

## 7 Test- und Veröffentlichungsstrategien für Apps

Bis hierhin haben Sie eine Menge über den App-Test und die unterschiedlichen Testtechniken und Testansätze erfahren. Dieses Kapitel beinhaltet App-Test- und Veröffentlichungsstrategien und was Sie in Ihren Strategien berücksichtigen müssen. Beide Strategien, sowohl die Teststrategie als auch die Veröffentlichungsstrategie, sind sehr wichtig für jedes Projekt und Sie sollten nicht unterschätzen, wie nützlich es ist, beide in einem schriftlichen Dokument festzuhalten. In den folgenden Abschnitten werde ich Ihnen einige Beispiele für Teststrategien und Veröffentlichungsstrategien vorstellen. Außerdem werden einige Fragen aufgeworfen, die für das Aufstellen Ihrer eigenen Strategien hilfreich sein können.

### 7.1 Mobile Teststrategie

Generell ist die Teststrategie ein Dokument, das den Testansatz und die Arbeit beschreibt, die im Softwareentwicklungsprozess gemacht wird. Diese Strategie kann benutzt werden, den Projektmanager, den Produktmanager, die Entwickler, die Designer und alle, die in die Softwareentwicklung involviert sind, über die zentralen Themen des Softwaretestprozesses aufzuklären.

Eine Teststrategie beinhaltet das Testobjekt, die Teststufen und Testtechniken, die für den Test des SUT (System Under Test) benötigten Ressourcen und die Testumgebung. Es werden außerdem die Produktrisiken beschrieben und wie diese für die Stakeholder und Kunden minimiert werden können. Schließlich sind Definitionen vom Testeingangs- und vom Testendekriterium enthalten.

Die definierte Teststrategie wird Ihnen helfen, wird Sie daran erinnern und wird Sie führen, die wichtigsten Komponenten und Features der App nicht zu vergessen. Sie sollten sich wirklich die Zeit nehmen, die einzelnen Schritte und Ressourcen aufzuschreiben, die benötigt werden, um die App zu testen. Außerdem dokumentiert die Teststrategie Ihre Arbeit und Ihre Bemühungen innerhalb des Projekts. Das Aufschreiben einer Teststrategie bedeutet nicht, dass diese in Stein gemeißelt ist und Sie keine Änderungen mehr machen können. Im Gegenteil, es ist wichtig, dass Sie von Zeit zu Zeit mit Ihrem Team über die Teststrategie

sprechen, damit Sie Produktänderungen oder andere aktuellere Umstände einbauen können.

Allerdings gibt es nicht die eine mobile Teststrategie, die von jedem Team oder für jede App genutzt werden kann, da fast jede App andere Anforderungen, Ziele und Zielgruppen hat und auf verschiedenen Plattformen läuft, was es unmöglich macht, die komplette Strategie in jedem Projekt wiederzuverwenden.

Die folgenden Abschnitte sollen Ihnen eine Idee skizzieren, wie Sie eine Teststrategie für Apps gestalten können. Sie können diese als Startpunkt und Leitfaden nutzen, um Ihre eigene mobile Teststrategie zu erstellen.

**Wichtig:**

Die Erstellung einer mobilen Teststrategie erfordert nicht unbedingt, dass Sie eine endlose Dokumentation schreiben, da Sie und andere Tester einfach nicht die Zeit und/oder die Ressourcen haben werden, alles durchzuspielen. Flexibilität ist gefragt im mobilen Testbusiness. Eine mobile Teststrategie soll dazu dienen, Sie und andere, die in das Projekt eingebunden sind, in Bezug auf die wichtigen Teile des Testprozesses auf dem Laufenden zu halten.

### 7.1.1 Anforderungen definieren

Als Erstes sollten Sie und Ihr Team in der Anfangsphase des Projekts die Anforderungen und Features Ihrer App definieren. Schreiben Sie sie alle auf und beschreiben Sie die Features und möglichen Nutzerszenarien, um ein besseres Gefühl für die App und die potenziellen Nutzer zu bekommen. Zu diesem Zeitpunkt sind grobe Beschreibungen der Anforderungen und Features völlig in Ordnung, da sie detaillierter im Entwicklungsprozess spezifiziert werden. Eine schriftliche Aufzeichnung der Anforderungen wird es Ihnen viel einfacher machen, die mobile Teststrategie abzuleiten.

Im Folgenden finden Sie eine Liste mit möglichen Beschreibungen und Features:

- Ein Anmeldeformular zur Verfügung stellen.
- Eine Login-Maske zur Verfügung stellen, um Zugriff auf den App-Inhalt zu erhalten.
- Eine Logout-Funktion zur Verfügung stellen.
- Eine Suchfunktion innerhalb der App implementieren.
- Der Nutzer soll in der Lage sein, ein Nutzerprofil zu erstellen.
- Der Nutzer soll in der Lage sein, Inhalte mit anderen Nutzern auszutauschen.
- Der Nutzer soll in der Lage sein, Inhalte in den sozialen Netzwerken auszutauschen.
- Der Nutzer soll in der Lage sein, Fotos zu machen.
- Die App soll in verschiedenen Sprachen (Englisch, Deutsch, Französisch etc.) verfügbar sein.

Außerdem sollten Sie wissen, wer die App nutzen wird. Wie in Kapitel 3 beschrieben, sollten Sie Ihre Zielgruppe und deren Erwartungen kennen. Sammeln Sie so viele Informationen wie möglich, um wichtige Einblicke in die Nutzerszenarien der Kunden zu bekommen.

Im Folgenden finden Sie eine Zusammenfassung von möglichen Informationen über die Zielgruppe (die komplette Liste finden Sie in Kap. 3):

- Geschlecht
- Monatliches Einkommen
- Bildungshintergrund
- Ort
- Andere genutzte Apps
- Smartphone-Gewohnheiten
- Genutzte Geräte

**Wichtig:**

Wenn Sie nichts über Ihre Zielgruppe wissen, suchen Sie nach Statistiken über mobilplattformspezifische Betriebssysteme und Hardwarespezifikationen. Außerdem analysieren und sammeln Sie Informationen über Apps, die Ihrer ähnlich sind. Das ist ein guter Ausgangspunkt, um potenzielle Nutzerinformationen zusammenzutragen.

Anhand der Anforderungen und Features Ihrer App und mit dem Wissen über Ihre Zielgruppe können Sie spezifische Fragen stellen, um Informationen für Ihre Testarbeiten und den Testumfang zu sammeln:

- Ist es wichtig, kritische Fehler schnell zu finden?
- Sollte die App nur in den gewöhnlichen Nutzerszenarien getestet werden?
- Auf welcher Mobilplattform sollten die Tests durchgeführt werden?
- Welches sind die Mobilfunkanbieternetze der Kunden?
- Gibt es in der App Teile, die sich wahrscheinlich regelmäßig ändern werden?
- Ist der Veröffentlichungszeitpunkt (Einreichung) schon bekannt?
- Gibt es einen Zeitplan für zukünftige Veröffentlichungen?

