Bernard Jehan & Vaneck Quentin

Master 1 – Software engineering

Résumé

Ce rapport a pour but de décrire l’environnement et les tests effectués pour l’application « Timber » dans le cadre du cours de test logiciel

Timber – Lecteur musical

Projet - Test logiciel

Une image contenant signe, alimentation, horloge

Description générée automatiquement

Sommaire

[1 Introduction 2](#_Toc39424710)

[2 Description de l’environnement 3](#_Toc39424711)

[2.1 Langages 3](#_Toc39424712)

[2.2 Versions de l’application 3](#_Toc39424713)

[2.3 Environnement 3](#_Toc39424714)

[2.4 Testing 3](#_Toc39424715)

[3 Structure du code 4](#_Toc39424716)

[4 Contact avec les développeurs 5](#_Toc39424717)

[5 Techniques de test employées 6](#_Toc39424718)

[6 Tests réalisés 7](#_Toc39424719)

[7 Pistes d’amélioration 8](#_Toc39424720)

[8 Conclusion 9](#_Toc39424721)

# Introduction

Timber est une application mobile offrant une interface permettant de lire les musiques disponibles sur l’appareil sur lequel elle est installée. Il sera question, dans ce rapport, d’une deuxième version de l’application, « TimberX », étant un remaniement en Kotlin de la première application en Java.

Premièrement, Une description de l’environnement sera établie ainsi qu’une description de la structure du code et des tests déjà présents dans le projet.

Ensuite, seront décrits la prise de contact avec les développeurs, les différents échanges qui ont eu lieu avec eux et l’aide que nos tests ont pu leur apporter dans leur résolution de bugs.

Enfin, sera développé la technique de test utilisée afin de mener à bien ce projet et viendront conclure ce rapport différentes pistes d’amélioration pour un meilleur déroulement dans le développement d’un logiciel et de son testing.

# Description de l’environnement

Dans cette section sera développé les points concernant les techniques de développement utilisées pour réaliser cette application, telles que : les différents langages utilisés, la version de l’application, le support et le Framework de développement utilisé ainsi que les tactiques de testing.

## Langages

Tout d’abord, l’application est développée en Kotlin qui est un [langage de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation), open source, [orienté objet](https://fr.wikipedia.org/wiki/Programmation_orient%C3%A9e_objet) et [fonctionnel](https://fr.wikipedia.org/wiki/Programmation_fonctionnelle) développé par JetBrains. L’un des avantages de ce langage est qu’il possède un [typage statique](https://fr.wikipedia.org/wiki/Typage_statique) qui lui permet d’être compilé par la [machine virtuelle Java](https://fr.wikipedia.org/wiki/Machine_virtuelle_Java) (JVM) et vers plusieurs plateformes en natif grâce au [LLVM](https://fr.wikipedia.org/wiki/LLVM)[[1]](#footnote-2). L’objectif de ce langage est d’être interopérable, sécurisé, clair et d’avoir un support d'outillage.  
De plus, comme dit précédemment, le langage Kotlin utilise une JVM, qui est une machine virtuelle simulée par un interpréteur, afin d'exécuter les programmes compilés sous forme de bytes code Java (qui est assez léger) ; ce qui permet d'avoir un plus grand niveau d'abstraction par rapport à la machine et donc d'être très portable.

Ensuite, Kotlin est également un langage de programmation qui est de plus en plus populaire dans l’univers du développement mobile, notamment sur Android.

## Versions de l’application

Ensuite, l’application est développée et compilée avec le sdk 28 de la version 9 d'Android (Android Pie) ; ce choix constitue la version minimum sur laquelle l’application peut être exécutée. Dès lors, l’ensemble des versions d’Android supérieurs à la 9ème devrait pouvoir exécuter l’application.

## Environnement

L’application a été réalisée sur l’environnement Android Studio qui permet de développer des applications mobiles sous Android. De plus, cet environnement se base sur IntelliJ IDEA qui est dévelloper par JetBrains et utilise le moteur de production Gradle pour build les applications.

