Transferleistung Theorie/Praxis* □

Matrikelnummer:	
Freigegebenes Thema:	
Studiengang, Zenturie:	

** Studierende, die unter den Anwendungsbereich der PVO bis 03.02.2015 fallen, fertigen Transferleistungen weiterhin in der Form von Praxisberichten an und der Begriff hält Einzug in das Abschlusszeugnis. Ab dem Jahrgang 2016 hat der Begriff vollumfängliche Gültigkeit. In der Kommunikation hält der Begriff Transferleistungen ab sofort Einzug.*

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung

Der Markt für mobile Anwendungen (Apps) ist eine sich schnell verändernde Umgebung. Entsprechend müssen Apps regelmäßig aktualisiert werden, um sich weiterzuentwickeln und anzupassen. Daher kann ein Update Intervall für Android Apps von weniger als 20 Tagen als normal angesehen werden[14]. Obwohl App Updates damit ein wichtiger Teil von der Entwicklung von mobilen Applikationen und entscheidend für den Erfolg sind, gibt es dennoch einige Besonderheiten und Limitierungen bei dem Releasevorgang von mobilen Applikationen die einen starken Einfluss auf die Entwicklung und Verbreitungsstrategien haben. CodePush versucht diese Besonderheiten für React Native Anwendungen zu adressieren, indem es einen alternativen Releaseprozess zur Verfügung stellt.

Das Ziel dieser Arbeit ist, festzustellen, inwieweit CodePush in der Praxis für eine aktuelle sich in der Entwicklung befindenden React Native Anwendungen eingesetzt werden kann. Während der Entwicklung ist CodePush als ein mögliches Tool für Updates in Betracht gekommen. Dazu wird zunächst der klassische Releaseprozess für mobile Anwendungen dargestellt und analysiert. Dabei werden die Probleme aufgedeckt, die dieser zur Folge hat. Anschließend wird CodePush vorgestellt und die Einschränkungen und Möglichkeiten diskutiert. Abschließend wird argumentiert, wie CodePush Probleme des klassischen Releaseprozesses lösen kann. Am Ende der Arbeit findet sich eine Anleitung für ein minimales Setup, um CodePush zu verwenden, die die Ergebnisse aus der Arbeit aufgreift.

2 Releaseprozess für mobile Anwendungen

Im Gegensatz zu dem Releaseprozess für Desktop Anwendungen, bei dem jederzeit ein Update veröffentlicht werden kann[1], oder im Web, bei dem anschließend alle Nutzer dieses verwenden und keine veralteten Versionen im Umlauf sind, gibt es für mobile Anwendungen einige Besonderheiten und Limitierungen. Diese entstehen durch die Stores, die verwendet werden müssen, um die Apps zu vertreiben. Die Releaseprozesse für iOS Anwendungen im App Store und Android Anwendungen in Google Play sind sehr vergleichbar und laufen folgendermaßen ab:

1. Das Update wird hochgeladen und es wird ein Updatetext hinterlegt. Gegebenenfalls werden weitere Eigenschaften der App (Icon, Beschreibung, unterstützte Sprachen etc.) angepasst.
2. Das Update wird für ein Review des Storebetreibers[2] eingereicht. Updates auf Google Play werden meistens nur durch ein automatisiertes Review überprüft.
3. Der Storebetreiber reviewed die App und gibt anschließend Feedback. Entweder wird die App zugelassen oder abgelehnt.
4. Sofern die App zugelassen wurde, hat der Entwickler die Möglichkeit die App sofort oder

[1] Dies gilt nicht für Anwendungen, die über einen Store wie den Mac App Store vertrieben werden, der den Releaseprozess ebenfalls einschränkt.

[2] Der Storebetreiber ist im Fall des AppStores Apple Inc. und im Fall von Google Play Google LLC.

zu einem späteren Zeitpunkt zu veröffentlichen.

5. Nach der Veröffentlichung können die Nutzer das Update installieren.

Dabei ist die Dauer der Schritte drei und fünf nicht von den Entwicklern beeinflussbar und sind auch die Schritte, die für einen verzögerten Release sorgen. Die Folgen dieser beiden Probleme werden im Folgenden analysiert.

