PWSafe

Android-App zur Passwortverwaltung

Modul: Software Engineering I
Dozent: Prof. Dr. Bernhard Convent

Inhalt

1. Einleitung

Im Rahmen der Vorlesung „Software Engineering I“ der IHK Nord Westfalen im Wintersemester 2014/2015 sollte mit besonderem Blick auf dem Werkzeugeinsatz ein Programm entwickelt werden.

Die Studierenden Nadine Feldmann, Lukas Huwe, Michael Kerkhoff und Sebastian Ochtrup entschieden sich dazu ein möglichst realitätsnahes Projektthema zu wählen, welches sowohl ansprechend als auch in der Zukunft einen Mehrwert darstellen würde. Aus diesen Anforderungen entschloss sich das Team, eine Android Applikation zur zentralen und sicheren Passwort-Speicherung zu entwickeln.

1.1 Ausgangssituation

Mit der fortschreitenden Technisierung des Alltags und der Zunahme an anwendungsbasierten Services haben Benutzer immer mehr Passwörter zu verwalten. Aus Sicherheitsgründen sollten die Passwörter immer komplexer gestaltet werden. Gleichzeitig sollte optimaler Weise pro Zugang ein individuelles Passwort verwendet werden, um das Sicherheitsrisiko im Falle eines Passwortdiebstahls zu minimieren.

Durch diese Situation ergibt sich die Frage, wie man seine Passwörter für den Fall des Vergessens festhält. Die Passwörter auf Papier zu schreiben ist nur bedingt praktisch, da ein Zettel beispielsweise sehr leicht verloren gehen kann und ein unverschlüsselt aufgeschriebenes Passwort ein großes Sicherheitsrisiko darstellt. Denkbar wäre eine Anwendung, die auf dem Computer installiert wird. Da man seinen Computer aber nicht überall griffbereit hat, scheint eine Passwortverwaltung direkt auf dem tragbaren Gerät - wie einem Smartphone oder Tablet - auf dem die Passwörter auch benutzt werden, am sinnvollsten.

1.2 Projektziel

Ziel dieses Projektes ist die Entwicklung einer Android-App, die dem Nutzer das Verwalten und Nutzen seiner Passwörter erheblich erleichtern soll. Dabei wird der Fokus auf die Sicherheit der gespeicherten Passwörter gelegt: Einerseits in der Hinsicht, dass niemand anders unbefugt Zugriff auf die Passwörter erlangen soll und andererseits so, dass der Anwender möglichst komplexe und damit sichere Passwörter nutzen kann. Der Komfort der Anwendung soll dadurch erhöht werden, dass sie auf mobilen Geräten wie Smartphones und Tablets immer zur Hand ist. Des Weiteren soll der Nutzer durch eine intuitive und übersichtliche Benutzeroberfläche unterstützt werden.

1.3 Projektabgrenzung

Folgende Funktionen sind nicht Bestandteil dieses Projektes:

- Synchronisation der Passwörter über mehrere Mobilgeräte
- Anbindung der App an Clouddienste
- Speicherung von Notizen zu Passwörtern
- Backup-Funktionen der gespeicherten Passwörter

Es wird jedoch nicht ausgeschlossen, dass die App nach erfolgreichem Projektabschluss um die oben genannten Funktionen erweitert wird.

2. Planung

2.1 Geplante Programmentwicklung

Die Anwendung soll zunächst Passwortmanagement im Allgemeinen komfortabel ermöglichen. Sie soll stabil auf einem Android-Smartphone oder Tablet betrieben werden können und eine intuitive, übersichtliche Oberfläche besitzen. Die folgenden Bausteine sollen enthalten sein:

1. Das Generieren sicherer Passwörter
2. Das Ver- und Entschlüsseln der gespeicherten Passwörter
3. Das Speichern und Lesen von Passwörtern

Beim Generieren neuer Passwörter soll der Nutzer die zu verwendende Zeichenmenge (Groß- & Kleinbuchstaben, Sonderzeichen, Zahlen etc.) und die gewünschte Länge des Passworts angeben können. Des Weiteren soll es möglich sein, die Sicherheit bereits erstellter Passwörter anzeigen zu lassen. Die Bewertung soll anhand der Länge und verwendeten Zeichen etc. erfolgen und direkt visualisiert werden.

Alle Passwörter sollen verschlüsselt gespeichert werden. Die Verschlüsselung erfolgt anhand eines Master-Passworts, welches beim ersten Start der Anwendung durch den Benutzer vergeben und später beliebig angepasst werden kann. Beim Speichern eines neuen Passworteintrags wird dieses mit dem Master-Passwort verschlüsselt und abgespeichert. Bei Änderung des Master-Passworts werden alle gespeicherten Passwörter neu verschlüsselt. Das Master-Passwort wird beim Öffnen der App abgefragt. Nach der Eingabe des Benutzers wird – unabhängig von der Korrektheit der Eingabe - eine Liste der Dienste angezeigt, für die Passwörter gespeichert sind. Diese Passwörter werden mit dem eingegebenen Masterpasswort entschlüsselt, daher werden nur bei korrekter Eingabe die tatsächlichen Passwörter berechnet. In der Liste kann auf einen Eintrag getippt werden, wodurch das entschlüsselte Passwort dazu angezeigt wird. Es soll eine Möglichkeit geben, das Passwort für eine beschränkte Zeit in die Zwischenablage zu übernehmen.

Um Passwörter ändern oder löschen zu können, muss man ein Bearbeitungspasswort eingeben. Dies muss ebenfalls beim ersten Anwendungsstart festgelegt werden. Im Gegensatz zum Master-Passwort wird hier zurückgemeldet, ob die Eingabe richtig oder falsch war. Nur mit einem korrekten Bearbeitungspasswort ist ein Ändern oder Löschen möglich.

Neben durch die Anwendung generierten Passwörtern soll es außerdem möglich sein, manuell erstellte Passwörter zu speichern.

2.2 Geplanter Werkzeugeinsatz

Es sollen Elemente der in der Vorlesung behandelten Werkzeuge verwendet werden. Im Rahmen dieses Projektes fiel die Entscheidung auf die Verwendung des Versionskontrollsystems Git in Kombination mit gebührenfreien Service GitHub. Der Zugriff auf das Git-Repository erfolgte mit dem kostenfreien Tool SourceTree der Firma Atlassian. Die eigentliche Programmierung erfolgte mit dem Android Studio welches zur freien Nutzung bereit steht. Android Studio beinhaltet zudem das Buildwerkzeug Gradle als auch Unit Testing Werkzeuge.

3. Realisierung

3.1 Werkzeuge

Im Folgenden werden die eingesetzten Werkzeuge erläutert.

3.1.1 Git

Git ist eine freie Software zur verteilten Versionskontrolle von Dateien. Anders als bei dem Versionskontrollsystem Subversion (SVN) gibt es nicht zwingend einen zentralen Server. Jeder Benutzer besitzt eine lokale Kopie des gesamten Repositorys und ist somit gleichzeitig Client und Server. Dieses Repository beinhaltet für alle Dateien des Projekts die komplette Versionsgeschichte. Ein Wechsel auf ältere Entwicklungsstände ist somit jederzeit ohne Server-Synchronisation möglich. Es werden alle Dateien des Projekts synchronisiert, außer solche, die in einem sogenannten *gitignore-File* definiert sind. Durch die lokale Speicherung des gesamten Repositorys gewinnt der Anwender an Schnelligkeit. Gerade bei größeren Projekten müssen so nicht viele Dateien über das Netzwerk transferiert werden. Git ist ein Kommandozeilentool, welches diverse Parameter bietet. Es ist auf fast allen unixartigen Systemen unter anderem auf Linux, Mac OSx als auch FreeBSD lauffähig. Eine native Windows-Integration gibt es nicht, weshalb Tools von Drittherstellern verwendet werden müssen. Git ermöglicht es Branches zur isolierten Entwicklung einzelner Funktionen zu bilden. Diese können anschließend zusammengeführt (gemerged) werden.

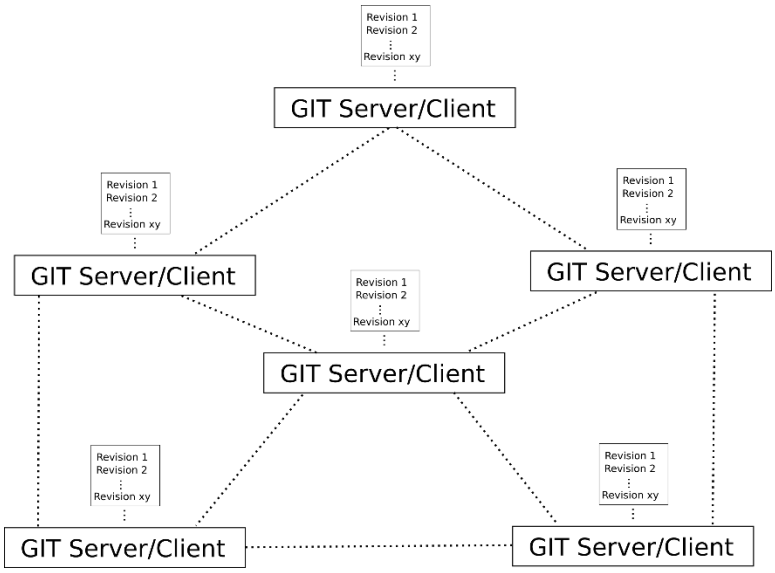

Abbildung 1: Systemstruktur

Basierend auf Git erschien 2008 der Dienst GitHub. GitHub ist webbasiert und in der Grundversion kostenfrei zugänglich. Bestehende Projekte sind für die Öffentlichkeit einsehbar und können bei Bedarf kopiert und angepasst werden. Gegen die Bezahlung von monatlichen Endgeldern können private Repositories erworben werden. Diese sind nur mit Berechtigungen einsehbar. GitHub ist in der Open-Source-Software-Entwicklung sehr populär. So werden unter anderem für die Entwicklung des Linux-Kernel, von PHP als auch von JUnit GitHub-Repositories genutzt. Im April 2011 befanden sich 2 Millionen Repositories auf GitHub.

3.1.2 SourceTree

SourceTree ist ein kostenloses Tool der Firma Atlassian. Es erweitert die Kommandozeilenfunktion des Versionskontrollsystem Git um eine einfach zu bedienende grafische Oberfläche. SourceTree arbeitet problemlos mit GitHub zusammen und bietet verschiedene Komfortfunktionen wie das Aufzeigen der bisherigen Änderungen als auch die verschiedenen Branches. Es können mehrere voneinander unabhängige Repositories gleichzeitig eingebunden werden.

