Acquiring Android App Development Skills in a Virtual Learning Environment

Extending the Virtual Programming Lab for Moodle towards Self-Assessed App Development Tasks employing Android and Gradle

Frank Neumann [1], Juan Carlos Rodríguez-del-Pino[2], Sebastian Homer[3]

Abstract: The presented project provides a virtual learning environment for university courses teaching app development using Android and Kotlin. It features both self-guided lessons and associated tests as well as practical app development tasks and associated self-assessed programming exercises. The well-known Moodle plug-in *Virtual Programming Lab* (VPL) serves as a starting point for the latter part. VPL gets extended by means for the usage of the Gradle build tool, the Android SDK tools and AndroidX tests for unit tests at the level of Android activities and fragments. In particular, the rather excessive resource requirements of the Gradle tool created numerous problems in the sandboxes used by VPL. These issues were addressed by configuring suitable VPL settings. In addition, problems associated with the proper usage of these settings had to be addressed in VPL Altogether the proposed virtual learning environment provided a suitable means for learning in the described field as determined by questionnaires send to students.

Keywords: virtual programming lab, VPL, automated grading, automated assessment, automated evaluation, Kotlin, Android, Gradle, Android X tests

1 Introduction

Learning to develop native Android apps is an exciting and challenging endeavor for undergraduate students in computer science. It involves obtaining manifold skills ranging from the used programming language, over various aspects of Android apps and associated frameworks, to high-level architecture and programming concepts needed for such app development.

1 HTW – Hochschule für Technik und Wirtschaft Berlin, Fachbereich 2: Ingenieurwissenschaften - Technik und Leben, Wilhelminenhofstr. 75a, 12459 Berlin, Germany, frank.neumann@htw-berlin.de, https://orcid.org/0000-0002-8530-3283
2 ULPGC - University of Las Palmas de Gran Canaria, Department of Informatics and Systems, Tafira Campus, 35017, Spain, jc.rodriguezdelpino@ulpgc.es
3 HTW – Hochschule für Technik und Wirtschaft Berlin, Hochschulrechenzentrum, Ostendstraße 25, 12459 Berlin, Germany, sebastian.homer@htw-berlin.de

https://doi.org/10.30844/AKWI_2022_04

In the present case, computer science undergraduate students (fifth semester) enroll for the course with a strong background in object-oriented programming, software engineering and some foundations in formal aspects of computer science as well as UML. The learning module starts from this competence level of students and aims at providing them with the above-mentioned theoretical foundations and practical skills for app development up to an intermediate level. The course's goal consists in the development of a medium size Android app by project groups of two students. For mastering the project, the students must take into account modern paradigms of Android development as advocated by the Android team with regards to architecture, design and navigation.

Altogether, the module consists of 12 units that each take three hours (1.5 hours lecture, 1.5 hours of programming exercises and homework) and three additional sessions offering advice and consultation time for the final app project.

The course is split into four learning sections as shown in Table 1.

#	Section name	Number of units	Content
1	Learning Kotlin	5	• Type system • Function • Classes and interfaces • Collections • Lambdas • Coroutines
2	Android basics	2	• Activities and layouts • Views and view groups • Intents • Manifest
3	Android app architecture	4	• Fragments • Navigation • Data binding • MVVM • Connectivity • Asynchronous programming
4	UI aspects	1	• Styles • Themes • Material design • Menus • App bars

Table 1: Content of the Android app development module

The first section focuses on providing insights into the Kotlin programming language and its concepts. In particular, the students learn the type system, functions, classes, interfaces, collection classes, lambdas and coroutines. The subsequent block on Android basics focuses on the various components used in an Android app (activities, intents, manifest, etc.)

and layouts. The following section aims at providing architectural guidance for modern Android app development as recommended by Google's architecture guides [GO21]. The last unit focuses on UI aspects such as themes, styles and menus.

2 Related Work

The following sections analyze what kind of approaches and solutions exist for a virtual learning environment for Android app development using Kotlin. The first section studies the topic of self-guided learning courses and their associated tests for self-assessment. The second section focuses on practical app development tasks and associated self-assessed programming exercises.

The analysis will take into account video guided self-learning courses (e.g. *Udacity*, *Udemy*, *Stepik*), self-learning tutorials (e.g. *Google codelabs*), and some tools aimed at creating self-assessed programming exercises (e.g. *CodeRunner*, *VPL*, *EduTools* plugin). Special attention will be paid to the integration into Learning Management Systems (LMS) used in the university context.