Zögern Sie nicht, diese Art von Fragen zu stellen, weil sie wichtig sind, und die Antworten Ihnen helfen werden, die nächsten Testschritte und die Prioritäten zu definieren. Machen Sie sich keine Sorgen, wenn Sie eine Frage vergessen haben zu stellen, bevor Sie angefangen haben, die Teststrategie zu schreiben – es ist besser, die Frage dann zu stellen, wenn sie auftaucht, als wenn Sie sie gar nicht stellen.

Im nächsten Schritt sollten Sie Informationen über die Entwicklungsumgebung im Unternehmen sammeln. Es ist wichtig, zu wissen, welche Werkzeuge verwendet werden, um eine Verbindung zwischen Entwicklung und Test aufzubauen. Welcher Continuous Integration (CI) Server wird genutzt? Welche Werkzeuge werden für die Erstellung der App benutzt? Und welche Backend-Technologien

werden eingesetzt, um die Anfragen der App zu beantworten? Außerdem müssen Sie die Architektur der Produktivumgebung kennen.

Um diese Informationssammlung anzugehen, sollten Sie die Antworten der folgenden Fragen ermitteln:

- Welche Arten von Softwareentwicklungswerkzeugen sind bereits im Unternehmen vorhanden und werden genutzt?
- Gibt es standardmäßig eine Build-Pipeline?
- Welcher Continuous Integration Server wird für das Projekt eingesetzt?
- Welche Werkzeuge werden verwendet, um die App zu erstellen?
- Welche Technologien werden benutzt, um die mobilen Anfragen zu bearbeiten?
- Welche Fähigkeiten, wie Programmiersprachen, sind im App-Team vorhanden?
- Wie viele Mobilgeräte für das Testen sind im Unternehmen vorhanden?
- Welche Werkzeuge und Technologien werden in der Produktivumgebung eingesetzt?
- Wie viele App-Nutzer erwarten wir?

Die Antworten auf diese Fragen werden Ihnen z.B. helfen, das Testautomatisierungswerkzeug mithilfe des technologischen Wissens innerhalb Ihres Teams auszuwählen. Sie werden Ihnen helfen, die Teststufen und Testtechniken zu definieren, und außerdem werden sie Ihnen einen ersten Überblick von allen Technologien liefern, die im Projekt enthalten sind. Kenntnisse über die Entwicklung, den Test und die Produktivumgebung sind sehr wichtig, wenn Sie später das Testen innerhalb des Projektes koordinieren.

**Wichtig:**

Die Anforderungen und Features zu sammeln ist wichtig, da Sie diese nützlichen Informationen für Ihre mobile Teststrategie brauchen. Diese Informationen sind ein guter Ausgangspunkt, um Ihre Testaktivitäten zu planen, und sie werden Ihnen helfen, den Testumfang zu definieren.

### 7.1.2 Testumfang

Wenn Sie die Anforderungen erst einmal definiert haben, können Sie den Testumfang für Ihre Teststrategie festlegen. Da es nicht möglich ist, die App auf jeder erdenklichen Hard- und Softwarekombination zu testen, sollten Sie den Rahmen für Ihre Testaufwände reduzieren und sich zunächst auf die wichtigen Dinge der App konzentrieren.

Es gibt vier Möglichkeiten, den Testumfang zu reduzieren:

- Testen von Einzelgeräten
- Testen von mehreren Geräten
- Testen der maximalen Anzahl an Geräten
- Testen der Anwendungsfälle

### Testen von Einzelgeräten

Der Testumfang für ein Einzelgerät konzentriert sich auf ein Mobilgerät während des Testens. Dieser Ansatz kann oder wird genutzt, wenn nur ein Gerät von der App unterstützt wird oder im Projekt nicht genug Zeit vorhanden ist. Bei zeitlicher Dringlichkeit werden Sie wahrscheinlich nur das beliebteste Gerät wählen, das von Ihrer Kundenzielgruppe benutzt wird. Dieses Gerät wird für das Testen nur mit einem Mobilfunkanbieter und eventuell einer Wi-Fi-Verbindung genutzt. Ein anderer Ansatz könnte sein, ein Gerät aus der Gerätegruppe mit älteren Hardwarespezifikationen auszuwählen, weil dieses für die Entwickler mehr Probleme hinsichtlich des Supports bereiten kann. Sie werden wahrscheinlich mehr Fehler und Probleme im Bereich Performanz oder andere Auffälligkeiten mit diesem Gerät finden als mit dem neuesten Gerät.

Nur ein Gerät zu benutzen, kann für die App, für den Erfolg des Projektes oder auch für das ganze Unternehmen sehr gefährlich sein. Es ist sehr wahrscheinlich, dass Sie wichtige Fehler nicht finden werden, die auf anderen Geräten auftreten, und dass Ihre Kunden schlechte Bewertungen in den App-Stores einstellen werden. Wenn die App aber nur ein Gerät unterstützt – wenn es sich z. B. um eine interne Firmen-App handelt –, kann dieser Ansatz genutzt werden. Auf der anderen Seite ist es besser, nur auf einem Gerät zu testen, als den gesamten Testprozess wegzulassen.

### Testen von mehreren Geräten

Wie der Name schon sagt, konzentriert sich der Testumfang für mehrere Geräte entweder auf mehrere Geräte auf einer Mobilplattform oder auf mehreren Mobilplattformen. Wählen Sie die Plattformen und Testgeräte anhand Ihrer Zielgruppe aus und gruppieren Sie die Geräte wie in Abschnitt 3.2.1 beschrieben. Wenn Sie keine Informationen über Ihre Zielgruppe haben, benutzen Sie das Internet, um nach plattformspezifischen Daten und Statistiken zu suchen, die Ihnen helfen werden, die Mobilplattform und die Mobilgeräte auszuwählen, auf die Sie sich konzentrieren können.

Eine wirklich gute Webseite, bereitgestellt von Google, ist »Our Mobile Planet«<sup>1</sup>, auf der Sie Informationen basierend auf dem Land, Alter, Geschlecht, Nutzerverhalten und Verhalten im aktuellen Jahr bekommen.

### Testen der maximalen Anzahl an Geräten

Der Testumfang für die maximale Anzahl an Geräten umfasst so viele Mobilplattformen und Geräte wie möglich. Dieser Ansatz kann für Massenmarkt-Apps genutzt werden, um so viele Kunden wie möglich überall auf der Welt zu erreichen ohne Restriktionen hinsichtlich Plattform, Gerät, Mobilfunkanbieternetz

---

1. <http://think.withgoogle.com/mobileplanet/en/>

oder Zielgruppe. Eine App für den Massenmarkt zu testen ist sehr schwierig, da es fast immer eine Hard- und Softwarekombination gibt, die nicht gut mit Ihrer App arbeiten wird, und es beinahe unmöglich ist, diese Kombination zu finden. Um dieses Risiko zu reduzieren, müssen Sie einen Weg finden, auf so vielen Geräten wie möglich zu testen.