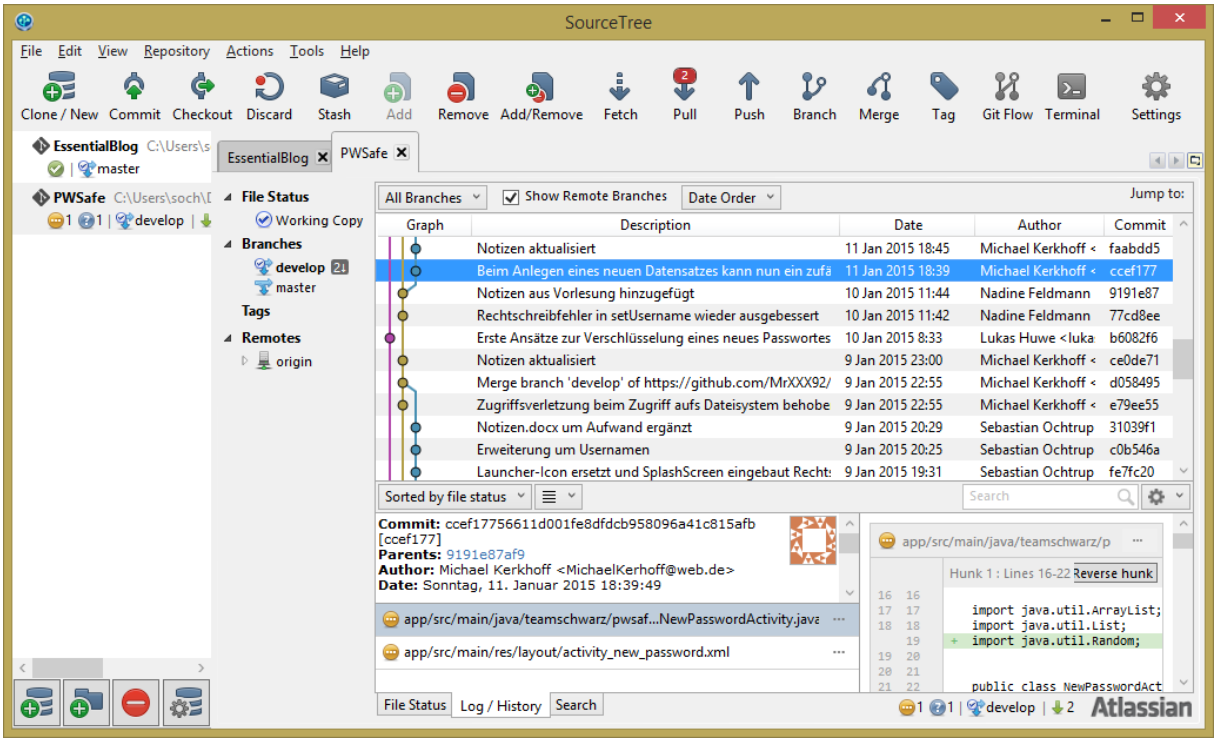

Abbildung 2: SourceTree

Kommentare zu den einzelnen Commits werden übersichtlich als Description angezeigt. Die erfolgten Commits werden Dateiweise im unteren Bildschirmbereich angezeigt und können so von allen Projektbeteiligten schnell und verständlich eingesehen werden. Durch Doppelklicks auf die einzelnen Entwicklungsstände kann bequem und schnell zu einem vorherigen/ neueren Entwicklungsstand gesprungen werden. In Zusammenarbeit mit GitHub werden Änderungen zeitnah visualisiert. Liegen noch nicht committete Änderungen vor, können diese mit einem Wechsel auf den obersten Punkt „Uncommitted changes“ eingesehen und in das zentrale GitHub-Repository gepushed werden.

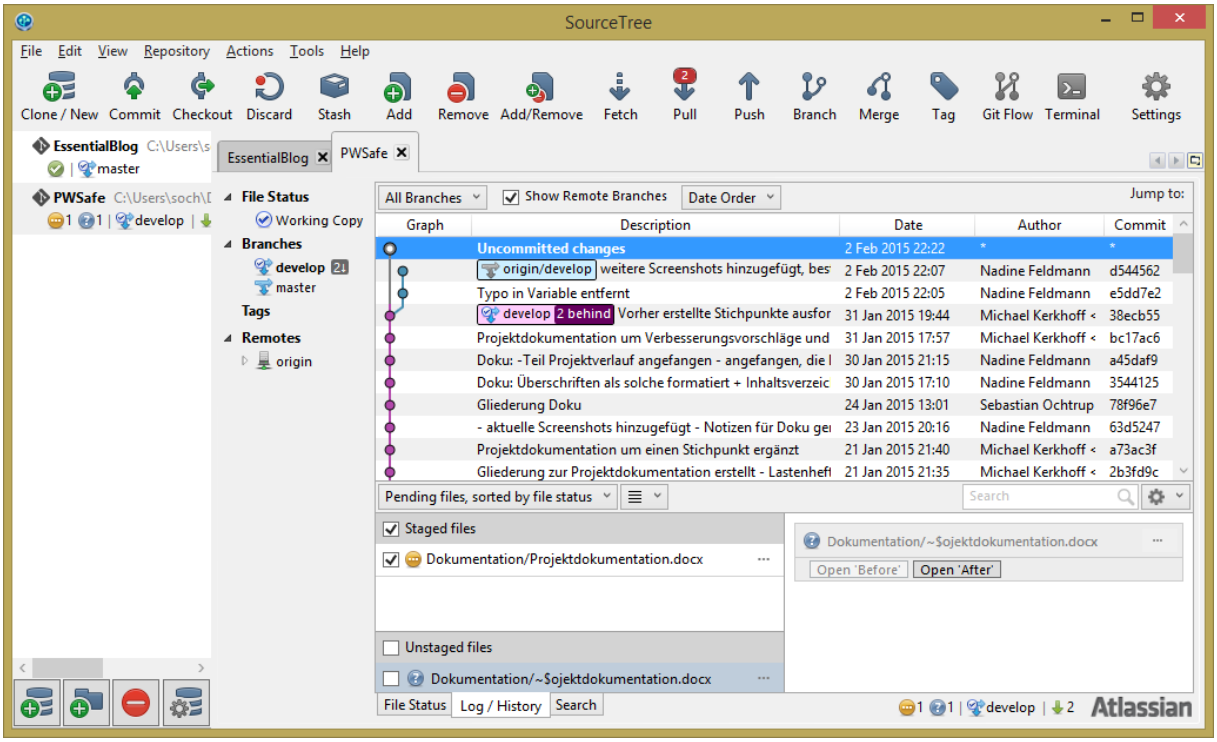

Abbildung 3: SourceTree Pushmöglichkeit und Änderungshistorie

3.1.3 Android Studio

Das Android Studio wird kostenfrei als offizielle IDE von Google zur Programmierung von Android Applikationen bereitgestellt. Basierend auf der IntelliJ IDEA, wurde das Studio erweitert und im Mai 2013 in der Beta-Version veröffentlicht. Die Beta-Phase wurde mit der Version 1.0 im Dezember 2014 beendet und aktuell liegt das Studio in der Version 1.0.2 bereit, welche auch noch im Dezember 2014 erschien.

Abbildung 4: Startbildschirm

Dadurch, dass das Android Studio auf IntelliJ basiert, gehören Features wie das Refactoring von Codeelementen, eine Unterstützung von Ant, JUnit und ein grafischer GUI-Editor zum

Funktionsumfang. Des Weiteren liefert IntelliJ ebenfalls Tools zur Versionsverwaltung und verschiedene Möglichkeiten zur automatischen Code-Generierung.

Google hat den Funktionsumfang deutlich für die Entwicklung von Applikationen für das eigene Betriebssystem erweitert. So gehört das Build-Tool Gradle mit zum festen Bestandteil der Software. Außerdem wurde der GUI-Editor angepasst. So werden verschiedene Geräte-, wie auch Plattformtypen angezeigt, die das Gestalten erleichtern sollen.

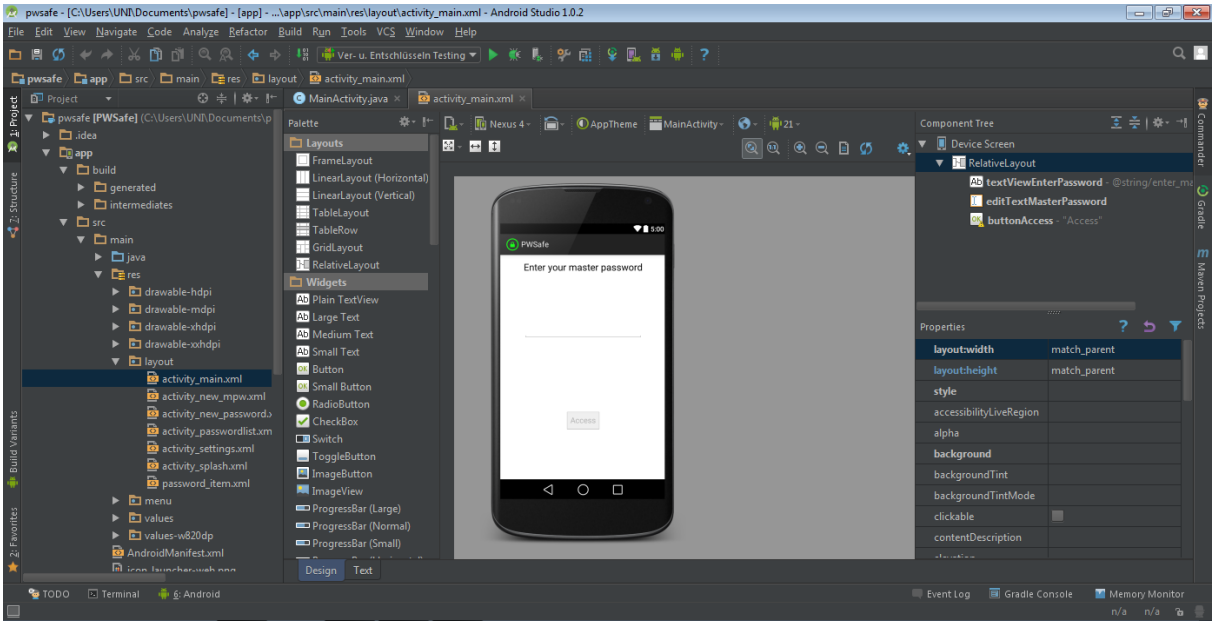

Abbildung 5: Erweiterter GUI-Editor

Google Dienste, wie das Google Cloud Messaging, wurden in die IDE integriert. So können diese individuell vom Entwickler konfiguriert und genutzt werden.