2.1 Self-guided Learning Courses – Conceptual Level

Over the last decade massive open online courses (MOOC) emerged and gained attraction both in academia (e.g. *OpenHPI*[4], *Standford Online*[5]) and on commercial eLearning platforms (e.g. *Coursera*[6], *Linkedin Learning*[7], *Stepik*[8], *Udacity*[9], *Udemy*[10]). MOOCs typically combine video lectures with subsequent assessment of competences. Although many free courses are available, no free MOOCs for the discussed topic of Android app development could be identified. Many of the above-mentioned MOOCs and eLearning platforms provide the necessary tooling to author new learning courses.

Overall, MOOCs are standalone solutions that in most cases are not integrated in the LMS of universities. In the opposite direction, typical LMS (e.g. *Blackboard*, *Canvas*, *ILIAS*,

LAMS, *Moodle*) used by universities may easily serve as platforms to setup self-guided learning courses combined with automated assessments. Courses in LMS may contain a blend of videos and other learning materials as well as test activities used to assess the learning progress.

4 https://open.hpi.de
5 https://online.stanford.edu/
6 https://www.coursera.org/
7 https://www.linkedin.com/learning/topics/linkedin-learning
8 https://stepik.org
9 https://www.udacity.com/
10 https://www.udemy.com

Google Developers Codelabs[11] provide self-learning tutorials through a step-by-step experience to setup an app or to use a particular feature in an existing app. They typically give precise coding instructions along with the necessary conceptual knowledge. On the subject of Android app development with Kotlin *Codelabs* offers excellent courses. Unfortunately, *Codelabs* do not provide means for assessing the learning progress, as their main target consists in guiding students through the stepwise process of writing code for the app.

2.2 Self-learning Exercises – Programming Level

A couple of platforms developed over the last decade that support self-learning programming exercises combined with automated assessment. *Codecademy*[12] is a commercial platform offering coding classes in currently 12 programming languages at different levels of proficiency. Users of the platform write their coding solutions, which can then be evaluated. *JetBrains Academy*[13] falls also in this category but offers significant fewer subjects. This platform uses the *EduTools* plugin[14] for Android Studio, IntelliJ and other IDEs. In addition, EduTools may also be connected to other eLearning platforms like Coursera. Unfortunately, no interface connecting *EduTools* to any LMS could be identified. *freeCodeCamp*[15] focuses on providing free self-learning courses initially for web-technologies (HTML, CSS and JavaScript) but extended its scope towards subjects as Python and data science. Unfortunately, these eLearning providers do not integrate in any of the LMS used by universities.

On the university side, various add-ins may be employed to build self-learning programming exercises within LMS. For Moodle two noteworthy plug-ins were identified. The *Virtual Programming Lab* (VPL) [RRH12] [VP22] [Wa15] [Th15] [RRH11a] [RRH11b] [RRH11c] is a plug-in to the Moodle LMS, which supports more than 30 programming languages out of the box. It can be easily extended for additional programming languages by supplying adapted bash scripts. It allows teachers to author programming exercises alongside with the expected results as testcases. Students can edit, run and evaluate their solution in the commonly used Moodle environment. If desired, teachers can accept the grades determined by self-evaluation performed by the students. In addition, *VPL* provides teachers with capabilities to detect plagiarism among the handed-in solutions. *CodeRunner*

is another plug-in to Moodle, which supports about 10 programming languages but does not offer any check for plagiarism [CO22].

11 https://codelabs.developers.google.com/
12 https://www.codecademy.com/
13 https://www.jetbrains.com/academy/
14 https://github.com/JetBrains/educational-plugin
15 https://www.freecodecamp.org/

3 Problem Analysis

In the authors perception, the ideal virtual learning environment for university courses teaching app development using Android and Kotlin would allow for the authoring and usage of:

- self-guided lessons including videos and other multimedia learning materials focusing on the teaching of concepts
- associated tests assessing the reached competences of the students
- self-learning programming exercises
- automated assessment of the students' solutions handed-in for the programming-exercises

In addition, the learning environment would allow the teacher to check for plagiarism among the coding solutions. Ideally, the students would work in their usual IDE for Android app development i.e. Android Studio. In order to keep all the data for the course in one place, the results of the tests and programming exercises should be stored within the LMS. Table 2 summarizes the functional and technical requirements identified for the virtual learning environment.