2.1 Review Dauer

Nachdem eine App eingereicht wurde, findet ein Review Prozess durch den Storebetreiber statt. Im Fall von Google Play ist dieser Prozess meistens automatisiert und dauert auch bei einem vollständigen Review weniger als einen Tag [12]. Damit stellt dies für normale Updates kein Problem dar. Im Gegensatz dazu braucht Apple für Updates im App Store etwa zwei bis vier Tage, kann aber, wie in Abbildung 1 dargestellt, auch deutlich mehr Zeit in Anspruch nehmen[15]. Das bedeutet, dass Fehlerbehebungen bei kritischen Bugs, die eine App zum Beispiel unbenutzbar machen, mehrere Tage brauchen, bis sie auf den Endgeräten installiert werden können. Dies kann zu direkten Umsatzeinbußen aber auch zu negativen Bewertungen führen, welche dauerhaft indirekt Umsatzeinbußen zur Folge haben können[6]. Auch können wichtige Sicherheitsupdates gegebenenfalls nicht zeitnah installiert werden. Es gibt von Apple die Möglichkeit ein beschleunigtes Review zu beantragen. Dies wird aber nur genehmigt, wenn Apple die Notwendigkeit sieht. Mögliche Gründe sind zum Beispiel `Critical Bug Fix` und `Time-Sensitive Event`. Außerdem werden nur begrenzt viel Anträge angenommen, sodass auch diese Methode keinen kurzfristigen Release sicher ermöglicht[4].

Die Review Dauer stellt damit vor allem im App Store und im begrenzten Maße auch bei Google Play ein Problem für zeitkritische Releases dar.

2.2 Verspätete Installation durch die Nutzer

Nicht alle Nutzer können oder wollen neue Updates zeitnah installieren. In Abbildung 2 ist dargestellt, wann wie viel Prozent der aktiven iOS Geräte[3] das aktuellste Update der freenet Mail App im App Store nach der Veröffentlichung installiert haben[4]. In Tabelle 1 sind wichtige Zeitpunkte dargestellt. Auffällig ist, dass nach zwei Wochen mehr als 10 % und nach einem Monat mehr als 5 % weiterhin eine älter Version verwenden. Führte man nun eine Änderung an Serversystemen durch, die inkompatibel zu den älteren Versionen wäre, verlöre man 5 % bis 10 % der Kunden, bis diese ebenfalls ein kompatibles Update installieren. Auch hier kann es zu Umsatzeinbußen und negativen Bewertungen kommen.

[3]Ein Gerät mit mindestens einer Session an dem betrachteten Tag wird als aktiv gezählt. Der Nutzer muss zustimmen, damit die Daten erhoben werden (Opt-in).

[4]Die freenet Mail App hat täglich etwa 8500 bis 10000 aktive Geräte.