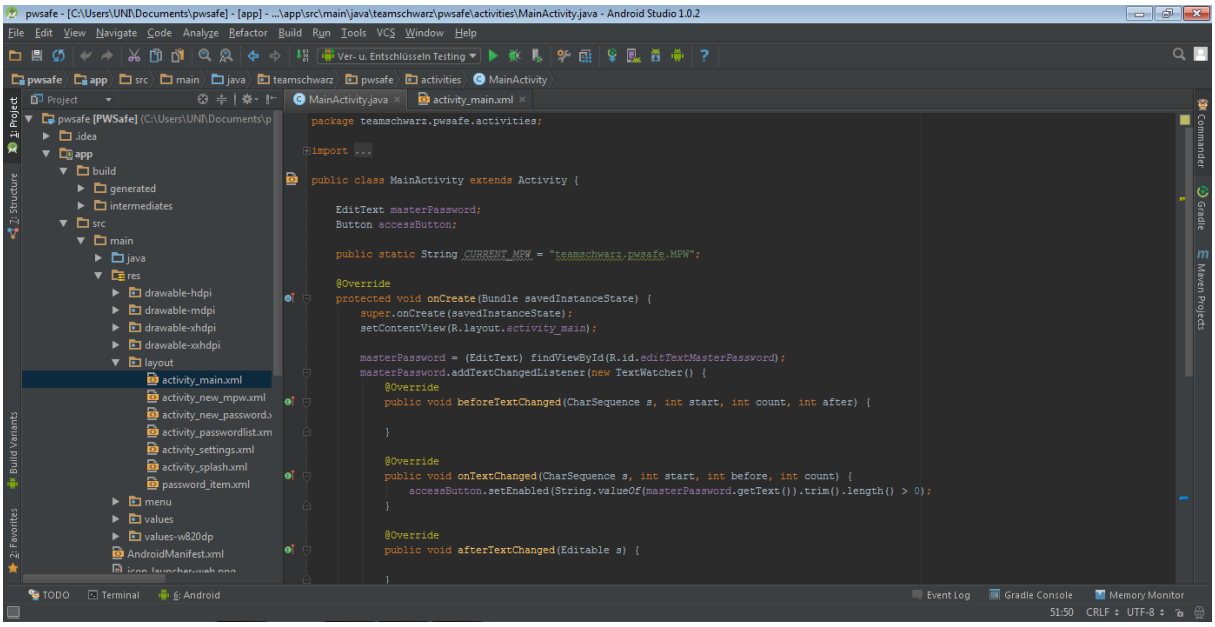

Abbildung 6: Codeentwicklung

Bei der Codeentwicklung kommt die IntelliJ IDEA zum Einsatz. Der Quellcode wird unter bestimmten Bedingungen farblich hinterlegt und beim Tippen werden passende Vorschläge gemacht. Die Navigation erfolgt anhand der Package-Struktur an der linken Seiten.

Da sich die gesamte Projektgruppe bereits mit dem Tool Eclipse auskannte und die Gruppe gerne etwas Neues ausprobieren wollte, wurde sich gegen eine Eclipse-Version mit Android-Plugin und für das Android Studio entschieden.

Zur Entwicklung von Applikationen benötigt man zudem das Android SDK, welches entsprechende Bibliotheken der einzelnen Androidversionen mitbringt. Diese können individuell über einen SDK-Manager herunterladen werden.

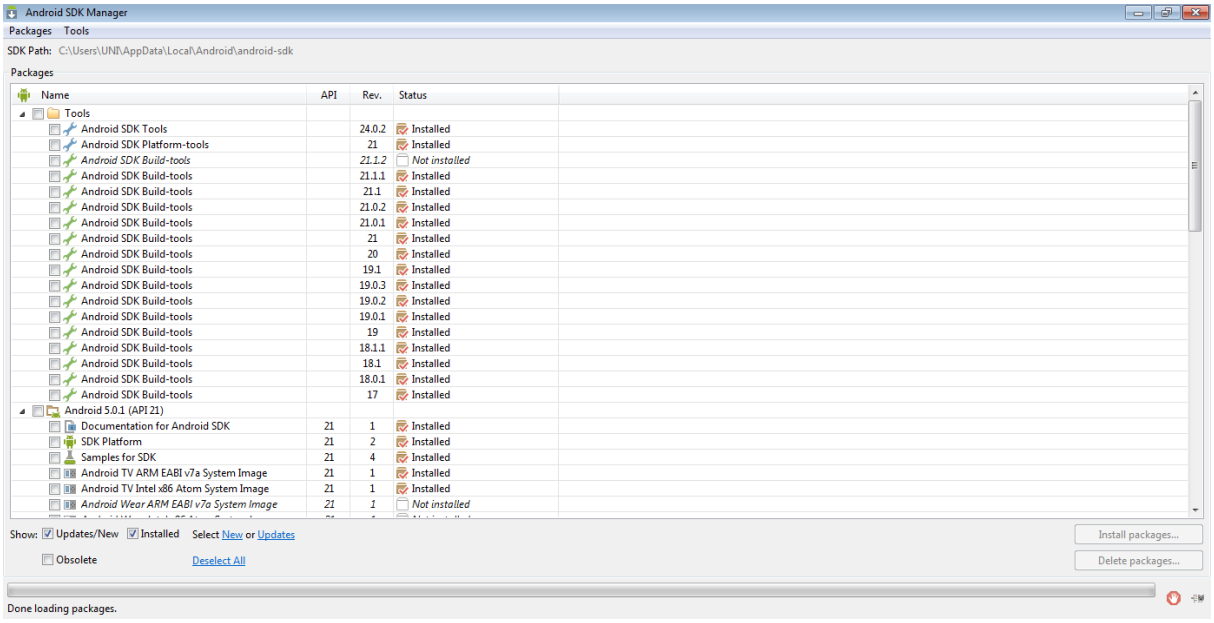

Abbildung 7: SDK-Manager

Zur Verwaltung und Konfiguration der Emulatoren gibt es einen AVD-Manager (Android Virtual Device), welcher einfache Bereitstellungsmöglichkeiten für individuelle Emulatoren bietet. Hierbei gibt es sowohl vorgegebene Emulatoren als auch umfangreiche Konfigurationsmöglichkeiten.

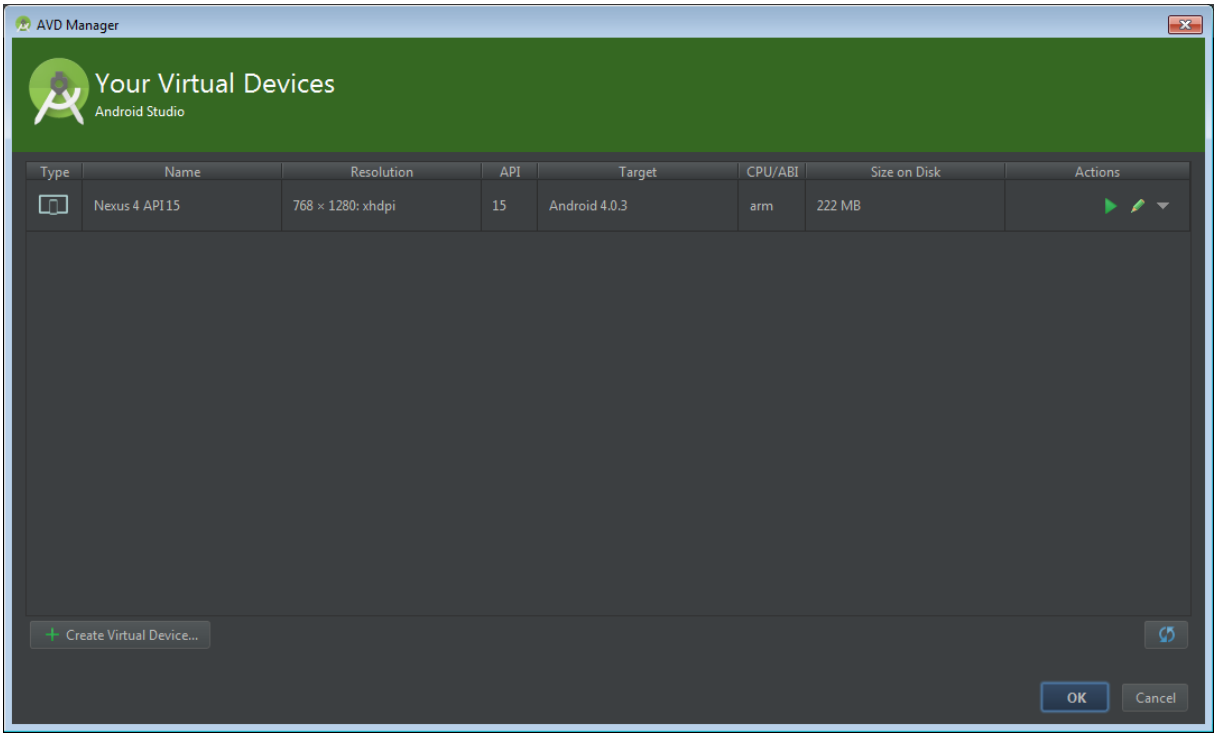

Abbildung 8: AVD-Manager

3.1.4 Gradle

Gradle ist ein automatisches Build-Tool, ähnlichen den in der Vorlesung behandelten Tools Apache Ant und Apache Maven. Gradle-Skripte sind, im Gegensatz zu Maven, direkt ausführbarer Code.

Abbildung 9: Logo

Damit Gradle bei umfangreichen Builds performant arbeitet, unterstützt das Tool sowohl ein inkrementelles, als auch paralleles Vorgehen. Bei inkrementellen Builds, werden nur Teile der Software neu gebaut. Dies sind vor allem Teile, die verändert wurden oder Teile die von geänderten Dateien abhängen. Beim parallelen Vorgehen werden verschiedene Aufgaben, die parallel ausführbar sind, auf verschiedene Prozessorkerne verteilt. Somit können beispielsweise die Tests parallel ablaufen.

Der Buildvorgang besteht aus zwei Hauptphasen. Die erste Phase, die Konfiguration, durchläuft die gesamte Build-Definition und es wird ein Abhängigkeitsgraph erzeugt. Dieser Graph enthält alle abzuarbeitenden Schritte des Builds. In der zweiten Phase, der Ausführung, wird der Graph anhand der Führung durchlaufen und alle Build-Tasks ausgeführt.

Als Tasks stellt Gradle die üblichen Phasen zur Verfügung. Dazu gehören das Validieren, das Kompilieren, die Ausführung der Tests, der Archiv-Erstellung mit Reporting und die Verteilung von gebauten Projekten.