ID	Group	Description
F-1	Authoring content, tests and exercises by teachers	
F-1.1		Authoring of lessons including videos and other multimedia learning materials focusing on the teaching of concepts.
F-1.2		Authoring of tests assessing the competence level of the students in the area of concepts.
F-2	Exploring content by students	
F-2.1		Explore and use course material.
F-2.2		Assessment of acquired competence level based on tests.
F-3	Authoring of programming exercises by teachers	
F-3.1		Authoring of programming exercise description.
F-3.2		Authoring of sample solution for programming exercise.
F-3.3		Coding of test cases used for the assessment of the programming exercise.

F-4	Editing, running and evaluation of coding solutions by students	
F-4.1		Editing of coding solution for programming exercise.
F-4.2		Upload of coding solution for programming exercise.
F-4.3		Running of coding solution for programming exercise.
F-4.4		Evaluation of coding solution for programming exercise.
F-5	Check for plagiarism by the teacher	Identification of similar coding solutions for programming exercises.
T-1	LMS	Moodle should be used as LMS for storing the content and the results of tests and programming exercises.

Table 2: Identified functional and technical requirements for the virtual learning environment

4 Proposed Solution

Starting from the gathered requirements and the technology landscape laid out in the previous sections, suitable technologies will be identified in a first step. Based on the selected technology stack, the desired solution will be further developed and described.

4.1 Selection of the Technology Stack

Chapter 2 described currently available technologies, platforms and tools for the realization of a virtual learning environment geared towards app development for Android and Kotlin. Those technologies will be filtered by the requirements collected in chapter 3 in order to select the best fitting technologies.

Due to the technical requirement *T-1* demanding an LMS integration, the many eLearning platforms unable to integrate with Moodle will be rejected. Moodle itself sufficiently covers the requirements groups *F-1* and *F-2*. Only *CodeRunner* and *VPL* as Moodle plug-ins will be further considered for the coverage of the requirements clusters *F-3*, *F-4* and *F-5*. Neither of both plug-ins supports developing Android apps out of the box. However, *VPL* supports the Kotlin programming language and offers plagiarism checks. In contrast, *CodeRunner* does not support Kotlin and does not provide any kind of plagiarism check. Consequently, the combination of Moodle and *VPL* covers more requirements than *CodeRunner* combined with Moodle. Neither of the two solutions supports Android app development out of the box. Finally, Moodle in combination with the *VPL* plug-in is selected as the technology stack to implement the virtual learning environment geared towards app development for Android and Kotlin.

4.2 Configuration of VPL Jail Server

Based on the chosen technology stack of Moodle and *VPL*, the first stage of implementing the desired solution will be detailed. The *VPL* jail server needs to be properly configured for building and running Android apps. In the *VPL* architecture, the jail server acts as an execution sandbox that effectively isolates code run by different users from each other. The development of Android apps requires the following parts of the Android SDK:

- *sdkmanager*: Allows for the installation of the different SDK components.
- *build-tools*: SDK component required for building Android apps.
- *platform-tools*: Contains platform tools like adb.
- *platforms*: The actual SDK packages

On the jail server, the *"/usr/lib/android-sdk"* directory is used for installing the SDK. The SDK manager[16] needs to be downloaded to this directory first. With its help the other three components will be installed. The following code lines depict the installation of the above-mentioned SDK components for the Android SDK version 30.0.3 using the *sdkmanager* tool:

```
sdkmanager "build-tools;30.0.3"

sdkmanager "platform-tools" "platforms;android-30"
```

Subsequently, experiments were conducted on how to use the Gradle build environment within a sandbox on the jail server. The Gradle tool has rather excessive resource requirements that create numerous problems in the sandboxes used by *VPL*. These issues were addressed by identifying and configuring suitable *VPL* settings. It turned out that both within the *VPL* plug-in and on the jail server the maximum used memory and the maximum file size of the execution file had to be increased to 2 GB. Here the required settings for the jail server's configuration file:

```
MAXMEMORY=2Gb

MAXFILESIZE=2Gb
```

When using those limits, a jail server of version 2.7.2 or higher is required due to issues in the XMLRPC protocol.