Dieser Ansatz benötigt eine Menge Recherchearbeit, um Informationen und Statistiken über die aktuelle Nutzung der Geräte, der Mobilplattformen und der Betriebssystemversionen zu sammeln. Es werden Informationen über die verschiedenen Mobilfunkanbieternetze und Verbindungsgeschwindigkeiten überall auf der Welt benötigt.

Wenn Sie erst einmal die erforderlichen Informationen gesammelt haben, überlegen Sie sich, eine Kombination aus internem Testen, Cloud-Testen und Crowd-Testen durchzuführen, um die Massenmarktsituation nachzustellen. Denken Sie daran, dass das Testen nur mit internen Ressourcen und Geräten vom Umfang her zu eingeschränkt oder zu teuer sein wird.

### Testen der Anwendungsfälle

Neben der Auswahl eines Testumfangs für die Hardware, um die Testarbeiten zu verringern, können Sie auch den Umfang der Anwendungsfälle auswählen, um die Arbeitsbelastung zu begrenzen. Mit diesem Ansatz können Sie sich auf bestimmte Teile oder Features der App konzentrieren und weniger wichtige auslassen, wie beispielsweise Hilfetexte oder Randfälle. Fast jedes Projekt steht unter extremem Zeitdruck, deswegen ist es wichtig, zu definieren, welche Teile der App im Testumfang einbezogen sein müssen und welche außen vor gelassen werden können. Schreiben Sie beide Teile in Ihrer Teststrategie auf und beschreiben Sie die Anwendungsfälle, die getestet werden müssen und warum. Wenn genug Zeit vorhanden ist, sollten die weniger wichtigen Teile ebenfalls getestet werden.

Sie sollten oder müssen die folgenden Informationen über den Testumfang in Ihrer Teststrategie mit einbeziehen:

- Welcher Ansatz sollte im Projekt genutzt werden?
- Ist es möglich, Ansätze für bestimmte Teile der App zu kombinieren?
- Warum wurde dieser Ansatz gewählt?
- Welches sind die benötigten und wichtigsten Hauptanwendungsfälle der App?

#### 7.1.3 Teststufen und Testtechniken definieren

Sobald Sie die Anforderungen und den Testumfang definiert haben, müssen Sie sich über die verschiedenen Teststufen und Testtechniken Gedanken machen, die Sie in Ihrem Projekt nutzen wollen. Beachten Sie die Übersicht der Qualitätssicherungsmaßnahmen aus Abschnitt 4.3, wenn Sie diese definieren. Dieser Überblick wird Ihnen helfen, die Teststufen und Testtechniken für Ihre App herzuleiten.

## Teststufen

Wie in Kapitel 5 »Mobile Testautomatisierungswerkzeuge« gezeigt, tendiert der Fokus von Teststufen von nicht mobiler Softwareentwicklung dazu, sich immer mehr in den mobilen Softwareentwicklungsprozess zu verschieben. Die »Testpyramide für Apps« zeigt, dass die Ebene Unit Test der kleinste Teil im Vergleich zu End-to-End-Tests, Betatests und manuellen Tests ist.

In den meisten Softwareentwicklungsprojekten ist der Entwickler für Unit Tests zuständig. Das ist auch in App-Projekten der Fall. App-Tester sind dafür verantwortlich, die End-to-End-Test-Automatisierung inklusive der Integrations-tests zu schreiben. Allerdings sollte jeder im Team für die Qualität der App verantwortlich sein – sie sollten alle den App-Tester bei ihrer oder seiner Arbeit unterstützen.

Wie schon in einigen Kapiteln dieses Buches erwähnt, ist manuelles Testen ein sehr wichtiger Teil eines mobilen Entwicklungsprojekts und bildet einen wesentlichen Teil der Teststufen in einem solchen Projekt. Allerdings gibt es auch andere Ebenen, die für manuelles Testen wichtig sind: Akzeptanz-, Alpha- und Betatests. Nutzerakzeptanztests können durchgeführt werden, um die App gegen die Nutzeranforderungen zu testen und um zu verifizieren, dass die App alle abdeckt. Dieser Schritt wird gewöhnlich von einem Tester, Produktmanager oder einem Kunden durchgeführt.

In mobilen Entwicklungsprojekten sind Alpha- und Betatests wichtige Teststufen, die einen Teil Ihrer Teststrategie bilden sollten. Jedes Mal, wenn ein Feature innerhalb Ihrer App implementiert wird, sollten Sie es mit potenziellen Kunden testen, um frühzeitiges Feedback darüber zu sammeln. Wenn der Test mit potenziellen Kunden innerhalb der Alphatests nicht möglich ist, könnten Sie versuchen, die Funktion mit Ihren Arbeitskollegen zu testen, um Feedback von außerhalb des Entwicklungsteams zu bekommen.

Sobald die App eine definierte Reife erreicht hat – zum Beispiel, wenn alle Features implementiert worden sind oder wenn nur zwei Fehler in der letzten Woche gefunden worden sind –, sollten Sie prüfen, ob Sie Beta-Distributionswerkzeuge benutzen oder einen Crowd-basierten Testansatz wählen, um Betatests mit potenziellen Kunden durchzuführen. Das Kriterium, das das Ende einer Testphase definiert und das anzeigt, dass die nächste Phase durchgeführt werden kann, muss außerdem in der Teststrategie dokumentiert werden.

Typische Softwareteststufen, die in einem mobilen Entwicklungsprojekt genutzt werden sollten, sind im Folgenden aufgelistet:

- Automatisiertes Testen
  - Unit Tests
  - End-to-End-Tests (inklusive Integrationstests)
- Manuelles Testen
  - Akzeptanztests
  - Alphatests
  - Betatests
- Regressionstests

Sobald Sie die Teststufen definiert haben, sollten Sie auch überlegen, wie intensiv jede Stufe von einem funktionalen und einem nicht funktionalen Gesichtspunkt aus getestet werden sollte. Denken Sie daran, dass nicht alle Testarten in jeder Teststufe einbezogen werden.

Außerdem kann es hilfreich sein, einige Metriken zu definieren, die den aktuellen Status der Applikation messen, wie z.B. folgende:

- Zu jedem Feature muss es mindestens einen Unit Test geben.
- Zu jedem Feature muss es mindestens einen End-to-End-Test geben.
- Es sollten keine Warnungen oder Fehler in der statischen Analyse des Codes vorhanden sein.

Funktionales Testen sollte das Folgende beinhalten:

- Die Funktionalität der App identifizieren.
- Die unterschiedlichen Teile gegen die funktionalen Anforderungen testen.
- Die Testfälle definieren und ausführen.
- Eingabedaten anhand der Spezifikation generieren.
- Die Ist-Ausgabe und die Soll-Ausgabe vergleichen.