Als Build-Datei dient Gradle die build.gradle-Datei, welche ein Groovy-Skript darstellt. In der Datei werden alle Tasks und Abhängigkeiten des Projektes festgelegt. Außerdem kann Vererbung eingesetzt werden.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 15
    buildToolsVersion "20.0.0"

    defaultConfig {
        applicationId "teamschwarz.pwsafe"
        minSdkVersion 15
        targetSdkVersion 20
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile "com.android.support:support-v4:20.0.0"

}
```

Abbildung 10: build.gradle aus dem Projekt

3.1.5 Unit Testing

Es gibt eine Vielzahl an Erweiterungen für das automatisierte Testen, doch mit dem Android-Studio werden diese nicht benötigt, da es diese Funktionalität bereits mitbringt. Um diese nutzen zu können, sind lediglich einzelne Konfigurationsschritte zu erledigen.

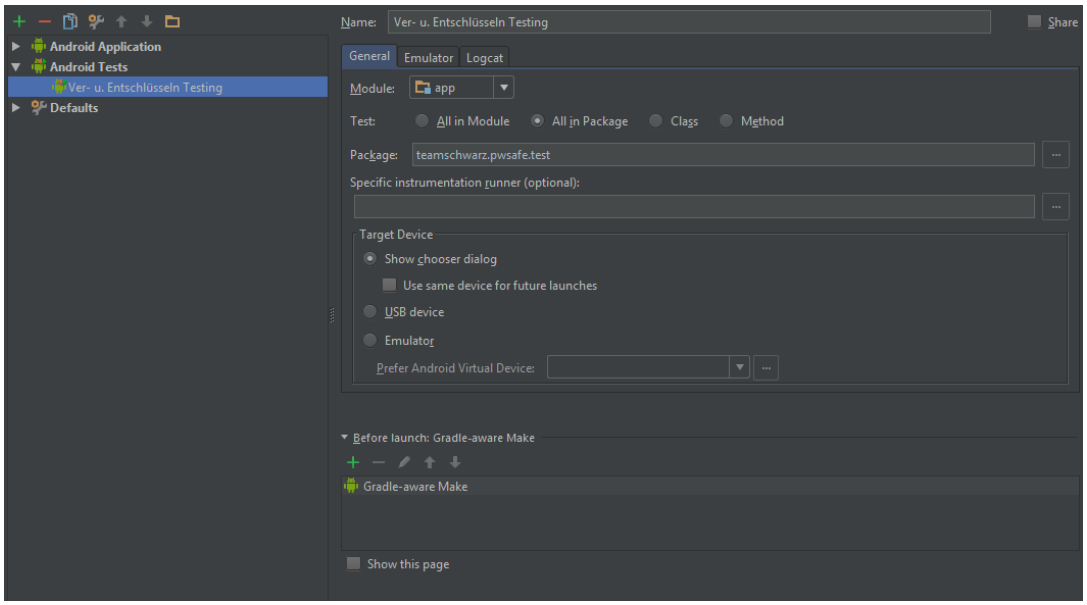

Abbildung 11: Konfigurierter Android Test

Das Android SDK liefert ein eigenes Testpackage, welches die Klasse *InstrumentalTestCase* zur Verfügung stellt. Diese erbt von der Klasse *TestCase* aus JUnit und dient als Oberklasse für alle Testklassen innerhalb des Projektes.

Testmethoden, welche ausgeführt werden sollen, beginnen mit dem Prefix „test“ und haben keinen Rückgabewert.

```
package teamschwarz.pwsafe.test;

import android.test.InstrumentationTestCase;

import teamschwarz.pwsafe.utils.Vigenere;

/**
 * Created by Lukas on 19.01.2015.
 */
public class PasswordTest extends InstrumentationTestCase{

    public void testVerfahrenKorrekt(){
        String original = "Hallo Welt, ich werde verschluesselt.";
        String passwort = "TestPassword";

        String verschluesselt = Vigenere.verschluesseln(original.toCharArray(), passwort.toCharArray());

        String entschluesselt = Vigenere.entschluesseln(verschluesselt.toCharArray(), passwort.toCharArray());

        assertTrue("Sind gleich", entschluesselt.equals(original));
    }

    public void testVerfahrenMitZweiSchluesseln(){
        String original = "Hallo Welt, ich werde verschluesselt.";
        String passwortVer = "TestPassword";
        String passwortEnt = "FalschesPassword";

        String verschluesselt = Vigenere.verschluesseln(original.toCharArray(), passwortVer.toCharArray());