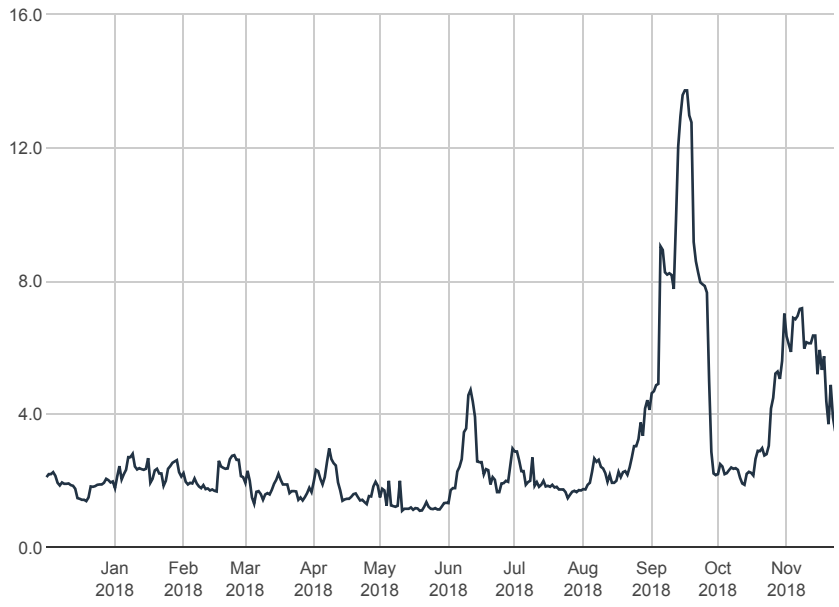

Abbildung 1: Durchschnittliche Review Dauer in Tagen im App Store

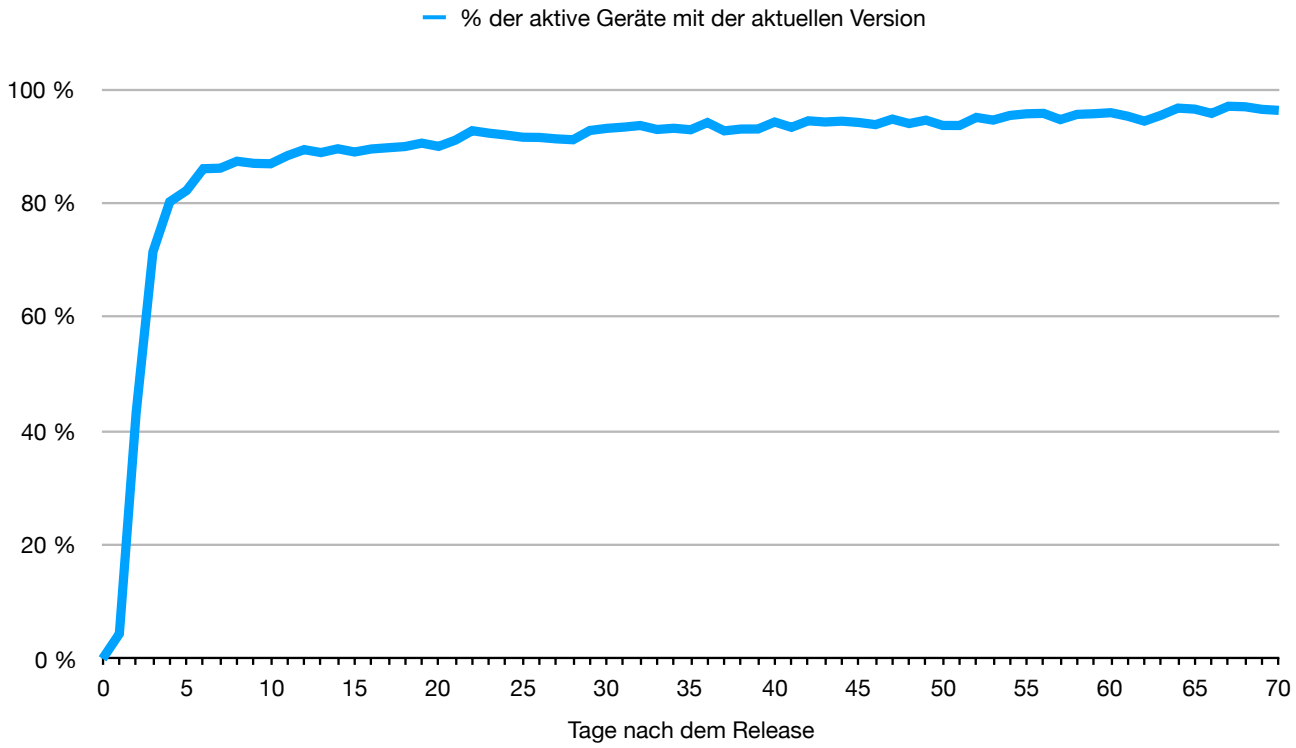

Abbildung 2: Aktive Geräte mit der aktuellen Version der freenet Mail App

Tabelle 1: Aktive Geräte mit der aktuellen Version der freenet Mail App

Tage nach dem Release	% der aktiven Geräte
1	4 %
2	43 %
3	71 %
4	80 %
17	90 %
47	95 %

3 CodePush

CodePush ist ein in App Center integrierter Service von Microsoft, um App Updates für React Native[5] auf Endgeräten zu installieren[9]. React Native ist ein Framework, um mobile Anwendungen für iOS und Android zu entwickeln und dabei möglichst viel Code für beide Plattformen verwenden zu können. Entsprechend besteht eine React Native App aus zwei Komponenten.

Zum einen gibt es eine JavaScript Komponente. Diese enthält die Anwendungslogik und definiert das User Interface. Diese wird in JavaScript oder TypeScript entwickelt. Außerdem können weitere Ressourcen wie Bilder mit eingebunden werden. Der Code wird für einen Release zusammen mit den Ressourcen mit dem Bundler *Metro* zu einem `.jsbundle` gepackt[10] und kann auf beiden Platformen ausgeführt werden.