Nicht funktionales Testen sollte die folgenden Punkte beinhalten:

- Stresstests
- Performanztests
- Benutzbarkeitstests
- Sicherheitstests
- Portabilitätstests
- Test zur Barrierefreiheit
- Internationalisierungs-/Lokalisierungstests



Die folgenden Informationen zu den Teststufen könnten in Ihrer Teststrategie genutzt werden:

- Welche Teststufen werden im Projekt genutzt und warum?
- Welche Teile werden für funktionales Testen und welche für nicht funktionales Testen verwendet?
- Definieren und Beschreiben der automatisierten Teststufen. Welche Teile müssen mit Unit Tests getestet werden und welche Teile werden getestet, indem man ein End-to-End-Test-Automatisierungswerkzeug benutzt?
- Definieren und Beschreiben, wann die Software eine bestimmte Reife erreicht hat und somit für Alpha- und Betatests genutzt werden kann.
- Definieren und Beschreiben von Metriken, die für das Projekt relevant sind.

### Testtechniken

Sie können auf Qualitätssicherungsmaßnahmen zurückgreifen (Kap. 4), um Ihre Testtechniken und Testmethoden zu definieren. Überlegen Sie sich, statische und dynamische Ansätze zu benutzen, um Ihre App aus unterschiedlicher Sicht zu testen.

Wie in Kapitel 4 beschrieben, empfehle ich, Werkzeuge zur statischen Codeanalyse in Ihrem statischen Testansatz zu benutzen, um den App-Code auf irgendwelche Fehler oder Probleme zu testen. Denken Sie daran, dass der App-Code während einer statischen Analyse nicht ausgeführt wird. Außerdem sollte die gesamte Projektdokumentation auf Vollständigkeit geprüft werden.

Im dynamischen Testansatz sollten Sie die Technik der White- und Blackbox-Tests benutzen, um Ihre App zu testen. Whitebox-Tests sollten von den Entwicklern durchgeführt werden und das Folgende abdecken:

- Anweisungsüberdeckung
- Pfadabdeckung
- Zweigüberdeckung
- Entscheidungsüberdeckung
- Kontrollflusstests
- Datenflusstests

Blackbox-Tests sollten von den Softwaretestern durchgeführt werden und das Folgende beinhalten:

- Äquivalenzklassenanalyse
- Grenzwertanalyse
- Entscheidungstabellen
- Zustandsübergangsanalyse
- Ursache-Wirkungs-Graph-Analyse

Allerdings sollten auch die Entwickler auf der Stufe von Unit Tests Grenzwerte und Zustandsübergänge testen, um sicherzustellen, dass jede Unit korrekt mit diesen Situationen umgeht.

In Ihrer Teststrategie sollte ebenfalls schriftlich dokumentiert sein, welche Testtechnik von wem angewandt wird.

Neben den bereits genannten Techniken sollten Sie exploratives Testen und risikobasiertes Testen in Betracht ziehen, um Ihre Testarbeiten innerhalb Ihres App-Teams zu organisieren und zu reduzieren.

**Wichtig:**

Definieren Sie Teststufen für Ihre App anhand Ihrer Features und Anforderungen. Qualitätssicherungsmaßnahmen werden Ihnen helfen, Ihre Testmethoden und Testtechniken zu definieren.

Die folgenden Informationen über Testtechniken könnten ein Teil Ihrer Teststrategie sein:

- Welche Testtechnik wird von Ihrem Projekt genutzt und warum?
- Definieren und beschreiben Sie die Reihenfolge der Testtechniken, zum Beispiel: Der statischen Codeanalyse und den Dokumentenreviews folgen Whitebox-Tests, Blackbox-Tests und dann explorative Testsessions, in denen das ganze Team mit einbezogen ist.
- Welche Teammitglieder wenden die verschiedenen Testtechniken an?
- Definieren und beschreiben Sie den manuellen Testprozess, wie zum Beispiel Akzeptanztests, exploratives Testen, Alpha- und Betatests.
- Definieren und beschreiben Sie das Testenkriterium für White- und Blackbox-Tests, zum Beispiel: 80 % Zweigüberdeckung mit Whitebox-Tests.

#### 7.1.4 Testdaten

Fast jeder App-Prozess erzeugt und sendet Daten von der App über Datennetze zu verschiedenen Backend-Systemen. Die verarbeiteten und übertragenen Daten unterscheiden sich von App zu App und haben verschiedene Anforderungen und Komplexitäten. Eine Komponente Ihrer mobilen Teststrategie sollten die benötigten Testdaten sein. Es ist wichtig, die Testdaten so realistisch wie möglich anhand von Features und Anforderungen Ihrer App zu definieren.

Im Folgenden sind Beispiele für verschiedene Testdatentypen aufgeführt:

■ **Konfigurationsdaten**

Diese Daten beinhalten Konfigurationen für die App oder die Backend-Systeme. Es können zum Beispiel Entscheidungsregeln für Rules Engines und/oder Einstellungen für Datenbanken und Firewalls enthalten sein.

■ **Stabile Daten**

Diese beinhalten gewöhnlich Daten mit einer geringen Veränderungsrate und einer langen Lebensdauer. Ein typisches Beispiel dafür wäre eine Kundeninformation, wie Nutzernamen und Passwörter, oder Produktinformationen.

■ **Temporäre Daten**

Dieser Datentyp wird sich eher häufig ändern, d.h., dass die Daten nur einmal benutzt werden können oder, während die App läuft, erstellt werden müssen, zum Beispiel: Bezahlungsdetails oder Buchungsbelege.

Sobald die Testdatenanforderungen klar sind, müssen Sie einen Weg festlegen, die Testdaten zu sichern, damit Sie die Daten, wann immer Sie sie brauchen, in einem definierten Zustand erneut erzeugen oder in einen solchen zurücksetzen können. Eine mögliche Lösung dafür ist eine Datenbank, aus der die gespeicherte Information während des Entwicklungs- und Testprozesses genutzt werden kann. Ein zusätzlicher Vorteil der Datenbank ist, dass Sie die Datenbank für den manuellen Test und den automatisierten Test gleichzeitig nutzen können. Auf der anderen Seite können Sie Testdatenmanagementwerkzeuge nutzen, um die Daten innerhalb Ihres Projektes zu organisieren.

Sobald Sie den Zurücksetzen- und Erzeugungsprozess definiert haben, sollten Sie die Daten so schnell wie möglich erstellen, weil das Ihnen und Ihren Kollegen während des App-Entwicklungsprozesses helfen wird.

Abhängig von der App werden Sie vielleicht eine Menge Testdaten benötigen. Wenn das der Fall ist, wäre ein Generator, der die Daten automatisch erzeugt, eine gute Idee. Wenn Sie einen Datengenerator nutzen, ist es wichtig, dass die funktionellen und erforderlichen Parameter dokumentiert werden.

Wenn die Testdaten in Ihrem Projekt verfügbar sind, dürfen Sie nicht vergessen, diese bei neuen Features und Änderungen anzupassen. Sehr wahrscheinlich werden sich mit der Zeit Features verbessern und die Testdatenanforderungen sich ändern. Ihre Teststrategie sollte deswegen einen Prozess für die Aktualisierung der Testdaten skizzieren, inklusive der Verantwortlichkeiten und der Trigger für den Aktualisierungsprozess. Es empfiehlt sich auch, dass Sie eine Strategie dafür definieren, wie veraltete Testdaten archiviert werden sollten, um Fehler zu reproduzieren, die noch in alten Features oder alten Versionen der App gefunden werden.