        String entschluesselt = Vigenere.entschluesseln(verschluesselt.toCharArray(), passwortEnt.toCharArray());
```

Abbildung 12: Quellcode der Tests

Unit- oder auch Modultests werden im Android Studio mit Hilfe des Emulators durchgeführt. Dieser muss am Anfang einmal gestartet werden. Dies kann unter Umständen etwas länger dauern und verzögert den Testprozess.

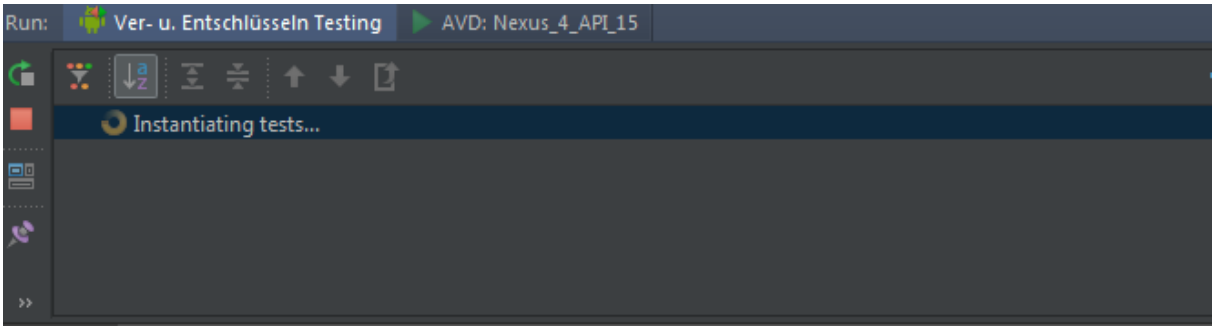

Abbildung 13: Initialisierung der Tests beim Laden des Emulators

Nachdem die Anwendung auf dem Emulator geladen ist, werden die Testfälle der Reihe nach durchlaufen und grafisch dargestellt. In dem unten aufgeführten Beispiel laufen alle Testfälle korrekt durch. Dies wird durch die grüne Farbe dargestellt.

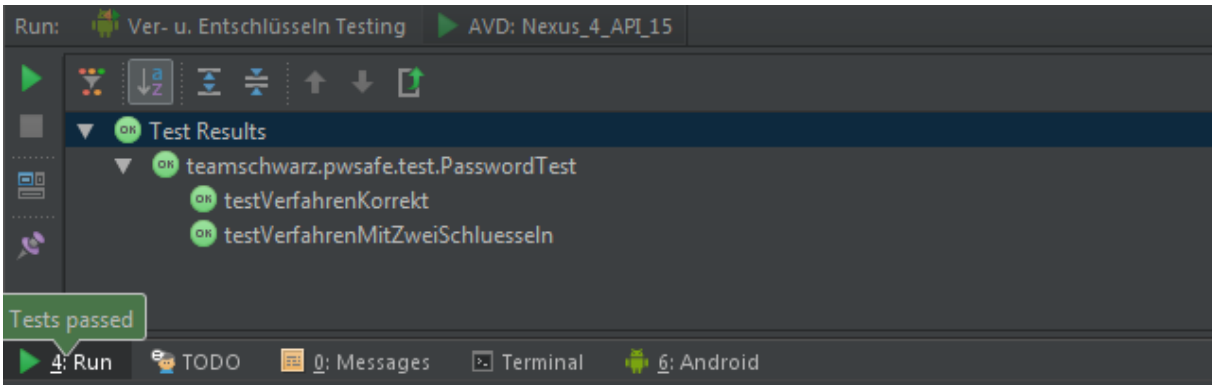

Abbildung 14: Testergebnisse

3.2 Projektverlauf

Die erste Besprechung bezüglich des Projekts fand am 10. Oktober 2014 in der Vorlesung Software Engineering statt. Die Gruppenkonstellation war schnell gefunden, da sie sich schon in anderen Arbeiten bewährt hatte. Als Thema wurde aus Ermangelung von kreativen Alternativen zunächst eine Anwendung zum Benchmarking von Datenbanken gewählt. Da die Gruppe davon jedoch nicht gänzlich überzeugt war, wurde noch nach einem anderen Thema gesucht. Es kam schließlich der Vorschlag, eine Android App zur Passwortverwaltung nach Vorlage eines bestehenden Produkts zu entwickeln. Dieser Vorschlag wurde von den anderen Gruppenmitgliedern angenommen und am 26. Oktober wurde die Entscheidung durch Prof. Dr. Convent bestätigt.

Bei der nächsten Absprache in der Vorlesung am 14. November 2014 wurden die Entwicklungsumgebung sowie die sonstigen benötigten Tools bei allen installiert und eingerichtet, dabei wurde sich gegenseitig unterstützt. Dann wurde die Absprache getroffen, sich bis zur nächsten Vorlesung etwas in die Android-Entwicklung einzulesen und sich ein wenig mit den Tools vertraut zu machen. Außerdem wurde ein grober Entwurf der App erstellt. Dieser umfasste drei Masken: Eine zur Eingabe des Masterpassworts, eine zur Anzeige der gespeicherten Passwörter und eine zum Hinzufügen neuer Passwörter. Dieser Grobentwurf sollte zu ersten Erfahrungen mit der App-Entwicklung und einer Diskussion über das Design führen. Passwörter wurden noch nicht auf dem Smartphone gespeichert und es wurde auch noch keinerlei Verschlüsselung verwendet. Das eingegebene Masterpasswort wurde ebenfalls nicht weiter benutzt.

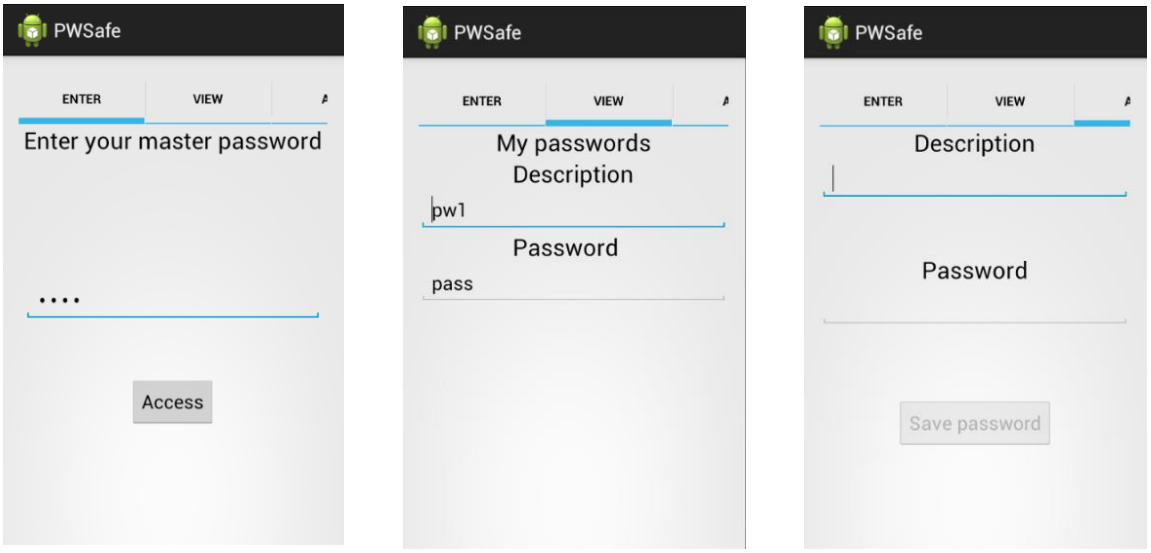

Abbildung 15: Erster Entwurf

Wie man auf Abbildung 15 sieht, ist das Design noch nicht bildschirmgerecht. Außerdem traten bei der Ausführung noch Exceptions auf, weil Elemente im Formular nicht gefunden werden konnten.

Am 29. November erfolgte die nächste Absprache im Rahmen der Vorlesung. Hier wurde der Entwurf gemeinsam betrachtet und es wurden Überlegungen für ein Re-Design angestellt. Der Entwurf sah drei Haupt-Masken vor: Eine für die Liste der gespeicherten Passwörter, eine zur Bearbeitung und Neuanlage von Passwörtern und eine „Settings“-Maske mit einem Button zur Änderung des Masterpassworts. Wie in der folgenden Skizze angedeutet ist, soll die Detail-Seite über den „+“-Button sowie über das Tippen auf ein Listenelement erreichbar sein. Außerdem soll sich beim Betätigen des Buttons „Change Masterpassword“ ein Popup zur Eingabe öffnen.

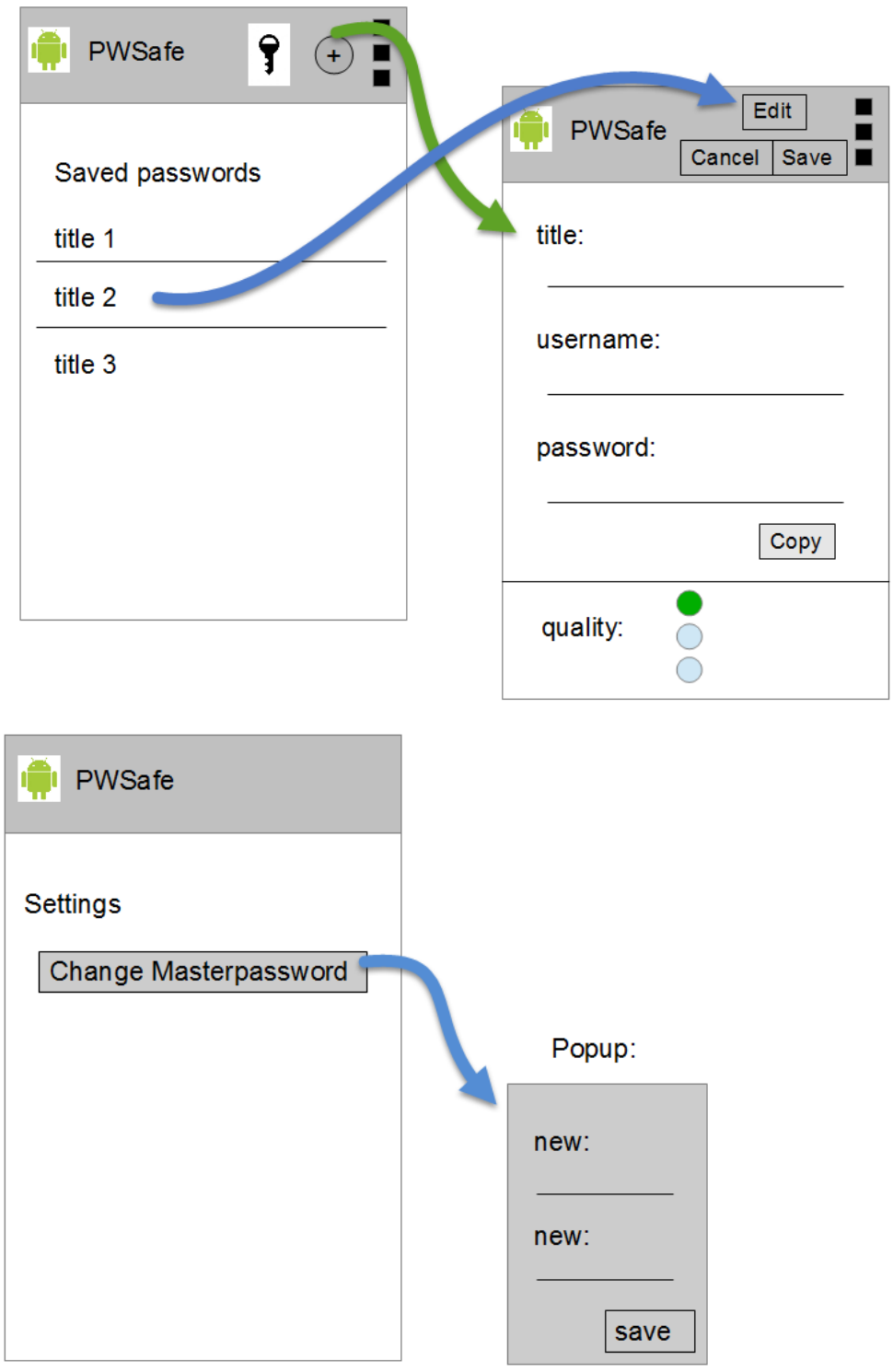

Abbildung 16: Entwurf für Re-Design

Als nächstes wurden die angesprochenen Exceptions beseitigt und die Namensgebung der Elemente wurde optimiert. Weiterhin wurden die Abhängigkeiten der Elemente untereinander verändert. Das Ziel einer ersten lauffähigen Entwurfs-Version wurde damit erreicht.

Am 13. Dezember 2014 fand eine erneute Besprechung im Rahmen der Vorlesung statt. Zwischen diesem Termin und dem neuen Jahr gab es keine Aktivitäten im Projekt, diese „Weihnachtspause“ war vorher so abgestimmt worden.

Am 02. Januar 2015 ging es weiter: Da der Code für den ersten Entwurf von einer Person erstellt wurde, mussten sich die anderen in diesen einlesen, um ihn nachvollziehen und erweitern zu können. Bis zum 07. Januar wurden das Re-Design durchgeführt: Die drei Tabs wurden aufgelöst und die bisherige Codestruktur wurde überarbeitet. Der Code war bisher vollständig in einer „Aktivität“ (eine Art Klasse für eine Maske in Android) gehalten. Dieser wurde nun auf mehrere Aktivitäten, eine pro Maske, aufgeteilt.

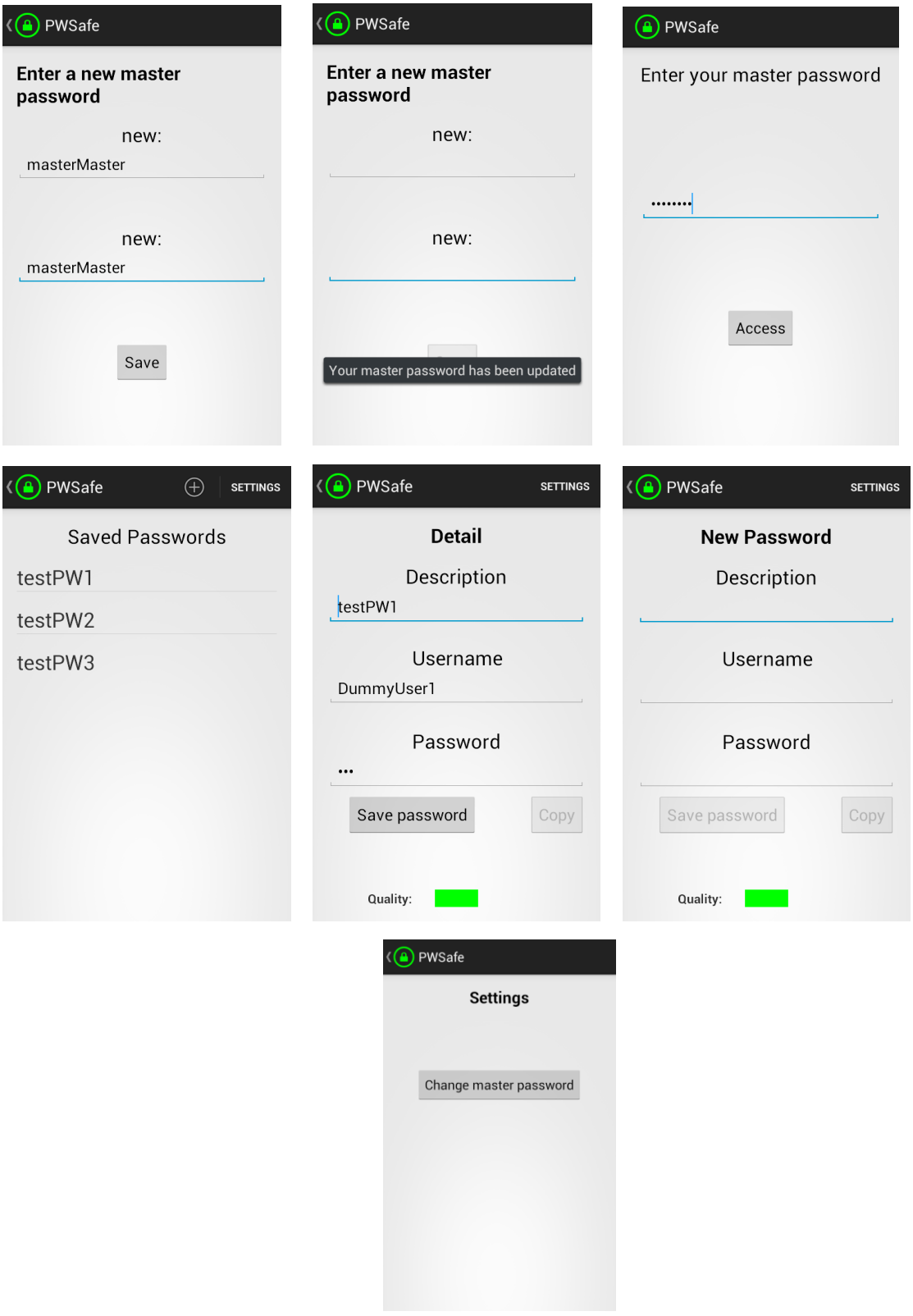

Abbildung 17: Screenshots der Version

Das neue Design hat wie vorher skizziert eine Start-Maske mit einem Eingabe-Feld für das Masterpasswort. Mit einem Klick auf den Button kommt man nun auf die Listenansicht der gespeicherten Passwörter und über den „+“-Button in der oberen Leiste lassen sich neue Passwörter anlegen. Es wurde außerdem ein „Settings“-Button implementiert, der ebenfalls im Menü der oberen Leiste enthalten ist und auf eine Maske zur Änderung des Masterpassworts führt. Der Button zum Kopieren in die Zwischenablage sowie die Ampel zur Anzeige der Passwortqualität wurden (noch funktionslos) angelegt. Die bisherige Liste der Passwörter wurde insofern erweitert bzw. geändert, dass nun nur noch die Beschreibung in der Liste angezeigt wird und man mit einem Klick auf einen Eintrag auf die Detail-Maske mit den Daten des Passworts gelangt.

Dabei traten einige Schwierigkeiten auf. Beispielsweise sollte oben links auf jeder Maske ein typischer „Up“-Button implementiert werden, der auf die letzte Maske zurück navigiert. Dies hat zunächst nicht so recht funktioniert, konnte am Folgetag aber gelöst werden, indem die korrekte Support-Bibliothek eingebunden und die Button-Funktion an der richtigen Stelle umgesetzt wurde. Eine weitere Hürde war der sogenannte „Overflow“ in der ActionBar (obere Leiste), der nicht funktionierte. Der Overflow besteht aus einer Schaltfläche mit drei Punkten, die in der ActionBar angezeigt werden. Tippt man auf die Schaltfläche, bekommt man eine Liste mit ActionButtons (Bezeichnung für die Buttons der ActionBar), für die in der ActionBar nicht genug Platz ist. Man kann normalerweise für jeden ActionButton festlegen, ob er immer, nie oder nur bei ausreichendem Platz in der ActionBar angezeigt werden sollen. Der „Settings“-Button war hier so geplant, dass er immer im Overflow angezeigt wird, das hat aber nicht funktioniert. Der „Settings“-Button wird nun also in der ActionBar angezeigt, solange genug Platz ist, was auch kein wirkliches Manko ist.

Ein weiteres anfängliches Problem war, dass der „Ok“-Button in der Tastatur bei Eingabefeldern fehlte. Dadurch konnte man die Tastatur im Emulator nicht mehr verlassen (auf dem Smartphone kann man die „Zurück“-Taste des Gerätes dafür nutzen). Dieses Problem konnte gelöst werden, indem an den Eingabefeldern ein „input type“ gesetzt wurde.

Am 07. Januar hat ein Gruppenmitglied ein angebotenes Update für das Android Studio durchgeführt, was zu Komplikationen führte: Durch das Update war das Projekt mit der bisherigen Gradle Version nicht mehr lauffähig und beim Build der App wurden das Android Support Repository sowie die Android Support Library nicht mehr gefunden. Dafür mussten einige Konfigurationen angepasst werden, außerdem mussten die weiteren Gruppenmitglieder das Update dann nachziehen, damit alle daran weiterentwickeln konnten.

