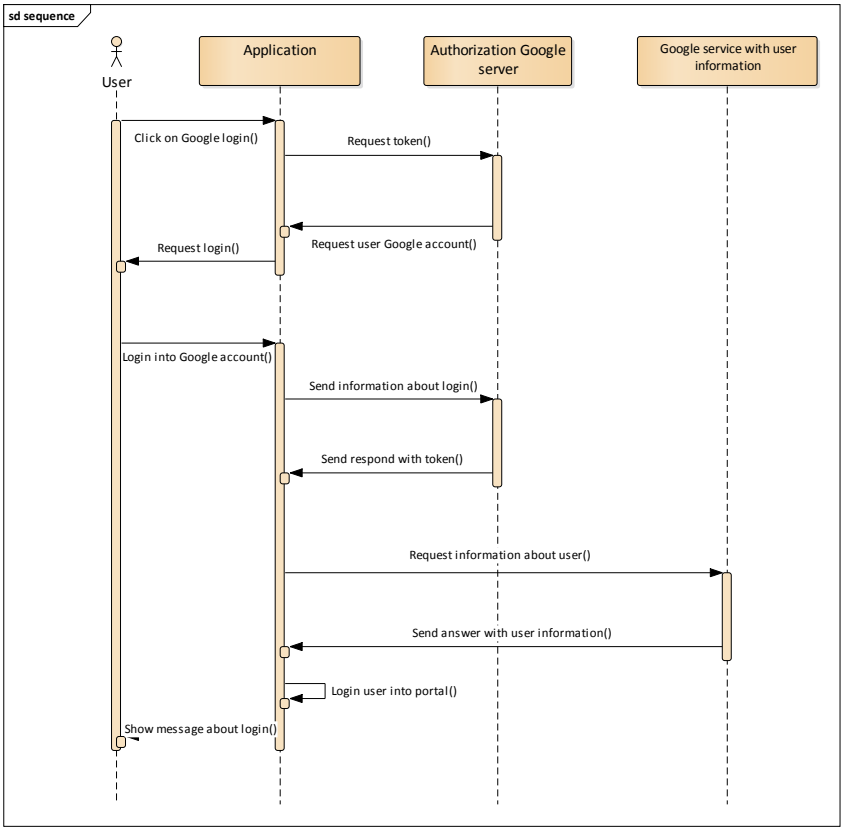

Fig. 1 Sequence diagram of the communication process of Google login

Elasticsearch uses similar principles for storing data in its indices compared to standard databases. It works over a cluster of nodes (server) which enables high-speed searches and protection against loss of data. These are the main reasons for choosing this technology. For implementation is used Grails plugin Elasticsearch 1.2.0. It does abstraction over required configurations on Elasticsearch and simplifies implementation. To enable search over specific records in domain model it is needed to mark them as searchable and give them boost. Boost gives a score to search hits in a specific record. Hits are then ordered by score and returned as a list from Elasticsearch. Other filters then filter the list if the user used them and again ordered by *Top* attribute. Offers marked as *Top* have the highest priority and the second priority is determined by resulting score from search. Example with a table of ordered job offers can be viewed in table 1. Final filtered list of job offers is then displayed to the user.

TABLE I
EXAMPLE OF ORDERED JOB OFFERS AFTER APPLYING SEARCH FILTERS

Order	Job offer	Top	Score
1.	C# developer	Yes	2.05
2.	Java developer	Yes	1.85
3.	Designer	Yes	1.13
4.	Project manager	No	2.36
5.	Service Technician	No	1.03

V. DEPLOYMENT AND TESTING

The application was after completion of new features deployed to the accessible server for testing purposes. For deployment is used the server provided by IAESTE Slovakia. This server runs on Linux with Ubuntu 14.04.5 distribution. For running application is used Apache Tomcat server. The application requires running instance of Elasticsearch. This needed to be installed first as a service running in the background. Next was application packaged in a WAR (Web Application Resource) format used for static deployment. It was then loaded to Tomcat application folder, and Tomcat server was started. Deployment details of iKariera can be noticed in Fig. 2.