Zum anderen gibt es die native Komponente. Diese stellt die Verbindung zwischen der JavaScript Komponente und dem Betriebssystem, also iOS oder Android, zur Verfügung. Die native Komponente, welche im engeren Sinne als die eigentliche App verstanden werden kann, kann beliebig erweitert werden, um mehr plattformspezifische Funktionalität der JavaScript Komponente verfügbar zu machen. Entsprechend muss diese Komponente für jede Platform einzeln entwickelt werden. Auf diesem Weg kann der JavaScript Code für alle Plattformen verwendet werden und dennoch plattformspezifische Funktionalitäten nutzen.

CodePush nutzt diese Zweiteilung, indem es ein versioniertes Repository der gepackten JavaScript Komponente bereitstellt, sodass diese auch nach der Installation der App ohne App Update mit dem klassischen Prozess ausgetauscht werden kann. Dazu gibt es ein SDK für React Native um mit dem Repository zu interagieren.

[5] CodePush unterstützt auch Apache Cordova. Darauf wird aber im weiteren Verlauf der Arbeit nicht eingegangen, da der Focus auf React Native liegt.

3.1 Aufbau

Jede App in Microsoft's App Center kann für CodePush beliebig viele Deployment Environments haben, wobei der Standard *Staging* und *Production* ist. Verschiedene Environments können verwendet werden, um ein Update intern zu testen, bevor es den Nutzern zur Verfügung gestellt wird. Jede Deployment Environment hat einen Deployment Key, über den die App später Updates finden und laden kann. In jeder Deployment Environment können bis zu fünfzig Versionen der JavaScript Komponente hinterlegt und downloadbar sein[11]. Autorisierte Benutzer können neue Updates für eine Deployment Environment erstellen oder bearbeiten. Ein Update hat folgende Eigenschaften:

- **Label:** Eine String wie `v1`.
- **Binary App Version:** Die Versionsnummer der nativen Komponente aus dem App Store oder von Google Play, die das Update installieren sollen. Ein Update kann mehreren Versionsnummern zugeordnet sein.
- **Rollout percentage:** Prozent der Nutzer, die dieses Update installieren sollen. Der Anteil kann nur erhöht werden, auch wenn noch kein Nutzer das Update installiert hat[6].
- **Mandatory:** Es handelt sich um ein Pflichtupdate, dass installiert werden muss, sobald durch eine Synchronisation das Update gefunden wurde.
- **Disabled:** Das Update kann nicht installiert werden. User, die das Update bereits installiert haben, installieren automatisch das vorherige Update.

3.2 Synchronisation

Die Synchronisation ist dafür verantwortlich, zu ermitteln, ob der Client upgedatet werden muss. Dabei wird der Prozess vom Client gestartet. Das bedeutet, dass kein Push mit der Infrastruktur von CodePush und App Center möglich ist. Stattdessen muss der Client eine Polling Funktion haben. Das SDK für React Native stellt drei Methoden für den Start der Synchronisation zur Verfügung, die Verschiedene Vor- und Nachteile haben und für den Einsatzzweck von CodePush in den einzelnen Projekten evaluiert werden müssen. Zunächst wird die Möglichkeit geboten, bei jedem Start der App eine Synchronisation durchzuführen. Auf diese Weise werden die Updates selten gefunden und es kann länger dauern, bis ein verfügbares Update installiert wird. Die zweite Methode startet den Prozess bei jedem Öffnen der App. Dadurch wird sichergestellt, dass die App beim Öffnen mögliche Updates kennt. Bei der letzten Methode wird der Prozess nie automatisch gestartet. In diesem Fall muss die App den Prozess selber starten. In jedem Fall besteht die Möglichkeit den Prozess jederzeit manuell zu starten, wie zum Beispiel, wenn der Nutzer dies fordert oder bei einer eingehenden Push Nachricht. Auf diese Weise kann eine Push Funktionalität zu CodePush hinzugefügt werden.

Um die Synchronisation zu starten, führt der Client eine https GET Anfrage durch. Dabei enthalten die Query Parameter unter anderem den Deployment Key, die Versionsnummer der

[6]Wurde die Rollout percentage zum Beispiel auf 50 % gestellt, kann der Wert anschließend nicht mehr auf Werte kleiner als 50 % verändert werden.

nativen Komponente und einen Hash der aktuell installierten JavaScript Komponente. Außerdem wird eine Identifikationsnummer für den Client mitgesendet, um Features wie *Staged Rollout* zu ermöglichen.

Tabelle 2: Query String der Anfrage

Key	Value (Beispiel)
deploymentKey	Clc3_...
appVersion	1.0
packageHash	56ac8...
isCompanion	
label	v8
clientUniqueId	C6878...