Am 09. Januar wurde außerdem der „SplashScreen“, also der Startbildschirm, der kurz beim Öffnen der App erscheint, erstellt, sowie das Launcher-Icon ausgetauscht. Bisher hatte die App nur das Standard Android-Icon, dies wurde nun durch ein grünes Schloss ersetzt.

Abbildung 18: SplashSreen

Als nächstes wurde die Anwendung so erweitert, dass zu jedem Passwort neben der Beschreibung nun auch ein Benutzername eingegeben werden kann. Außerdem wurde nun mit der Speicherlogik angefangen: Ab diesem Zeitpunkt wurden beim Betätigen des "Access"-Buttons auf der ersten Maske die zuvor gespeicherten Daten aus einer XML-Datei ausgelesen, die beim Hinzufügen eines neuen Datensatzes aktualisiert wurde. Dabei kam es zunächst zu Zugriffsverletzungen, die das Schreiben der XML-Datei verhinderte. Der zuvor konfigurierte Android Emulator hatte keinen externen Speicher, obwohl dies so definiert war. Als Ursache konnte durch eine Forenseite[1] ein Bug im Android Studio ermittelt werden. Dort ist auch beschrieben, dass man eine Konfigurationsdatei zusätzlich ändern und in dieser die virtuelle SD-Karte angeben muss.

Am 10. Januar erfolgte die nächste Absprache im Rahmen der Vorlesung in Bocholt. Hier wurden die noch umzusetzenden Erweiterungen geplant und unter den Gruppenmitgliedern aufgeteilt. Die geplanten Erweiterungen umfassten die tatsächliche Nutzung des eingegebenen Masterpassworts. Dazu musste die Verschlüsselung der Passwörter sowie das Ändern des Masterpassworts umgesetzt werden. Außerdem wurde geplant, Unit-Tests für die Verschlüsselungslogik zu erstellen. Für die Detail-Maske von Passwörtern wurden ebenfalls Änderungen vorgesehen: Die angefangene „Qualitäts-Ampel“ sollte wieder entfernt werden, da sie nicht mehr wichtig erschien. Außerdem sollte der „Copy to Clipboard“-Button mit Funktion versehen und ein Button zur Generierung zufälliger Passwörter erstellt werden. Des Weiteren sollte das eigentliche Passwort von nun an bei Öffnen der Detail-Maske nicht lesbar sein, ein zu implementierender „Show“-Button sollte die Zeichen sichtbar machen können. Als Deadline für all diese Punkte wurde der 17. Januar definiert.

Für die Generierung eines zufälligen Passworts wurde eine parametrisierbare Methode implementiert: Es sollte aus den Einstellungen ausgelesen werden, ob Buchstaben und/ oder Ziffern und/oder Sonderzeichen verwendet werden sollen, sowie die gewünschte Länge der generierten Zeichenkette. Diese Parametrisierung wurde allerdings später wieder entfernt, wie weiter unten noch beschrieben wird.

Um die Passwortverschlüsselung umzusetzen, musste zunächst eine passende Verschlüsselungsmethode gefunden werden. Anfangs wurde das „Cipher“-Package aus der Java API ins Auge gefasst. Darin enthalten sind einige Methoden um gängige Verschlüsselungsverfahren, wie AES (Advanced Encryption Standard), DES oder RSA zu nutzen. Im Zuge des Projektes entschied man sich anfänglich für die AES-Verschlüsselung. Damit ergab sich die Herausforderung, dass der Key sehr lang sein musste. Erste Idee war es, das eingegebene Masterpasswort zu verlängern, indem man es wiederholt, was auch funktionierte. Danach war die Segmentierung der Passwörter ein Problem. Ein Passwort wird zur Verschlüsselung in Segmente unterteilt, die eine bestimmte Länge haben müssen. Da eine schnelle Lösung zur Festlegung der Paketgrößen fehlte, wurde die AES-Verschlüsselung verworfen. Schlussendlich fiel die Wahl auf die simplere Vigenère-Verschlüsselung, da diese einfacher zu programmieren war. Die nächste Hürde bestand darin, dass die verschlüsselten Passwörter in die XML-Datei geschrieben werden mussten und der Zeichensatz vorgab, wie die Buchstaben verschoben wurden. Dabei kam es zu einer Verschiebung auf Grundlage der ASCII Codes, wodurch die Zeichenketten hin und her geparst werden mussten. Weiterhin funktionierte die UTF-8-Kodierung nicht vollständig, daher wurde die vom Android Studio vorgeschlagene Kodierung gewählt.

Anschließend wurde sich mit dem Schreiben von Testfällen auseinandergesetzt. Schließlich wurden zwei Testfälle für die Verschlüsselungsmethoden implementiert: Im ersten wird eine Zeichenkette ver- und wieder entschlüsselt und es wird sichergestellt, dass die

[1] Stackoverflow: http://stackoverflow.com/questions/27120754/sd-card-created-in-avd-shows-as-removed-in-emulator-for-android-studio

entschlüsselte Zeichenkette wieder dem Original entspricht. Dies stellt also sicher, dass ein verschlüsselt gespeichertes Passwort wieder korrekt ausgegeben werden kann. Im zweiten Testfall wird eine Zeichenkette mit einem Schlüssel verschlüsselt und einem anderen wieder entschlüsselt. Dann wird geprüft, ob die entschlüsselte Zeichenkette ungleich der anfangs verschlüsselten ist. Damit wird sichergestellt, dass Passwörter nur mit dem korrekten Masterpasswort wieder ausgelesen werden können. Beide Testfälle konnten erfolgreich ausgeführt werden.

Als nächstes wurde noch die Struktur der Pakete im Projekt verändert. Die Klassen, die zuvor alle im Haupt Java-Paket lagen, wurden sinnvoll auf die Pakete *activities* (für die verschiedenen Masken), *test* (für die Testklasse) und *utils* (für die Passwort-Klasse, den SplashScreen und die Klassen zur Verschlüsselung) aufgeteilt. Dies ist auf Abbildung 19 zu erkennen.

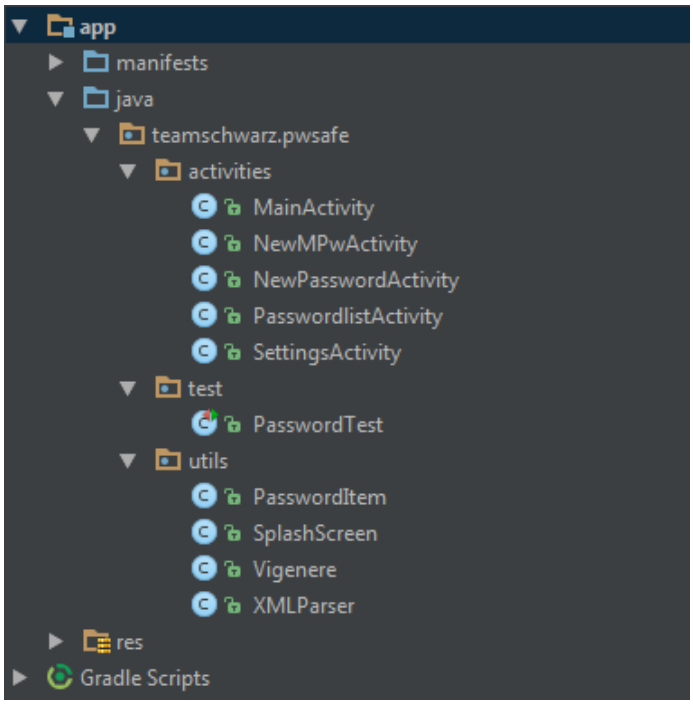

Abbildung 19: Neue Paket-Struktur

Am 17. Januar fand das nächste Treffen zur Vorlesung statt und die aktuelle Lage wurde besprochen. Jeder hat von seiner Umsetzung berichtet und bis auf die Änderung des Masterpassworts und die Unit-Tests wurden alle Funktionen umgesetzt.

Dann wurde erneut geklärt, was im nächsten Schritt, der letzten Stufe, noch umzusetzen war. Das Ändern des Masterpassworts, das Löschen von gespeicherten Passwörtern sowie die Umsetzung der Unit-Tests sollten noch implementiert werden. Daher wurden diese Aufgaben besprochen und auf die Mitglieder aufgeteilt. Daneben sollten sich alle Gruppenmitglieder Notizen zu allen Punkten der bisher groben Gliederung der Dokumentation machen, sodass die Erfahrungen aller möglichst leicht in die Dokumentation einfließen konnten. Als Deadline hierfür wurde der 24. Januar definiert.

Das Ergebnis dieser letzten Stufe wird folgend beschrieben. Bei Eingabe eines neuen Masterpassworts werden die Daten der XML-Datei nun mit dem alten Masterpasswort ausgelesen, mit dem neuen verschlüsselt und wieder in die XML-Datei geschrieben. Außerdem wurden die Buttons auf der Detail-Maske umsortiert, da sie nicht auf allen Testgeräten korrekt angezeigt wurden.

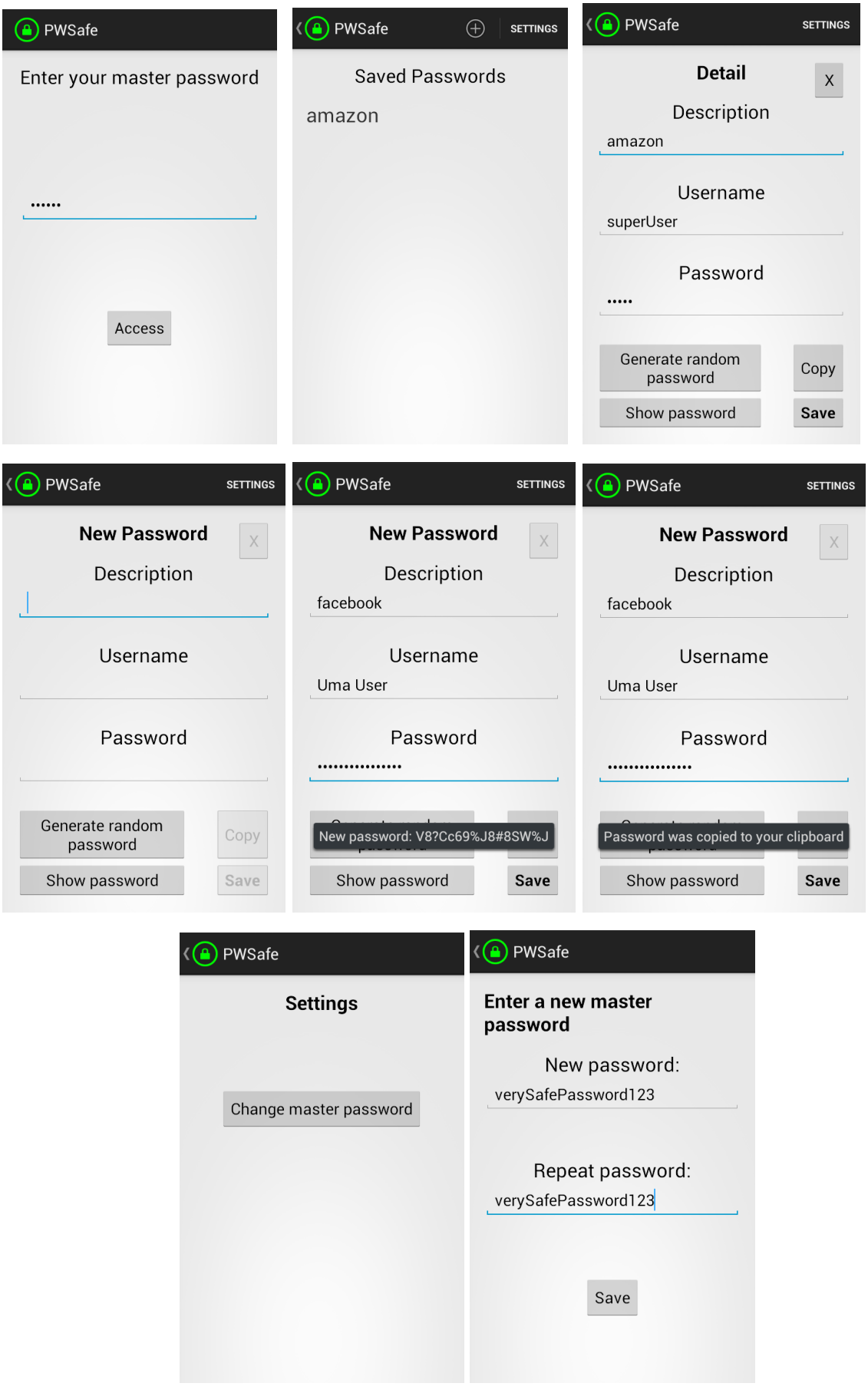

Abbildung 20: Ansichten der Endversion

Insgesamt wurde also ein inkrementelles Vorgehen zur Entwicklung der App gewählt. Die Meilensteine wurden dabei grob nach den Veranstaltungen in Software Engineering ausgerichtet.