Die folgenden Testdateninformationen könnten ein Teil Ihrer Teststrategie sein:

- Wie werden die Testdaten generiert?
- Wo werden die Testdaten gespeichert?
- Wie wird die Dokumentation gehandhabt, zum Beispiel: Werden Testdaten zusammen mit den Testergebnissen dokumentiert?
- Wie oft werden die Testdaten aktualisiert?
- Wer ist verantwortlich für die Testdaten?

### 7.1.5 Auswahl der Zielgeräte und der Testumgebung

Nachdem Sie nun die Features, die Anforderungen, die Teststufen und Testtechniken, aber auch die Testdaten beschrieben haben, müssen Sie über die Testumgebung und die Zielgeräte für den Test in Ihrer Strategie nachdenken. Wie Sie in Kapitel 3 erfahren haben, ist es ein guter Ansatz, die Testgeräte oder die mobilen Webbrowser zu gruppieren, um zu bestimmen, welche Geräte für die App genutzt werden sollten. Die Erstellung solcher Mobilgerätegruppen erfordert Informationen über die Kundenzielgruppe und ihre Nutzerszenarien. Vergessen Sie dabei nicht den Testumfang, der in diesem Kapitel beschrieben wurde.

Sobald die Gerätegruppen festgelegt worden sind, müssen Sie Geräte aus diesen Gruppen wählen, um sie Ihrem Team bereitzustellen. Es wird empfohlen, dass Sie mindestens ein Gerät von jeder Gerätegruppe für den Test zur Verfügung haben. Allerdings empfehle ich Ihnen mindestens fünf Geräte aus jeder Gruppe, um einen breiteren Mix an Hard- und Softwarekombinationen mit verschiedenen Formen und Bildschirmen zu haben. Sie sollten auch schriftlich festhalten, warum Sie diese Testgeräte gewählt haben.

Nachdem Sie nun wissen, welche Geräte für das Testen benötigt werden, müssen Sie sie kaufen oder mieten. Alle erforderlichen Geräte zu kaufen kann sehr teuer werden und ist wegen des Projektbudgets vielleicht keine Option. Eine gute Möglichkeit, um ein wenig Geld zu sparen, sind Onlineauktionen, wo Sie gebrauchte Geräte kaufen können. In den meisten Fällen sind die Geräte gut genug erhalten für Ihre Tests.

Wenn Kaufen überhaupt keine Option ist, können Sie die Geräte mieten. Wie in Kapitel 3 erwähnt, gibt es einige Anbieter für Mobile Device Labs auf dem Markt, die Ihnen die Geräte, die Sie brauchen, für eine bestimmte Zeit ausleihen. Allerdings sollten Sie die Leihgebühren prüfen und mit dem Gerätepreis vergleichen. Wenn Sie die Geräte für einen längeren Zeitraum mieten wollen, kann mieten teurer sein, als die Geräte gleich zu kaufen.

Eine andere Alternative sind Open Device Labs<sup>2</sup>, wo die Geräte kostenlos geliehen werden können. Schauen Sie in die »Open Device Labs«-Karte, um eines in Ihrer Nähe zu finden. Wenn Sie ein kostenloses Gerät haben wollen, können

---

2. <http://opendevicelab.com/>

Sie auch in Ihrer Firma oder Familie herumfragen, ob jemand das Gerät hat, das Sie brauchen, und Sie es für eine Weile ausleihen können.

Sobald Sie die Gerätestrategie definiert haben, müssen Sie sich über die Testumgebung, d.h. die Backend-Systeme, Gedanken machen. Sie müssen die Architektur der Backend-Systeme, wie Datenbanksysteme, Bezahlssysteme, APIs und alle anderen Arten von Systemen, die in Ihr Projekt eingebunden sind, kennen. Wenn Sie Systeminformationen haben, müssen Sie sichergehen, dass diese auch in der Testumgebung erreichbar sind, sodass Sie so testen können, als wären Sie in der Produktivumgebung. Wenn Ihre Teststrategie Testen in freier Wildbahn oder Tests von unterwegs enthält, gibt es eine weitere Anforderung: Die Testumgebung muss außerhalb Ihres Firmennetzwerks erreichbar sein.

Die folgenden Informationen über Zielgeräte und die Testumgebung könnten Teil Ihrer Teststrategie sein:

- Welche Geräte werden für das Testen genutzt?
- Warum werden diese Geräte für das Testen genutzt?
- Sind die Geräte in der Firma vorhanden oder müssen diese erworben werden?
- Was sind Ihre Gründe für die Wahl dieser Testgeräte?
- Welche Anforderungen an die Testgeräte und die Testumgebung gibt es?
- Gibt es eine Aktualisierungsrichtlinie für die Mobilgeräte?
- Wann werden neue Geräte in den Entwicklungs- und Testprozess integriert?
- Was sind die Nutzungsszenarien des Systems?
- Woraus besteht das Backend-System?
- Ist die Testumgebung der Live-Umgebung ähnlich?
- Kann die Testumgebung für Testzwecke außerhalb des Firmennetzwerkes genutzt werden?

### 7.1.6 Manuelles und »In the Wild«-Testen bzw. Tests in freier Wildbahn

Wie Sie bisher erfahren haben, erfordert App-Testen viel manuelles Testen und Testen in realen Umgebungen. Denken Sie an das Beispiel mit der Ski- und Snowboard-App aus Kapitel 1, bei dem das Testen auf dem Berg erforderlich war, um zu sehen, ob die App wirklich unter echten Bedingungen funktioniert.

Manuelles Testen in freier Wildbahn ist essenziell für Ihre App und erfordert zuvor eine Menge Planung, um nutzlose Testszenarien während der Tests von unterwegs zu vermeiden. Deswegen sollten Sie übliche Szenarien aus der realen Welt für Ihre App und ihre Features identifizieren. Schreiben Sie diese Szenarien in Ihrer Teststrategie auf und ordnen Sie sie nach Priorität und Wichtigkeit Ihres Projektes.

Außerdem wird empfohlen, dass Sie Testrouten, wie beispielsweise mit dem Bus, Zug, Auto, Flugzeug oder beim Laufen, definieren. Innerhalb dieser Routen beschreiben Sie mögliche Szenarien, die getestet werden sollten. Diese Routen

erlauben Ihnen, echte Mobilgerätenutzer zu simulieren, während sie zur Arbeit pendeln oder herumreisen.

Nicht zuletzt sollten Sie Datennetzscenarien anhand Ihrer Zielgruppe definieren. Nachdem Sie Informationen über Ihre Zielgruppe gesammelt haben, werden Sie wissen, in welchen Regionen sie leben und welche Datennetze und Geschwindigkeiten dort verfügbar sind, zum Beispiel: 4G, 3G oder EDGE. Aufgrund dieser Information können Sie den App-Test auf diese Netzwerkgeschwindigkeiten begrenzen, aber vergessen Sie nicht, verschiedene Netzanbieter zu testen.