Comme dit précédemment, l’application est développée grâce au langage Kotlin qui a été créé par JetBrains, tout comme Android Studio. Dès lors, l’utilisation d’Android Studio facilite grandement la création de projet ou l'ajout de fichiers en Kotlin, étant donné que cet environnement supporte entièrement Kotlin.

## Testing

Finalement, l’application dispose de quelques tests unitaires réalisés avec JUnit qui est un Framework de tests unitaires notamment utilisé pour Java. Ces tests sont sont essentiel pour tout projet logiciel car ils permettent de comprendre les problèmes pendant le développement.

# Structure du code

Dans cette section, il est question de décrire la structure du projet et la façon dont les différentes fonctionnalités sont réparties afin de clarifier et ainsi mieux comprendre le fonctionnement de l’application.

## Structure principale

Le projet est divisé en deux packages principaux, un contenant le code de l’application et un autre contenant les fichiers de test. Le code principal est lui divisé en plusieurs packages correspondants à des fonctionnalités spécifiques comme l’accès à la base de données, à une api, ou celui concernant la vue.

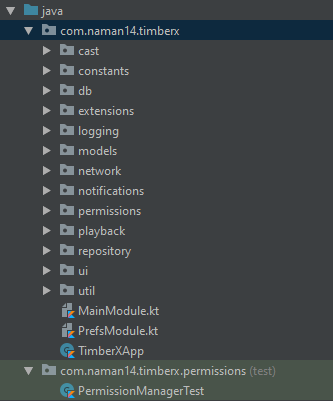


Figure 1: Structure des fichiers du projet de base

## Les tests

Il n’y a pour le moment qu’un seul fichier de tests unitaires, portant sur les permissions de stockage sur un appareil et suivant bien un schéma classique « setup, exercise, verify, tear down ».

# Contact avec les développeurs

[Demander pk TimberX et conversion Kotlin et changer raison de V2 dans l’intro]

Méthodologie particulière pour le développement

Demander Ordre d’importance des fonctionnalités à tester

Est-ce que ça les intéresse d’avoir des retours de nos tests ou qu’on push nos tests

Pourquoi n’avoir tester que les permissions de stockage sur l’appareil

Hello,

We are students at the University of Namur in Belgium and for our software test course we had to choose an application to test. As users of Timber and having found that it had only a few unit tests, we decided that our analysis would focus on this application.

From then on, we would like to ask you some questions:

Our first question is about the reason why you decided to migrate your application? Indeed, your first application, Timber, was made in Java, then you created another application, Timberx, which is developed in Kotlin, so we would like to know what motivated your decision.

Next, we would like to know if you have used a particular work’s methodology for the development of your application?

Then, we would like to know, according to you, which particular functionalities would be most important to test. Also, why did you only test the storage permissions on the device? Or did you make different kinds of other tests or on another platform? And if so, what are they?

Finally, would you be interested to get some feedback from our tests or, possibly, that we push them on your github.

Thank you in advance for your feedback,

Kind regards,

Bernard Jehan.

# Stratégie de test employée

Il est à noter qu’une bonne pratique concernant les tests est de les faire au fur et à mesure que le code est fait, et de tester soit une grande partie soit les parties les plus importantes du code. Or ici tout reste à faire ou presque, et la tâche est plus complexe encore du fait que le code ne soit pas documenté.

Plusieurs sortes de tests sont utilisées afin de mener à bien ce projet.

## Tests unitaires

## Tests d’intégrations

# Pistes d’amélioration

* Faire tests dès le début (prevent bugs)

# Conclusion

Table des figures

[Figure 1: Structure des fichiers du projet de base 4](#_Toc39424729)

1. LLVM*, Low Level Virtual Machine*, est une infrastructure de compilateur conçue pour l'optimisation du code à la compilation [↑](#footnote-ref-2)