Die Antwort enthält Informationen über das Update, das installiert werden kann. Dazu gehört unter anderem die Download URL und das Flag, ob es sich um ein Pflichtupdate handelt.

Antwort des Servers:

```
{
    "updateInfo": {
        "downloadURL": "https://codepushupdates.azureedge.net/storagev2/<id>",
        "isAvailable": true,
        "isMandatory": true,
        "appVersion": "1.0",
        "packageHash": "1bd6c900d2d10125...",
        "label": "v8",
        "packageSize": 219113,
        "updateAppVersion": false,
        "shouldRunBinaryVersion": false
    }
}
```

Anschließend entscheidet die App, ob das Update ignoriert, heruntergeladen oder installiert wird. Wie der Client vorgehen soll, lässt sich durch das SDK konfigurieren. Um Statistiken zu erstellen wird anschließend eine https POST Anfrage gesendet. Auch hier wird wieder die Identifikationsnummer mitgesendet. Eine Anfrage kann folgendermaßen aussehen:

Body der Anfrage:

```
{
    "appVersion": "1.0",
    "deploymentKey": "Clc3_...",
    "clientUniqueId": "C6878A70-99D0-43FB-B0BD-8772904338A7",
    "label": "v8",
    "status": "DeploymentSucceeded",
    "previousLabelOrAppVersion": "1.0",
    "previousDeploymentKey": "Clc3_..."
}
```

3.3 Funktionalität

Im Folgenden werden einige Funktionen vorgestellt, die von CodePush unterstützt werden und für eine Entscheidung über die möglichen Einsatzgebiete hinzugezogen werden können.

3.3.1 Rollback

CodePush unterscheidet zwischen zwei Arten von Rollback. Stürzt die App nach der Installation eines Updates ab, führt der Client einen *client-side rollback* durch. Dabei wird das neue Update gelöscht und das alte weiterverwendet[10]. Ein *server-side rollback* bedeutet, dass das Update in App Center deaktiviert wird. Clients auf älteren Versionen installieren dieses Update dann nicht mehr und bei Clients mit dieser Version wird automatisch eine ältere Version installiert.

Durch diese Funktionalität kann sichergestellt werden, dass durch ein Update mit CodePush die App nicht mehr nutzbar ist.

3.3.2 Staged Rollout

Mit Staged Rollouts kann ein Update nur für einen Teil der Nutzer ausgespielt werden, wie es auch bei Google Play und im App Store möglich ist. Über diesen Weg kann getestet werden, ob ein Update Bugs enthält, die beim Testen nicht aufgefallen sind. Dabei lässt sich der Anteil der Nutzer die das Update erhalten sollen auf 1 % genau angeben. Damit ist das System flexibler als im App Store[5].

3.3.3 Statistik

Wie in Abschnitt 3.2 beschrieben, werden von App Center Statistiken geführt. Folgende Daten können in App Center eingesehen werden:

- Anzahl der aktiven Geräte
- Anzahl der aktiven Geräte eines Updates
- Anzahl der Installationen

- Anzahl der Downloads[7]
- Anzahl der client-side Rollbacks

Mithilfe dieser Daten kann ermittelt werden, ob es mit einem Update Probleme gibt. Wenn es zu Problemen kommt, kann mit CodePush anschließend ein neues Update erstellt oder das fehlerhafte deaktiviert werden.

3.4 Einschränkungen

CodePush hat auch einige Limitierungen, die man beachten muss. Diese werden im Folgenden weiter erläutert.

3.4.1 Ressourcen

Es lassen sich nur Ressourcen austauschen, die mit `require("./image.png")` eingebunden werden. Dadurch lassen sich die meisten Ressourcen wie Bilder oder Videos mit CodePush austauschen, wenn diese entsprechend eingebunden wurden. Nicht bearbeiten lassen sich hingegen alle Ressourcen, die von der nativen Komponente aus verwendet oder die nicht mit `require(path: string)` eingebunden wurden. Daher sollte man darauf achten folgende Syntax zu vermeiden um das Tauschen der Ressourcen später einfach durchführen zu können:

```
<Image source={{ uri: './image.png' }>
```

Es gibt jedoch Komponenten, die nur diese Art der Ressourceneinbindung ermöglichen und damit über CodePush nur mit deutlichem Mehraufwand angepasst werden können. Außerdem lassen sich nicht das App Icon, der Splash Screen, Berechtigungen wie der Zugriff auf die Kamera und weitere Einstellungen der App anpassen, sodass hierfür ein klassisches App Update benötigt wird.

