Hybrid or native - what to choose?

Matthias Altstadt
matthias.altstadt@fau.de

Tobias Fertig
tobias.fertig@fau.de

Christian Happ
christian.happ@fau.de

Daniel Knogl
daniel.knogl@fau.de

Daniel Lohse
daniel.lohse@fau.de

Ulrich Mackert
ulrich.mackert@fau.de

Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg
AMOS SS15 Team 5

1. INTRODUCTION

2. ANDROID

In diesem Abschnitt sollen die Stärken und Schwächen der nativen Entwicklung für Android Systeme analysiert und zusammengefasst werden. Wir haben dafür die wichtigsten Punkte herausgenommen, die wir innerhalb des Sommersemester 2015 prüfen konnten.

2.1 IDE

Die von Google empfohlene IDE ist Android Studio, welche auf IntelliJ Idea aufbaut. Die IDE bekommt von uns volle Punktzahl, da es sehr angenehm war damit zu arbeiten. Die Autovervollständigung ist eine der intelligentesten, die wir bisher genutzt haben und auch die Funktionen bezüglich Refactoring sind unübertroffen. Das einzige Manko, das wir während des Semesters bemerkt haben, trat im Bereich der Unit-Tests auf. Hierfür musste zunächst der Modus gewechselt werden, was in dem *Build Variants* links unten möglich war. Dies war allerdings dank umfassenden Dokumentation von Google schnell gefunden.

2.2 Language

Bei Android Entwicklung treten vorrangig die Sprachen *Java* und *XML* auf. XML wird ausschließlich zum Layouting verwendet, während in Java die ganze Logik implementiert wird. Hier könnten jetzt alle Vor- und Nachteile der erwähnten Sprachen aufgelistet werden, allerdings würde das den Rahmen der Evaluierung sprengen. Für uns war Java definitiv kein negativer Punkt, da wir diese Sprache sehr gut beherrschen. Bei der Umsetzung der einzelnen Funktionalitäten waren wir defintiv nicht durch die Sprache eingeschränkt und hatten zumindest von dieser Seite her keinerlei Probleme.

2.3 Support

Die Kategorie Support erhält für Android maximale Punktzahl, da nicht nur die Community im Web rießig ist, sondern auch die Dokumentation für Entwickler[1], die Google zur Verfügung stellt. Wir hatten im Laufe des Semesters kein Problem, für das es nicht schon eine Lösung im Internet gab. Wichtig ist an dieser Stelle zu erwähnen, dass zunächst immer in der Dokumentation von Google selbst nachgeschlagen werden sollte, da einige Lösungen aus Foren oder ähnlichem in der aktuellen Android-Version nicht mehr funktionieren. Es war demnach meistens erheblich schneller direkt in der offiziellen Dokumentation nachzulesen als andere Quellen zu verwenden. Erst wenn die Beispiele aus der Dokumentation nicht funktionieren sollten, werden die Lösungen aus der Community interessant.

[1] `http://developer.android.com/index.html`

2.4 Geolocation

Auch diese Kategorie erhält 3 von 3 Punkten. In der Developer Doku von Google steht findet sich ein funktionsfähiges Code-Beispiel sowie zusätzliche Hinweise für korrekte Nutzung der Libraries. Es lässt sich ohne Probleme einstellen, ob ausschließlich GPS oder auch Wifi Ortung verwendet werden soll. Zudem steht in der Doku noch der Hinweis, dass der Listener für die Standpunktänderung wieder beendet werden muss, um Akku zu sparen. Die Einbindung von Google Maps gestaltete sich als etwas aufwändiger, da hierfür zunächst ein API-Key erstellt werden muss.