16 https://developer.android.com/studio/command-line/sdkmanager

Configuration of VPL Plug-in

Accordingly, the following limits have to be permitted in the configuration of the VPL plug-in:

```
Maximum memory used: 2 GB

Maximum execution file size: 2 GB

Maximum number of processes: 200
```

4.3 Authoring of VPL Activities

When creating a new VPL activity in Moodle, those settings have to be selected under *Advanced Settings -> Maximum execution resources limits*.

A general issue is the high number of files being used when developing an Android app in Android Studio. The layout of a minimal app consists of various files in a very specific structure depicted in Figure 1.

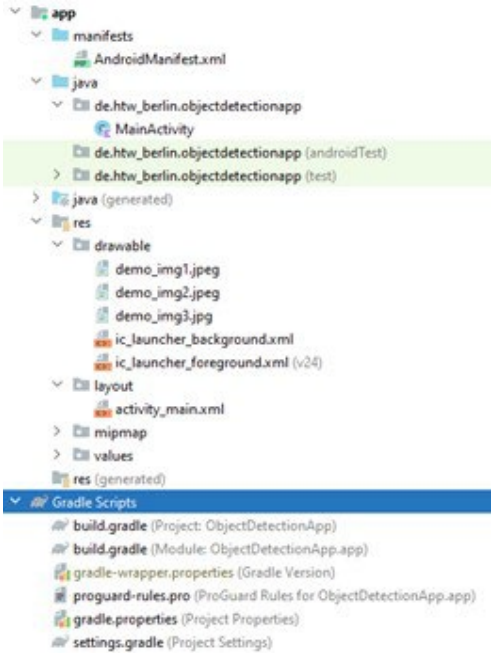


Figure 1: Structure and content of an Android app project

Besides the actual source files, it contains resource files as well as configuration files for Gradle. Drawing from previous experiences, students often fail to hand in a higher number of files. Therefore, the actual submission for the above-mentioned case will consist of only three files:

- MainActivity.kt
- activity_main.xml
- AndroidManifest.xml

Consequently, the other required files have to be provided by the setup of the *VPL activity* as shown in Figure 2. They should be added under *Advanced Settings -> Execution files.*

Figure 2: Android app files contained as VPL execution files

The Gradle version specified in *gradle-wrapper.properties* should be set to 7.3, as previous versions had memory issues in the jail server's sandbox environment.

Finally, the *vpl_run.sh* and the *vpl_evaluate.sh* files need to be adapted for running and evaluating an Android app. Here, the JDK version 15.0.2 is required for running *RoboElectric* tests at the level of activities and fragments. The difference between the run and the evaluate scripts consists only of the different build targets used for Gradle:

- Run: *compileReleaseSources*
- Evaluate: *testDebugUnitTest*

```
. common_script.sh

check_program java

check_program kotlinc

if [ "$1" == "version" ] ; then

       echo "#!/bin/bash" > vpl_execution

       echo "kotlinc -version &> .kotlinc_version" >> vpl_execution

       echo "cat .kotlinc_version | sed 's/.*kotlin/kotlin/'" >> vpl_execution

       chmod +x vpl_execution

       exit

fi

JUNIT4=/usr/share/java/junit4.jar

if [ -f $JUNIT4 ] ; then

       CLASSPATH=$CLASSPATH:$JUNIT4

fi

export CLASSPATH

# search Android SDK

export ANDROID_SDK_ROOT=/usr/lib/android-sdk

# Use JDK 15.0.2

export JAVA_HOME=/usr/lib/jvm/jdk-15.0.2
```

```
# build and test using Gradle wrapper

chmod +x ./gradlew

cat common_script.sh > vpl_execution

echo "export ANDROID_SDK_ROOT=$ANDROID_SDK_ROOT" >> vpl_execution

echo "export JAVA_HOME=$JAVA_HOME" >> vpl_execution

echo "./gradlew compileReleaseSources" >> vpl_execution

chmod +x vpl_execution
```

5 Experiences and Discussion

We designed a questionnaire containing questions about the course in general and about the design, handling and the benefits of the exercises. There were 25 students enrolled in the course. 14 of them received grades in the exercises and thus can be regarded as active participants. 10 of the students filled out the questionnaire.