3.4.2 Fehlender App Store und Google Play Auftritt

Updates, die über CodePush durchgeführt werden, sind im App Store und Google Play nicht sichtbar. Dies kann bedeuten, dass bei vielen Updates über CodePush weniger Aktivität in den Stores sichtbar ist. Außerdem bekommen Nutzer gegebenenfalls nicht mit, wenn Bugs behoben werden. Dies spielt jedoch nur eine untergeordnete Rolle, da die Nutzer bei automatischen Updates die Aktivität ebenfalls nicht wahrnehmen.

3.4.3 Veränderung des nativen Codes nicht möglich

Aufgrund der Funktionsweise von CodePush lässt sich die native Komponente der App nicht verändern. Da diese jedoch für manche Änderungen notwendigerweise verändert werden muss, bietet CodePush hier keine Möglichkeit das Update auszuliefern. Auch lässt sich die React Native Version nicht ändern, da diese ebenfalls Teile in der nativen Komponente hat.

[7] Dieser Wert scheint sich nicht zu erhöhen, wenn das Update interaktiv sofort installiert wird.

Daher sollte vor jedem Release mit CodePush getestet werden, ob die nativen Komponenten, die das Update installieren soll, kompatibel zu der JavaScript Komponente sind.

3.4.4 Apple Guidelines

Während es durch Google für Google Play keine Beschränkungen für das Nachladen von ausführbarem Code gibt[8], macht Apple für Apps des App Stores einige Vorgaben.

Sowohl das *Apple Developer Program License Agreement*[1, S. 15] als auch das *Apple Developer Enterprise Program License Agreement*[2, S. 13] enthalten folgenden Absatz:

> 3.3.2 Except as set forth in the next paragraph, an Application may not download or install executable code. Interpreted code may be downloaded to an Application but only so long as such code: (a) does not change the primary purpose of the Application by providing features or functionality that are inconsistent with the intended and advertised purpose of the Application as submitted to the App Store, (b) does not create a store or storefront for other code or applications, and (c) does not bypass signing, sandbox, or other security features of the OS.

Außerdem enthalten die *App Store Review Guidelines*[3] folgenden Absatz:

> 2.5.2 Apps should be self-contained in their bundles, and may not read or write data outside the designated container area, nor may they download, install, or execute code which introduces or changes features or functionality of the app, including other apps. [...]

Entsprechend ist die Nutzung von CodePush im App Store möglich, solange mit den Updates keine Funktionalität geändert oder hinzugefügt wird[8]. Dennoch kann Apple jederzeit die Richtlinien ändern und damit das Nachladen von Code weiter verbieten.

4 Einsatzmöglichkeiten

Aus den genannten Funktionalitäten und Einschränkungen lassen sich die Einsatzmöglichkeiten von CodePush ableiten. Zunächst ist CodePush kein Ersatz für die klassischen Updates, die die Stores zur Verfügung stellen. Dies liegt an den verschiedenen Einschränkungen von CodePush. Zum einen können Teile der App nicht mit CodePush verändert werden und zum anderen dürfen im App Store keine größeren Änderungen durchgeführt werden. Außerdem sollte beachtet werden, dass die Nutzer den klassischen Releaseweg über den App Store oder Google Play gewohnt sind und Änderungen möglicherweise nur nach einem Update erwarten.

Dennoch gibt es Szenarien, in denen CodePush eine gute Ergänzung zu dem klassischen Releaseweg ist. In Abschnitt 2 wurden zwei Probleme identifiziert. Zum einen die lange Review Dauer, die schnelle Releases nicht möglich machen und zum anderen die späte Installation

durch die Nutzer. Im Folgenden wird für diese beiden Fälle jeweils eine Möglichkeit gezeigt mit CodePush die Probleme zu umgehen.

4.1 Schnelle Releases

Eine große Stärke von CodePush sind schnelle Releases, die durch das nicht benötigte Review durch die Storebetreiber möglich werden. Dies bietet eine Möglichkeit Bugfixes in der UI oder der Anwendungslogik schnell zu veröffentlichen. Dadurch kann man schlechte Bewertungen, die auch langfristig schaden können[13], oder in Extremfällen Ausfälle über mehrere Tage verhindern[7, S. 8].