Vergleicht man das Projektergebnis mit den anfangs definierten Anforderungen, lässt sich feststellen, dass einige Punkte nicht umgesetzt wurden. Beispielsweise ist es bei der Generierung von Passwörtern derzeitig nicht möglich, die zu verwendende Zeichenmenge und die Länge des Passworts anzugeben. Dies wurde nicht umgesetzt, da die Daten zur Generierung von Passwörtern über die Laufzeit der App hinaus in eine zusätzliche Datei mit Einstellungen geschrieben und hieraus wieder gelesen werden müssten und dies zu umfangreich geworden wäre. Da es neben diesen Einstellungen keine weiteren gab wurde eine feste Zeichenmenge mit Klein-, Großbuchstaben, Sonderzeichen und Zahlen gewählt und die Länge des Passworts fest auf 16 gesetzt. Des Weiteren wurde die Ampel zur Anzeige der Passwortsicherheit nicht implementiert. Dies wäre lediglich bei manuell erstellten Passwörtern sinnvoll, wenn ein möglicher Angreifer weiß, dass ein bestimmtes Passwort manuell erstellt wurde. In diesem Fall ist ein Brute-Force Angriff oder sogar ein Wörterbuchangriff durch Erraten des Masterpassworts möglich. Wenn das zuvor manuell erstellte Passwort nicht kryptisch ist und einen Sinn ergibt, ist mit hoher Wahrscheinlichkeit das Masterpasswort gefunden und somit sind alle anderen Passwörter auch unsicher. Ein direkter Angriff auf das Masterpasswort bringt nichts, da ein Angreifer keine Rückmeldung zur Korrektheit des Passworts erhält. Das beschriebene Szenario stellt allerdings eine Randsituation dar, weshalb dieses Feature nicht implementiert wurde.

Eine andere nicht umgesetzte Anforderung ist, dass in die Zwischenablage übernommene Passwörter nach einer bestimmten Zeit wieder daraus gelöscht werden sollten. Da dies nicht zu den Hauptfunktionen der App gehört, wurde dies niedrig priorisiert und somit nicht mehr umgesetzt.

Des Weiteren wurde im Lastenheft definiert, dass man zum Ändern und Löschen der Passwörter ein Bearbeitungspasswort eingeben müsste. Dies wurde nicht umgesetzt, da entschieden wurde, dass das Merken eines zweiten Passworts für den User möglicherweise als zu lästig empfunden würde.

Außerdem sei erwähnt, dass die App für die Verwendung auf einem Smartphone optimiert ist. Da der Fokus auf Mobilität gelegt wurde und Smartphones häufiger mobil zur Hand sind als Tablets, wurde die Verwendung auf Tablets weniger betrachtet.

4. Bewertung

4.1 Bewertung der Werkzeuge

4.1.1 Git

Drei der vier Projektbeteiligten haben im Vorfeld noch nicht mit Git gearbeitet. Da allerdings die gesamte Gruppe Subversion kannte, haben wir uns schnell für Git entschieden, um die Möglichkeit nutzen zu können etwas Neues kennen zu lernen. Da niemand aus der Gruppe einen extra Server für die Versionsverwaltung installieren konnte und die Einarbeitung in das Thema zu aufwändig gewesen wäre, sind wir sehr schnell auf den kostenlosen Dienst GitHub gestoßen. Somit haben wir bereits am Anfang der Vorlesung ein Repository erstellt und in den Grundzügen verwaltet. Sehr praktisch war dabei, dass GitHub für bestimmte Arten an Projekten bereits *gitignore-Files* liefert, sodass für wir uns darum auch nicht mehr kümmern mussten.

Da wir nur wenige größere Funktionen separat entwickelten, wurde in der Regel jeder Commit der Projektmitglieder direkt gepusht.

4.1.2 SourceTree

Das kostenlose Tool SourceTree eignete sich für unsere Entwicklung sehr gut. Es unterstützt den normalen Ablauf des Git-Servers und bot ein paar Zusatzfunktionen. So gefiel die grafische Darstellung des Projektverlaufes sehr gut und man konnte immer sehr gut nachvollziehen, wann wer welche Änderungen vorgenommen hatte. Die Nutzung war von Anfang an sehr intuitiv und wenn man die Struktur von Git verstanden hatte, sehr schnell anzuwenden.

Leider fehlt SourceTree eine Merging-Funktion, so war es teilweise etwas kompliziert verschiedene Stände zusammen zu führen. Da wir allerdings nur auf einem Branch entwickelten und die Programmierung sehr wenige Überschneidungen bot, gab es nicht viele Konflikte.

Es wurden nicht alle Funktionen des Programms genutzt. GitFlow beispielweise wurde anfangs kurz eingeführt, stellte sich aber in Verbindung mit der Einführung von Git als zu komplex für die Gruppe heraus.

Ein weiterer Kritikpunkt ist, dass das Tool nicht zu jedem Zeitpunkt mitbekommt, wann eine neue Version zum Pull bereit steht. So kam es vor, dass man auf einem älteren Stand entwickelte und erst im Nachhinein beim Pushen angezeigt bekam, dass doch eine neuere Version auf dem zentralen Server bereit stünde. Das bedeutete wieder etwas aufwändiges Lösen der Konflikte.

4.1.3 Android Studio

Das Android Studio ist für die Entwicklung von Android Applikationen sehr gut geeignet. Es bietet eine Menge Funktionen und durch den Einsatz des Emulators ist es für das Debugging noch nicht einmal notwendig ein eigenes Android-Gerät zu besitzen. Die Integration von Gradle verlief soweit sehr gut. Von Anfang an war ein entsprechendes Build-File vordefiniert und dem Projekt beigefügt, sodass wir uns um die Konfiguration anfänglich nicht kümmern mussten. Als jedoch ein Android Studio-Update bereitstand, welches über eine Benachrichtigung auf der Oberfläche installiert werden konnte, wurde auch, ohne Hinweis,

die Gradle-Version aktualisiert. Somit kam es zu Compilefehlern innerhalb der Builddatei. Erst durch längere Recherche im Internet fanden wir heraus, dass die neue Version ein paar wenige Aufrufe nicht mehr kannte, bzw. diese umbenannt worden waren. Leider war diese Version auch nicht abwärtskompatibel und so mussten letztendlich alle Projektmitglieder die Version vom Android Studio aktualisieren. Dies war bei der Entwicklung der App sehr zeitraubend.

Die Erstellung der Benutzeroberfläche gestaltete sich als kompliziert. Um Elemente richtig auszurichten, werden diese an andere Elemente, wie zum Beispiel Textboxen, oder auch an den Rand angeheftet und der Abstand definiert. Hinzu kommt die Komplexität innerhalb der verschiedensten Smartphone-Modelle, die Android als Betriebssystem verwenden. Somit müsste man auf sehr vielen Emulatoren testen, sodass überall das User-Interface korrekt dargestellt wird. Diese Komplexität konnte geschmälert werden, indem sich auf ein paar wenige Modelle beschränkt wurde. So wurden auch Tablets als Zielgerät ausgeschlossen.

Die Oberfläche des Android Studios bietet einen schnellen Zugriff auf die wichtigsten Elemente und der Codeeditor hilft einem sehr gut bei der Entwicklung von Applikationen.

Leider ist der Emulator sehr langsam und somit ist das Programmieren ohne ein separates Android-Gerät zu nutzen zwar grundsätzlich möglich, aber doch sehr langsam. Außerdem muss man den Emulator sehr gut konfigurieren. Somit dauerte es etwas, bis wir die Funktion fanden, damit die Tastatureingabe auch im Emulator genutzt werden kann. Am Anfang war das Testen nur über die Maus möglich. Ein weiteres Hindernis war es, dass es teilweise zu einem Fehler in der Konfiguration kam. Auf der Oberfläche des AVD-Managers wurde angezeigt, dass eine SD-Karte konfiguriert sei, doch dies wurde nicht in die Konfigurationsdateien des Emulators geschrieben. Im Bezug zu der SD-Karte war es sehr schwierig zu testen, denn leider hat man über den Emulator keinen Zugriff auf die Karte und kann somit auch nicht die geschriebene XML-Datei einsehen und prüfen. Doch da einige Gruppenmitglieder über entsprechende Smartphones verfügen, konnte auch diese Funktionalität getestet werden. Das Debugging und Testen über eine extra Hardware ist deutlich performanter und erleichtert die Arbeit erheblich.