2.5 Notifications

Für die Notifications außerhalb der Application bietet Android interessante Möglichkeiten, die uns auch dieses Mal wieder zu 3 von 3 Punkten überzeugten. Im Fall von regelmäßigen Notifications oder Erinnerungen gibt es garkeine Probleme, da diese einfach beim Starten der App aktiviert werden können und dann bestehen bleiben. In unserem Fall hatten wir aber die Aufgabe, dass der Nutzer nur dann eine Notification erhält, wenn er am heutigen Tage noch keine Zeiten eingetragen hat. Zudem gilt die weitere Einschränkung, dass der Nutzer nur an einem Werktag benachrichtigt werden soll.

Um diese Notifications mit Constraints umzusetzen, bietet Android die Möglichkeit einen Receiver zu implementieren. Hierfür wird im Alarm-Manager des Systems ein Event gesetzt. Dies ist zu einem beliebigen Zeitpunkt in beliebigen Intervallen möglich. Zum gewünschten Zeitpunkt wird dann ein Intent an die Receiver Klasse der Application gesendet. Die Receiver-Klasse kann dann alle gewünschten Constraints prüfen und dann je nach Situation angepasste, einmalige Notifications erstellen.

2.6 Email

Das Versenden von Emails per Default-App stellte keinerlei Schwierigkeiten dar, so dass wir auch hier guten Gewissens die maximale Punktzahl vergeben konnten. Das einzige was dabei beachtet werden muss, sind die Berechtigungen für den Dateizugriff. Soll ein Anhang per Email versendet werden, so muss dieser zwingend in einem externen Speicherbereich abgelegt werden, ansonsten wird der Zugriff durch

die Email-Application verweigert und eine Email ohne Anhang wird versendet. Der Anhang wird aber trotzdem in der Email-App angezeigt, so dass ein Nutzer denken würde, dass die Email korrekt versendet wird.

2.7 Persistence

Auch hierfür bietet Android viele unterstützende Klassen, die ein schnelles Aufsetzen einer Datenbank ermöglichen. Sollte die standardmäßige SQLite Datenbank nicht den Anforderungen entsprechen, kann allerdings auch eine komplett manuell erstellte Persistenz verwendet werden. Dadurch werden auch NoSQL Datenbanken ermöglicht.

Sofern die Datenbankstrukturen nicht zu komplex sind, sind die von Android zur Verfügung gestellten Klassen eine große Erleichterung. Anhand der Versionsnummer triggert Android verschiedene Methoden im Lifecycle, die einem Umstieg auf andere Datenstrukturen vereinfachen sollen.

2.8 Design

Google bietet einige Designguides, die befolgt werden sollten, zeigt aber kaum Beispielcode, wie dieses Design ermöglicht werden kann. Der Entwickler muss hier also alles selbst entwickeln und wird nicht gezwungen sich an vorgegebene Guidelines zu halten. Zudem ist in Android sehr viel Spielraum beim Design vorhanden und es ist sehr viel möglich. Allerdings dauert es dadurch auch länger, ein paar Screens zu erstellen. Eine weitere Schwierigkeit ist die Vielfalt bei den Versionen, so kann es sein, dass manche UI-Anpassungen nicht auf allen Versionen funktionieren. Hierfür müssen dann je nach Version unterschiedliche Änderungen vorgenommen werden, was zu einem etwas erhöhten Aufwand führen kann. Der erhöhte Zeitaufwand bewegte uns dazu nur 2 von 3 Punkten zu vergeben.

2.9 Testing

Zum Testen der Android Apps gibt es viele Frameworks, sowohl kostenfrei als auch kostenpflichtig. Auch gewöhnlche Unit-Tests mit JUnit sind möglich, allerdings müssen dafür ein paar Konfigurationen innerhalb der Files als auch innerhalb der IDE vorgenommen werden. Diese sind zwar in der Developer Dokumentation zu finden, aber gerade die Nutzung von JUnit sollte einfacher funktionieren, weshalb wir hier nur 2 von 3 Punkten vergeben. Jedoch gab es gerade in diesem Bereich schon einige Verbesserung in der vergangenen Zeit, so dass wir zuversichtlich sind, dass dieses Problem Google bewusst ist.