Da die CodePush Updates nur die JavaScript Komponente tauschen, lassen sich Bugs im nativen Teil nicht beheben, dennoch besteht hier die Möglichkeit fehlerhafte Feature über den JavaScript Code zu deaktivieren und ein reguläres Update über den klassischen Releaseweg einige Tage später bereitzustellen. Durch diese Kombination von klassischen Releases und CodePush Releases kann man dem Nutzer neue Features mithilfe von Updates über den App Store und Google Play zur Verfügung stellen und hat eine Möglichkeit im Notfall innerhalb von Minuten ein Bugfix zu veröffentlichen und dadurch Ausfälle zu vermeiden. Um sicherzustellen, dass das Update schnellstmöglich installiert wird und die Nutzer keine schlechten Bewertungen veröffentlichen, sollten diese Pflichtupdates sein und die Synchronisation möglichst häufig durchgeführt werden. Hier eignet sich zum Beispiel jedes Öffnen der App, da damit sichergestellt wird, dass Nutzer vor dem Verwenden der App mögliche kritische Bugfixes installiert haben.

4.2 Updates für alte Versionen

Eine weitere Stärke von CodePush ist die Möglichkeit Updates für alte Versionen zu erstellen. Wie in Abschnitt 2.2 dargestellt, dauert es lange, bis alle Nutzer ein Update installiert haben. Dies erschwert das Updaten von zum Beispiel Backendsystemen, da weiterhin alte Clients verwendet werden, die alte Systeme erwarten. Mithilfe von Updates der alten Apps, kann dort die Integration mit Backendsystemen aktualisiert werden, sodass diese weiterhin funktionieren. Dies setzt voraus, dass alle nötigen Elemente in der JavaScript Komponente implementiert sind.

Man muss beachten, dass man nun eine Kombination aus der Versionsnummer der nativen Komponente, die der Versionsnummer aus dem App Store oder von Google Play entspricht, und der Versionsnummer der JavaScript Komponente. Diese muss bei Crashreports, Bugreports und Supportanfragen berücksichtigt werden. Man sollte deshalb auch den Nutzern die Möglichkeit geben die vollständige Versionsnummer einzusehen. Gegebenenfalls kann es bei seltenen Synchronisieren auch sinnvoll sein den Nutzern die Möglichkeit zu geben, diesen manuell zu starten, um zum Beispiel bei Supportanfragen sicherstellen zu können, dass die neuste Version installiert ist. Des weiteren sollte beachtet werden, dass die Versionierung bei vielen CodePush Updates unübersichtlich werden kann und sehr viele Versionskombinationen entstehen können. Besonders wenn CodePush Updates für mehrere Versionen der nativen Komponente ausgelegt

sind, ist ein umfassendes Testen nicht mehr möglich. Dagegen hilft vor allem das Beschränken der Updates auf ein Verfahren. Da viele der Updates nicht über CodePush durchgeführt werden können, sollten CodePush Updates nur selten eingesetzt werden.

CodePush Updates stellen damit eine sinnvolle Ergänzung zu den klassischen Updates über die Stores dar und können, wenn diese richtig eingesetzt werden, die aufgezeigten Probleme lösen.

5 Setup

Die folgenden Schritte erstellen ein Projekt mit CodePush oder fügen CodePush einem bestehenden Projekt hinzu, sodass Updates der JavaScript Komponente durchgeführt werden können.

5.1 Voraussetzungen

1. Es wird eine React-Native App benötigt. Diese kann gegebenenfalls mit `react-native init <name>` erstellt werden.
2. Die node Command Line Tools `appcenter-cli` und `code-push-cli` müssen installiert sein. Diese können gegebenenfalls mit `yarn global add appcenter-cli code-push-cli` installiert werden.

5.2 Anleitung

1. Zunächst muss sowohl für iOS als auch für Android eine App im App Center angelegt werden:

   ```
   code-push app add <name>-ios ios react-native
   code-push app add <name>-android android react-native
   ```

2. Anschließend können die *Deployment Keys* ausgelesen werden:

   ```
   code-push deployment ls <name>-ios -k
   code-push deployment ls <name>-android -k
   ```

3. Um CodePush in der App zu verwenden, muss das CodePush SDK installiert und gelinked werden:

   ```
   yarn add react-native-code-push
   react-native link react-native-code-push
   ```

   Während des Linkvorgangs müssen die Deployment Keys aus Schritt 2 angegeben werden. Hier können zunächst die Keys der Staging Environment verwendet werden. In Schritt 6 wird die App so umgebaut, dass verschiedene Deployment Keys verwendet werden können.