Im Folgenden sind einige Beispiele für Nutzungsszenarien aufgelistet:

- Testen Sie Ihre App basierend auf den Features in sonniger Lage im Freien oder innerhalb des Büros.
- Testen Sie, ob die App bei unterschiedlichen Wetterbedingungen genutzt werden kann.
- Nutzen Sie mehrere Apps gleichzeitig, wie E-Mail, Chat und News, während Ihre App im Hintergrund läuft. Prüfen Sie, ob die App von anderen Apps beeinflusst wird.

Im Folgenden sind einige Beispiele für Routenszenarien aufgelistet:

- Nutzen Sie die App, während Sie mit dem Zug, Bus oder Auto zur Arbeit pendeln.
- Nutzen Sie die App beim Joggen und prüfen Sie das Verhalten der Sensoren.
- Nutzen Sie die App während eines Spaziergangs durch eine Stadt oder die Landschaft und prüfen Sie die Sensoren und das GPS oder den Kompass.

Im Folgenden sind einige Beispiele für Datennetzscenarien aufgelistet:

- Testen Sie, wie die App in schnellen Datennetzen wie 4G oder 3G funktioniert.
- Testen Sie, wie die App mit einem Wechsel des Datennetzes von 4G zu 3G oder sogar 2G umgeht.
- Testen Sie, wie die App den Verlust von Paketen verkraftet oder mit einem Netzausfall umgeht.

Das Testen in freier Wildbahn verlangt eine Menge Bewegung und ist eine herausfordernde Aufgabe, die während des Entwicklungsprozesses der App durchgeführt werden muss. Wenn Sie nicht die Zeit oder Möglichkeit haben, die App in realen Szenarien zu testen, überlegen Sie, Crowd-Tests zusammen mit den internen Tests zu nutzen. Denken Sie dabei aber an die Vor- und Nachteile von Crowd-Tests aus Kapitel 6, da sie einen Einfluss auf Ihre Projektplanung, Koordination, Zeitplanung und Budget haben können.

Schreiben Sie keine komplexen Testfälle und Szenarien mit exakten Schritten für Tests in freier Wildbahn. Wenn Sie unterwegs sind, werden Sie wahrscheinlich keinen Laptop dabei haben, um sich die Testfälle anzusehen. Das wäre auch ineffizient, weil es das wirkliche Testscenario, das Verhalten und die Nutzersimulation zerstört. Sie werden wahrscheinlich eine Tasche mit vielen Geräten für das

Testen in freier Wildbahn dabei haben. Wenn Sie solche Szenarien planen, halten Sie sie kurz und informativ, sodass sie von unterwegs aus genutzt werden können. Ich empfehle Ihnen, die Szenarien auszudrucken oder sie auf Papier aufzuschreiben, sodass Sie sie lesen und im Kopf behalten können.

Die folgenden Informationen über das Testen in freier Wildbahn könnten Teil Ihrer Teststrategie sein:

- Definieren und beschreiben Sie die Nutzungsszenarien von Ihren Kunden.
- Definieren und beschreiben Sie die Testszenarios in freier Wildbahn.
- Definieren und beschreiben Sie die verschiedenen Datennetze, die fürs Testen genutzt werden müssen.

### 7.1.7 Checklisten und Testtouren

Sie können Checklisten und Testtouren zu Ihrer Teststrategie hinzufügen. Wie in Kapitel 4 beschrieben, können Checklisten sehr hilfreich für Ihre App sein, da Sie damit Punkte festhalten, die nicht automatisiert werden können oder die sich häufig ändern. Wenn Sie die Anforderungen und Features Ihrer App kennen, werden Sie wahrscheinlich auch wissen, welche Teile der App wiederholt getestet werden müssen. In diesem Fall ist es gut, diese Features zu einer Checkliste als Teil Ihrer Teststrategie hinzuzufügen.

Anhand der Features Ihrer App ist es nützlich, Testtouren zu definieren, um die Testaufwände auf spezielle Teile der App zu konzentrieren. Cem Kaner<sup>3</sup> beschreibt eine Tour als »... eine Exploration eines Produkts, die um ein bestimmtes Thema herum organisiert ist«.

Die Einbindung von Testtouren in Ihren Testarbeiten hilft Ihnen, herauszufinden und zu verstehen, wie die App funktioniert. Außerdem können Sie dabei neue Testideen entwickeln, während Sie die App testen. Bitte sehen Sie in Kapitel 4 nach einer Beschreibung für einige Touren, die durch Merkhilfen gestützt werden.

Im Folgenden finden Sie einige Beispiele für Testtouren:

- **Feature-Tour**  
Erkunden und testen Sie alle möglichen Features in der App.
- **Konfigurationstour**  
Erkunden und testen Sie alle Teile der App, die konfiguriert werden können.
- **Gestentour**  
Nutzen Sie jede mögliche Geste auf jedem Bildschirm, um zu sehen, wie die App mit den verschiedenen Eingaben umgeht.
- **Orientierungstour**  
Verändern Sie die Orientierung von jedem Bildschirm von Hochformat zu Querformat und andersherum, um zu sehen, ob irgendwelche Problem auftreten.

---

3. <http://kaner.com/?p=96>

Ich benutze die folgenden Merkhilfen in meinen Projekten:

- FCC CUTS VIDS  
(<http://michaeldkelly.com/blog/2005/9/20/touringheuristic.html>)  
von Michael Kelly
- I SLICED UP FUN  
([www.kohl.ca/articles/ISLICEDUPFUN.pdf](http://www.kohl.ca/articles/ISLICEDUPFUN.pdf))  
von Jonathan Kohl

Die folgenden Informationen über Checklisten könnten Teil Ihrer Teststrategie sein:

- Welche Checklisten werden im Projekt genutzt und warum?
- Beschreiben Sie die verwendeten Checklisten und Touren.
- Beschreiben Sie, wann die Checklisten und Touren genutzt werden sollten und von wem.

### 7.1.8 Testautomatisierung

Testautomatisierung kann auch ein Teil einer mobilen Teststrategie sein. Abhängig von der App und ihrem Lebenszyklus, müssen Sie sie vielleicht nicht automatisieren. Wenn das der Fall ist, ist es wichtig, dass Sie die Gründe, warum die Testautomatisierung nicht notwendig ist, dokumentieren und beschreiben.

Wenn Ihre App aber Testautomatisierung benötigt, sollten Sie sich so schnell wie möglich über Automatisierung und die Werkzeuge, die Sie benutzen wollen, Gedanken machen. Denken Sie zurück an die verschiedenen Testautomatisierungskonzepte aus Kapitel 5 und deren Vor- und Nachteile. Wählen Sie das Werkzeug, das am besten in Ihre jetzige Projektsituation passt und mit dem Sie bereits einige Erfahrung oder Programmierkenntnisse haben sammeln können, oder für das Sie einen entsprechenden Testaufbau oder eine Testumgebung zur Hand haben. Das wird Ihnen eine Menge Geld und Zeit sparen.