4.1.4 Gradle

Gradle funktionierte von Anfang an. Lediglich die im letzten Abschnitt beschriebenen Probleme im Zusammenspiel mit dem durchgeführten Android Studio-Update verzögerten die Arbeit erheblich.

4.1.5 Unit Testing

Die Modultests boten sich in dem Projekt nur selten an, da das meiste Oberflächennavigation darstellt. Doch die implementierten Modultests waren sehr schnell konfiguriert. Die Unterstützung des Android Studios ist sehr gut und übersichtlich. Lediglich die Notwendigkeit des Emulators verlangsamt den Testdurchlauf sehr stark und mindert die Motivation, weitere Tests zu schreiben.

4.2 Bewertung des Projektverlaufs

Übergehend zur Bewertung des Projektverlaufs lassen sich anfänglich folgende Besonderheiten im Projekt ausmachen.

Da bisher niemand aus unserem Projektteam Erfahrung in der App Entwicklung sammeln konnte, war die komplette Thematik für uns Neuland. Zu Anfang war bereits abzusehen, dass dies für uns zusätzlichen Einarbeitungsaufwand bedeutete. Dem gegenüber stand jedoch die Chance eine bis dato völlig neue Technologie kennenzulernen. Eine weitere Besonderheit waren die unterschiedlichen Kompetenzen im Team. Das Team bestand aus drei Anwendungsentwicklern und einem Systemintegrator. Jedoch konnten die Aufgaben trotzdem zu gleichen Teilen, mit Berücksichtigung der verschiedenen Kompetenzen, auf die Teammitglieder verteilt werden.

Dies führt direkt zu den Dingen, die das Projekt auszeichneten. Besonders hervorzuheben ist das inkrementelle Vorgehen. Dies war besonders in unserer Situation geeignet, da es anfänglich schwer war, die Komplexität unseres Vorhabens in einen konkreten Aufwand zu überführen. Die Komplexität des Projekts wurde dabei maßgeblich von dem uns unbekannten Gebiet der App Entwicklung bestimmt. Auch positiv hervorzuheben sind die regelmäßigen Absprachen, die meist während der Veranstaltungen in Software Engineering stattfanden. Diese Absprachen umfassten sowohl den derzeitigen Status des Projekts, wie auch die weitere Vorgehensweise bis zum nächsten Treffen. Hier wurde auch besprochen, wer welche Aufgaben übernimmt. Da die Aufgabenzuteilung unter Berücksichtigung der Kompetenzen jedes einzelnen Teammitglieds erfolgte konnte jeder seinen Teil zum Gesamtvorhaben beitragen. Dabei lässt sich im Nachhinein feststellen, dass die Aufgabenverteilung mit Fortschreiten des Projekts immer klarer wurde. Dies ist, wie auch viele andere Entwicklungen auf den stetigen Erfahrungs- und Wissensgewinn im Bereich der App Entwicklung zurückzuführen. Trotz des zunehmenden Erfahrungsgewinns ließ es sich nicht vermeiden, Fehler zu machen. Diese Fehler manifestierten sich bei uns als Bugs, die wir durch zwischenzeitliche Tests entdeckten und so frühzeitig beseitigen konnten. Somit stellten diese Tests sicher, dass die zuvor umgesetzten Funktionen, wie auch die neu hinzugekommenen Funktionen zu jeder Zeit korrekt funktionierten. Durch die klare Aufgabenteilung und rege Kommunikation zur Arbeit am Code konnten wir außerdem erreichen, dass wenig Mergekonflikte entstanden. Ein letzter Punkt, der als besonders positiv hervorgehoben werden kann, ist die klare Definition von Deadlines. Die von uns gesteckten Deadlines wurden ausnahmslos eingehalten. Dies ermöglichte es uns regelmäßig und vor allem termingerecht Absprachen zum weiteren Vorgehen zu führen.

Allerdings tauchten, wie nicht anders zu erwarten, einige Schwierigkeiten auf, auf die folgend eingegangen wird. Mit dem Beginn der Veranstaltung Software Engineering wurden wir damit konfrontiert, uns ein Projekt einfallen zu lassen, welches wir mit Hilfe ausgewählter Werkzeuge umsetzen sollten. Dies erwies sich auf Grund der kurzen Bedenkzeit als sehr schwer. Nach einigem Hin und Her entschlossen wir uns schließlich für die Entwicklung eines Passwortmanagers für Android. Nachdem klar war, was Gegenstand unseres Projekts sein sollte, schrieben wir ein Lastenheft in dem wir die anfänglichen Anforderungen an die App festhielten. Jedoch verging hierauf folgend einige Zeit, da niemand wusste, wie er dieses komplett fremde Gebiet erschießen sollte. Nachdem Michael eine erste Version der App erstellt hatte kam jedoch auch die eigentliche Implementierung in Fahrt. So haben wir uns beginnend mit Hilfe von Tutorials und Foren einen Einblick verschafft. Durch den gleichzeitigen Versuch das Gelernte umzusetzen, stiegen neben unserem Wissen in diesem Bereich auch die praktischen Erfahrungen. Hierbei mangelte es beinahe nie an der nötigen Kommunikation im Team. Lediglich in einem Fall wurde das Verwenden der Android Support Library nicht explizit kommuniziert, was bei einigen Beteiligten für Verwirrung sorgte. Trotz des stetigen Erfahrungsanstiegs bereitete uns die Gestaltung eines einheitlichen Layouts für

die App bis zum Schluss Probleme. Dies ist zurückzuführen auf die unterschiedlichen Bildschirmgrößen der Testgeräte. Aus diesem Grund haben wir uns entschlossen, die Entwicklung der App auf die Nutzung mit einem Smartphone auszurichten, da Tablets nur weitere Komplexität bringen würden.

Auf Grundlage des nun abgeschlossenen Projekts und insbesondere mit Blick auf die Schwierigkeiten lassen sich einige Verbesserungen für das nächste Projekt in diesem Umfeld erkennen. Dabei ist es von entscheidender Wichtigkeit sich rechtzeitig in das Thema einzuarbeiten. Da die App Entwicklung für uns komplettes Neuland war, wäre dies besonders zuträglich zum Projektfortschritt gewesen. Neben dem frühzeitigen Einarbeiten sollte in der Entwicklungsphase darauf geachtet werden die Ziele anhand fachlicher Anforderungen zu definieren. Die sich hieraus ergebenden Aufgaben sollten sinnvoll unter den Projektbeteiligten aufgeteilt werden. Dies führt dazu, dass weniger technische Abhängigkeiten entstehen. Auch hat dies den Vorteil, dass der Blick für die fachlichen Anforderungen nicht verloren geht.

4.3 Fazit

Abschließend können wir auf ein erfolgreiches Projekt zurückblicken. Mit der Erstellung dieser App konnten wir uns einerseits einen Einblick in die Entwicklung von Android Apps verschaffen. Andererseits ist das Ergebnis dieses Projekts eine lauffähige App, die sich mit ihrem alltagsbezogenen Hintergrund nun auch nutzen lässt.

Die im Projekt eingesetzten Werkzeuge haben die Entwicklung gut unterstützt, das Kennenlernen dieser Werkzeuge war zudem interessant. Das gewonnene Wissen wird sicher auch in Zukunft von Nutzen sein.

5. Ausblick

Die App kann im aktuellen Status bereits genutzt werden, wird aber nicht frei zugänglich sein. Dazu fehlte uns vor allem die Zeit zur Qualitätskontrolle. Die nicht umgesetzten Punkte können in Zukunft noch programmiert werden. Ein weiterer Ergänzungspunkt wäre die Implementierung einer neuen Verschlüsselungstechnik. Momentan ist es nicht möglich ein Passwort so zu ver- und entschlüsseln, dass die verschlüsselten Zeichenketten jedes Mal unterschiedlich lang werden. Somit ist die Sicherheit der App nicht vollumfänglich gegeben, da Brute-Force-Angriffe etwas erleichtert werden.

6. Abbildungsverzeichnis