4. Nun kann CodePush in der App implementiert werden. Die einfachste Möglichkeit ist das automatische Überprüfen beim Appstart. Dazu muss die `App.js` folgendermaßen angepasst

werden:

```
// Import CodePush SDK
import CodePush from "react-native-code-push";

// Remove `export default`
class App extends Component<Props> {...}

// Export wrapped app
export default App = CodePush(App);
```

5. Jetzt kann ein Release über CodePush ausgespielt werden. Dafür muss zuerst die App auf einem Gerät oder in einem Simulator/Emulator installiert werden. Wenn nun der Javascript Code verändert wird, kann ein Release erstellt werden.

   ```
   appcenter codepush release-react -a <username/organization>/<appname>-<ios/android>
   -d Staging
   ```

   Wenn nun die App neu gestartet wird, wird das update installiert.

6. Abschließend sollte die App so konfiguriert werden, dass man sowohl *Staging* als auch *Production* nutzen kann. Unter iOS wird der Deployment Key aus der `Info.plist` geladen. Unter Android wird der Key in der `MainApplication.java` aus der `strings.xml` verwendet. Außerdem wird in den meisten Fällen nicht gewünscht sein, dass in Debug Versionen CodePush verwendet wird, da dann der Packager nicht verwendet werden kann.

6 Literaturverzeichnis

[1] Apple, *Apple Developer Program License Agreement*. 2018.

[2] Apple, *Apple Developer Enterprise Program License Agreement*. 2018.

[3] Apple, *App Store Review Guidelines*. 2018.

[4] Apple, „Contact the App Review Team“, 2018. [Online]. Verfügbar unter: https://developer.apple.com/contact/app-store/?topic=expedite. [Zugegriffen: 30-Nov-2018].

[5] Apple, „What is phased release for automatic updates“, 2018. [Online]. Verfügbar unter: https://itunespartner.apple.com/en/apps/faq/Managing%20Your%20Apps_Metadata%20and%20Submissions. [Zugegriffen: 30-Nov-2018].

[6] S. Hassan, W. Shang, und A. E. Hassan, „An empirical study of emergency updates for top android mobile apps“. Springer US, Feb-2017.

[7] M. E. Joorabchi, A. Mesbah, und K. Philippe, „Real Challenges in Mobile App Development“, in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, Baltimore, MD, USA, 2013.

[8] Microsoft und Contributors, „Store Guideline Compliance“, *Visual Studio App Center*, 10-Nov-2018. [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/appcenter/distribution/codepush/react-native#store-guideline-compliance. [Zugegriffen: 28-Nov-2018].

[9] Microsoft und Contributors, „CodePush“, *Visual Studio App Center*, 08-Aug-2018. [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/appcenter/distribution/codepush/. [Zugegriffen: 28-Nov-2018].

[10] Microsoft und Contributors, „React Native Client SDK“, *Visual Studio App Center*, 08-Aug-2018. [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/appcenter/distribution/codepush/react-native. [Zugegriffen: 28-Nov-2018].

[11] Microsoft und Contributors, „Releasing Updates“, *Visual Studio App Center*, 08-Aug-2018. [Online]. Verfügbar unter: https://docs.microsoft.com/en-us/appcenter/distribution/codepush/cli#releasing-updates. [Zugegriffen: 28-Nov-2018].

[12] Microsoft und Contributors, „APP Review Times: How fast will your app be published“, 04-Aug-2016. [Online]. Verfügbar unter: https://www.androidb.com/2016/08/app-review-times-for-google-play-aka-how-fast-will-your-app-be-published/. [Zugegriffen: 03-Dez-2018].

[13] S. Oh, H. Baek, und J. Ahn, „The effect of electronic word–of–mouth (eWOM) on mobile application downloads: an empirical investigation“, *International Journal of Mobile Communications*, Bd. 13, Nr. 2, 2015.

[14] S. Shen, X. Lu, Z. Hu, und X. Liu, „Towards Release Strategy Optimization for Apps in Google Play“, in *Proceedings of the 9th Asia-Pacific Symposium on Internetware*, Shanghai,

China, 2017.

[15] Shiny Development, „Average App Store Review Times“. [Online]. Verfügbar unter: http://appreviewtimes.com. [Zugegriffen: 28-Nov-2018].