Beschreiben Sie das Testautomatisierungswerkzeug in Ihrer Teststrategie und begründen Sie, warum Sie es für dieses Projekt ausgewählt haben. Anhand der beschriebenen Features und Anforderungen können Sie die Teile der App definieren, die automatisiert und die nicht automatisiert werden sollten.

Im nächsten Schritt sollten Sie die Geräte beschreiben, auf denen die automatisierten Tests laufen sollen, und in welchen Umgebungen die Tests ausgeführt werden sollten.

Ihr App-Projekt sollte die richtige Balance zwischen physischen und virtuellen Geräten finden, falls Sie nicht die Möglichkeit haben, alles auf physischen Geräten zu testen. Ein Mix aus physischen und virtuellen Geräten kann auch kostengünstiger sein.

Sobald Sie die Geräte und die Testautomatisierungsumgebung definiert haben, empfehle ich Ihnen, auch Testsuiten zu definieren, in denen die verschiedenen automatisierten Testfälle basierend auf den üblichen Features, Abschnitten und Anforderungen innerhalb der App gruppiert werden. Mithilfe von Testsuiten



können Sie entscheiden, welche Tests wann und wie oft ausgeführt werden sollten. Zum Beispiel können Sie eine Smoke-Testsuite mit einigen automatisierten Tests aus jedem Teil der App definieren, um sicherzustellen, dass ein Commit in das zentrale Code-Repository nichts kaputt gemacht hat. Diese Testsuite sollte klein sein und immer dann laufen, wenn Codeänderungen stattgefunden haben, um schnelles Feedback zu bekommen und die Entwickler zügig über irgendwelche Probleme informieren zu können.

Im Folgenden finden Sie einige Beispiele von Testsuiten:

- Smoke-Testsuite mit wenigen Testskripten, um die Basisfunktionen der App zu prüfen und schnelles Feedback zur Verfügung zu stellen
- Medium-Testsuite oder eine Testsuite nur mit spezifischer Funktionalität
- Komplette Regressionstestsuite mit allen Testskripten, die einmal am Tag oder in der Nacht ausgeführt werden
- Testsuite nur mit Nutzerszenarien wie dem Registrierungs- oder Abmeldeprozess

Testsuiten sind eine gute Möglichkeit, um die Balance zwischen schnellem Feedback und umfangreicher Testabdeckung zu wahren.

Nachdem die Testsuiten erstellt sind, legen Sie fest, wann sie ausgeführt werden sollen, zum Beispiel: Die Smoke-Testsuite muss nach jedem Commit durch den Entwickler ausgeführt werden. Die Medium-Testsuite könnte alle zwei Stunden ausgeführt werden. Die komplette Regressionstestsuite könnte einmal in der Nacht ausgeführt werden.

Sie sollten auch definieren, wo die Testautomatisierung ausgeführt werden sollte. Sie kann auf der lokalen Umgebung des Entwicklers erfolgen, wo die Unit Tests ausgeführt werden, bevor Änderungen zum zentralen Repository übertragen werden. Die Unit Tests und End-to-End-Tests können auf dem CI-Server ausgeführt werden.

Die folgenden Informationen über Testautomatisierung könnten Teil Ihrer Teststrategie sein:

- Wird ein CI-Server für den Build-Prozess genutzt?
- Welcher CI-Server wird genutzt und warum?
- Welche Testumgebung wird genutzt, um die Testautomatisierung auszuführen?
- Welche Testautomatisierungswerkzeuge werden im Projekt genutzt?
- Werden die Tests auf echten und auf virtuellen Geräten ausgeführt?
- Welche Geräte müssen mit dem CI-Server verbunden sein und warum?
- Wird ein Cloud-Anbieter für die Testautomatisierung genutzt?
- Definieren und beschreiben Sie die Testsuiten und Gruppen.
- Definieren und beschreiben Sie die verschiedenen Build-Trigger und Durchführungszeiten.
- Definieren und beschreiben Sie, wo die verschiedenen Testautomatisierungssuiten ausgeführt werden, zum Beispiel: alles auf der lokalen Entwicklermaschine oder auf dem CI-Server.

### 7.1.9 Produktrisiken

Jedes Projekt ist verschiedenen Arten von Risiken ausgesetzt. Es ist wichtig, sowohl das Projektrisiko als auch das Feature-Risiko herauszuarbeiten, damit Sie dann mögliche Lösungen für diese definieren können. Betrachten Sie sowohl die Eintrittswahrscheinlichkeit des Risikos als auch die Auswirkung des Risikos. Wenn die Produktrisiken klar sind, können Sie damit beginnen, einen Risikoanalyseansatz zu implementieren, der vom gesamten Team genutzt wird, wenn neue Features definiert, implementiert und getestet werden.

Die folgenden Informationen über Produktrisiken könnten Teil Ihrer Teststrategie sein:

- Welche Teile sind kritisch für den Betrieb?
- Wie hoch ist die Eintrittswahrscheinlichkeit des Risikos?
- Wie sollen die betriebskritischen Teile getestet werden?
- Was ist der potenzielle Schaden, wenn ein kritisches Problem auftaucht?
- Wie wird die Feature-Risikoanalyse durchgeführt?
- Gibt es einen Katastrophenplan?

**Wichtig:**

Eine mobile Teststrategie zu erstellen ist nicht einfach, da sie viele Testinformationen zu App und Gerät abdecken muss. Ihre Strategie muss eventuell auch während des Entwicklungsprozesses wegen geänderter Produktfeatures oder Prioritäten angepasst werden.

## 7.2 Veröffentlichungsstrategie für Apps

So wichtig wie die App-Teststrategie ist, ist es genauso wichtig, eine Veröffentlichungsstrategie für Apps aufzuschreiben. Die Markteinführung einer App ist nicht einfach und es können eine Menge Probleme auftreten, während und nachdem sie in den Markt eingeführt wurde. Dieser Teil des Kapitels wird die wichtigen Aktivitäten vor und nach der Markteinführung für eine App beschreiben.

### 7.2.1 Vor der Markteinführung – Untersuchung des Releasematerials

Wie in Kapitel 4 erwähnt, sollten Sie darüber nachdenken, eine Release-Checkliste zusammenzustellen, um sicherzugehen, dass die kompletten Releaseinformationen vorhanden und verfügbar sind. Sie sollten außerdem den Aktualisierungs- und den Installationstest durchführen, bevor Sie die App in den App-Store übergeben.

Die Tests, die Sie durchführen, bevor Sie die App im App-Store einreichen, sollten sich nicht nur auf die App selbst beschränken, sondern Ihre Markteinführungsstrategie sollte auch skizzieren, ob und wie der Backend-Service vor einem neuen Release getestet wurde.

Stellen Sie sich selbst die Frage: »Benötigt die neue Version der App einige neue Backend-Services oder API-Aufrufe?« Wenn ja, sind diese Services oder API-Aufrufe bereits im Backend-System veröffentlicht worden? Wenn das noch nicht geschehen ist, ist es sehr wahrscheinlich, dass Ihre App vom Hersteller des App-Stores zurückgewiesen wird. Überprüfen Sie die neuen und bestehenden Features der App in der Produktivumgebung ein letztes Mal, wenn die Backend-Services erreichbar sind.