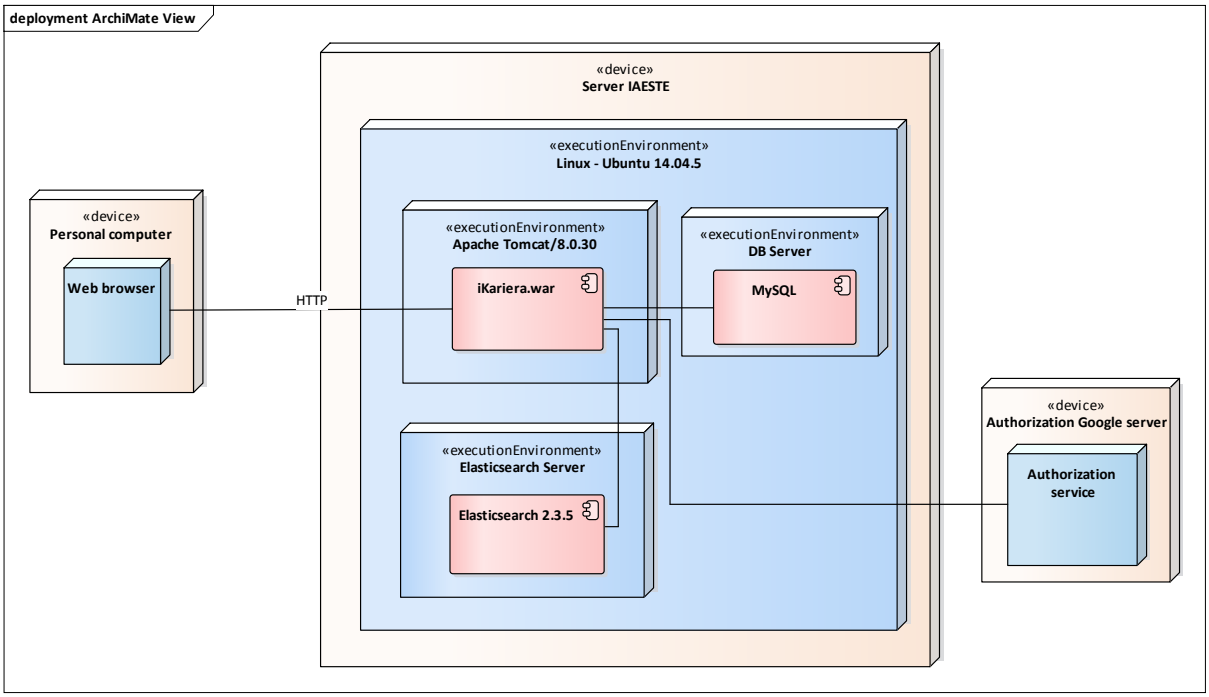

Fig. 2 Deployment diagram of iKariera

Testing of the final version of the application from this development phase was done manually using various versions of all popular web browsers and various devices. For this purpose, was used online tool Browser Sandbox from Turbo.net. Testing was focused on new implemented features and responsiveness of the application. Results showed minor problems with displaying some elements on mobile devices which was subsequently fixed by appropriate CSS rules. There was also a small problem with date input field which was not supported by Internet Explorer and older versions of Mozilla Firefox. Users with these browsers are now allowed to insert date into plain text input field. Overall the application and new features performed well. Performance tests will be used after completion of the application

VI. CONCLUSION

Portal iKariera is still in its state of developments. It is obvious that many new features still need to be implemented and many features have to be changed. During development of this work were found and fixed various bugs, missing translations and done other minor or more significant fixes. Upgrade of the used technologies proved to be a step in the right direction. Also, the addition of new technologies modernized the whole project and provided new possibilities for the future of the iKariera. The primary aim of this bachelor thesis was to

advance the development of the portal and brought it closer to its completion. It was successfully achieved as the application is now fully functional and accessible for testing purposes.

REFERENCES

[1] A. K. Kousen, “Making Java Groovy,” Manning, pp. 25–47, September 2013
[2] D. K. Abhinandan, “Gradle Essentials,” Packt Publishing, pp. 11–30, December 2015
[3] M. Scharhag, “What’s new in Grails 3,” April 2015, Available at https://www.mscharhag.com/grails/whats-new-in-grails-3
[4] C. Walls, “Spring in Action,” Manning, vol. 4, pp. 29–68, November 2014
[5] C. Walls, “Spring Boot in Action,” Manning, pp. 12–26, December 2015
[6] C. Olaru, “Grails 3 – Step By Step,” Lean Publishing, pp. 10–60, April 2017
[7] G. Smith, P. Ledbrook, “Grails in action,” Manning, pp. 16–28, April 2009
[8] P. Behl, “Migrating from Filters to Interceptors,” November 2016, Available at http://www.tothenew.com/blog/grails-3-migrating-from-filters-to-interceptors/
[9] M. Ploed, “Grails 3.x update,” Video record, Youtube, seen February 2018, available at: https://www.youtube.com/watch?v=IhehO9aM5bk&t=338s