For each question, we used a Likert scale with a range that goes from 1 for the lowest score (strongly disagree) to 5 for the highest (strongly agree). Table 3 contains the questions used, the mean and two distribution plots of the student's responses. The left box plot shows the locality and spread of the answers' numerical values. The median is plotted as a solid vertical line and outliers as dots. The left and right hinges represent the 1. and 3. quartile. The left whisker reaches from the hinge to the lowest value at most 1.5 * inter-quartile range, the right one to the largest value not further than 1.5 * inter-quartile range [GG22]. The right histogram shows the absolute number of each answer value in the data set.

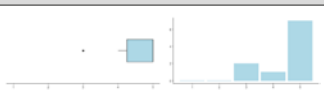
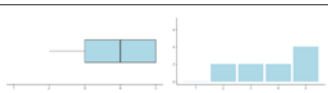
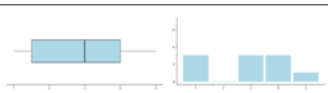
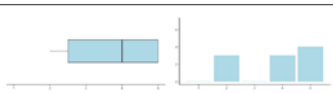
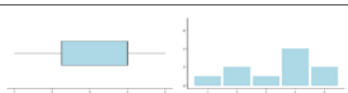
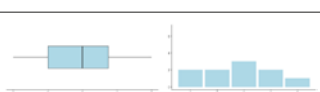
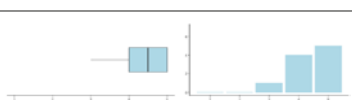
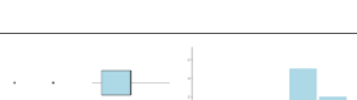
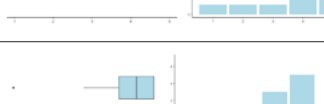
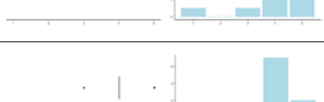
Question	Mean	Distribution plots
The programming exercises were an important supplement to the lectures.	4.5	
The goal of the tasks provided by the professor was always clear.	3.8	
I had no technical difficulties handing in and evaluating my solutions.	2.9	
The provided exercises made me think about related problems of programming.	3.8	
I was able/asked to create my own solutions for the given problem.	3.4	
I used other sources to solve the problem.	2.3	
I discussed my approaches with other students in my class.	2.8	
The programming exercises helped me to better understand the concepts of Android apps such as activities, fragments, navigation and data binding.	4.4	
I developed skills and techniques.	3.6	
The lecture and exercises increased my interest in developing Android app.	4.1	
I feel empowered to solve typical problems/issues in developing Android applications.	4.1	
The combination of lectures and practical exercises was a good preparation for the final project	4.2	

Table 3: Evaluation questions and statistics

A pleasant result of the questionnaire – the didactic goal of the course was achieved: Only one student neither agreed nor disagreed with the statement, that the exercises helped to understand the basic concepts of programming Android apps, the others agreed (4) or strongly agreed (5). 7 out of 10 students strongly agreed, that the programming exercises were an important supplement of the lectures. The majority reports an increased interest in developing Android Apps and feel empowered to solve problems or issues in that field.

3 students were faced with technical problems when handing in or evaluating their solutions, only 1 student strongly agreed, that no problems occurred. None of the students asked the technical helpdesk for assistance or failed uploading a solution.

The answers show that students did not use other sources than the provided broadly. We asked them for a list of used sources, they mentioned Google Codelabs, StackOverflow and video tutorials hosted by YouTube.

6 Conclusions

Altogether the proposed virtual learning environment provides a suitable means for learning in the described field. This statement was confirmed by the questionnaires handed in by the students. However, certain aspects of the current setup can still be improved. The answers to the question on technical difficulties when handing in and evaluating solutions suggest that especially the reliability of this aspect can be significantly increased. Another area of improvement is the long build time required by the Gradle tool. When using Gradle from within Android Studio, the long build time appears also as an issue when building a project for the first time. Afterwards, incremental builds are performed by Gradle that require significantly less time. However, in the sandbox environment of the jail server, Gradle always executes a full build. Improvements addressing this issue are under discussion at the level of *VPL* and the jail server. In addition, the *VPL* developers are studying how to incorporate the Gradle build tool, the Android SDK tools as well as the AndroidX tests so can be used in *VPL* out of the box.

On the level of the requirements gathered in chapter 3, all of them were completely covered by the selected technology stack and the implemented solution.