Wenn die App für die Markteinführung bereit ist, prüfen Sie die Release Notes und Feature-Beschreibungen für die neue App im App-Store. Gehen Sie die Texte durch und vergleichen Sie diese mit den neuen und existierenden Features. In den Release Notes ist es wichtig, konkret und gut verständlich die neuen Features zu beschreiben und erklären Sie diese den Nutzern. Es ist außerdem wichtig, die Release Notes und App-Beschreibungen in jeder unterstützten Sprache anzubieten.

Vergessen Sie nicht, die Screenshots der App zu prüfen. Diese sollten in der gleichen Sprache verfasst sein wie die Release Notes, die gleiche Größe haben wie die Screenshots zuvor und in der Statusbar die gleichen Symbole enthalten, die zum Beispiel die Zeit, den Batterie- und den Netzstatus anzeigen. Abbildung 7-1 und 7-2 zeigen Ihnen, was Sie nicht tun sollten: Die Statusbar enthält unterschiedliche Symbole für die gleiche App.

Ein professionelles Vorgehen beinhaltet, dass die App-Store-Screenshots die gleiche Statusbar und das gleiche Look-and-Feel haben müssen.

Wenn die neuen Features durch ein Video beschrieben werden, sehen Sie sich das Video erneut an und hören Sie sich die Beschreibungen der Features und die mitgelieferten Informationen an. Fragen Sie, ob es weiteres Marketingmaterial gibt, das geprüft werden muss.

Last, but not least gibt es einen enorm wichtigen Punkt: Veröffentlichen Sie die App an Ihre Kunden nicht am Freitagnachmittag oder kurz bevor Sie das Büro verlassen. In den meisten App-Stores dauert es eine Weile (bis zu mehreren Stunden), bevor Ihre App gelistet wird und als Download zur Verfügung steht. Wenn das Büro leer ist, kann niemand auf kritische Fehler oder Releaseprobleme reagieren, die vielleicht direkt nach dem Release auftauchen.

Manche App-Stores haben keinen Einreichungsprozess und Sie können deswegen schnell auf die Probleme reagieren, indem Sie eine Hot-Fix-Version Ihrer App direkt nach dem letzten Release veröffentlichen.



**Abb. 7-1** App-Store-Screenshot



**Abb. 7-2** App-Store-Screenshot der gleichen App mit unterschiedlichen Statusbar-Informationen und Größen

Wenn möglich, veröffentlichen Sie Ihre App entweder morgens oder am Wochenanfang, damit Sie einige Zeit haben, auf kritische Probleme zu reagieren.

Die folgenden Informationen über das Releasematerial könnten Teil Ihrer Veröffentlichungsstrategie sein:

- Definieren und benutzen Sie eine Release-Checkliste.
- Führen Sie den Aktualisierungs- und Installationstest erneut aus.
- Stellen Sie sicher, dass alle Backend-Services und APIs in der Produktivumgebung erreichbar sind.
- Definieren Sie, welche neuen und alten Features in der Live-Umgebung erneut überprüft werden müssen.
- Definieren und beschreiben Sie, wie das App-Store-Material inklusive der Release Notes, Screenshots und Videos geprüft werden soll.
- Definieren und beschreiben Sie, wann die App im App-Store eingereicht werden sollte.

### 7.2.2 Post-Release – Was passiert nach der Markteinführung der App?

Die Post-Release-Phase startet, sobald Sie Ihre App veröffentlicht haben. Während dieser Phase sollten Sie und Ihr Team einige Punkte beachten, um Feedback von Ihren Kunden zu bekommen und um irgendwelche Fragen oder Probleme zu behandeln, die sie Ihnen senden.

Das Erste, was Sie direkt nach der Veröffentlichung Ihrer App machen sollten, ist, sie aus dem App-Store herunterzuladen und zu installieren, um sicherzustellen, dass sie wie erwartet funktioniert. Wenn die Version okay ist, sollten Sie sie auf einem Dateiserver archivieren, um später in der Lage zu sein, sie erneut zu installieren, zum Beispiel um Probleme oder Fehler zu reproduzieren, die vielleicht gemeldet werden. Dieser Schritt kann auch von Ihrer Build-Pipeline innerhalb des Continuous Integration System durchgeführt werden.

Allerdings gibt es ein paar weitere Themen, um die Sie sich, Ihr Team und Ihre Firma kümmern müssen. Die folgenden Abschnitte beschreiben die Aktivitäten, die nach der Veröffentlichung Ihrer App durchgeführt werden können, um zusätzliche Informationen über Ihre Kunden und irgendwelche potenziellen Probleme zu erhalten. Es ist wichtig, diese Punkte in Ihrer Markteinführungsstrategie für Ihre App zu definieren und aufzuschreiben.

### 7.2.3 Unterstützung der Community

Das Anbieten von Community- oder Kundensupport für ein veröffentlichtes Produkt ist essenziell, insbesondere wenn Sie eine kostenpflichtige App anbieten. Wann immer Nutzer ein Problem haben, sollten sie in der Lage sein, jemanden in Ihrer Firma oder im App-Team zu kontaktieren, um Antworten auf Fragen oder Probleme zu bekommen. Wenn niemand die Nutzer oder Communitys betreut, werden sie schlechte Reviews schreiben oder einfach aufhören, Ihre App zu nutzen.

Sie sollten deshalb dafür sorgen, jeden möglichen Social-Media-Kanal zu überwachen und auf Kundenfeedback, Kundenfragen oder Probleme, die untersucht werden müssen, zu überprüfen. Es ist wichtig, diese Anfragen zu beantworten, um den Kunden das Gefühl zu geben, dass ihnen jemand zuhört. Es ist auch möglich, manche Nutzer auszuwählen und ihnen Fragen über die neue Version der App zu stellen. Das gesammelte Feedback kann dann dazu verwendet werden, die App in zukünftigen Releases zu verbessern.

**Wichtig:**

Wenn Ihre Firma eine Abteilung für Kundensupport hat, empfehle ich, dass Sie ein wenig Zeit mit den Kollegen aus dieser Abteilung verbringen, um ein Gefühl für die Kundenbedürfnisse und Probleme zu bekommen.

### 7.2.4 Reviews

Der nächste Kanal, den Sie nach der Veröffentlichung der App beobachten müssen, sind die App-Store-Reviews. Lesen Sie sie sorgfältig, um Feedback und Fehlerberichte von Ihren Nutzern zu bekommen. Allerdings empfehle ich Ihnen dringend, dass Sie diese Reviews vorsichtig behandeln. Es gibt viele Leute da draußen, die gerne negatives Feedback über Apps schreiben, das nicht zutrifft. Immer wenn sich ein Nutzer über Ihre App beschwert, versuchen Sie das Problem zu reproduzieren, um einen Fehlerbericht zu erstellen mit dem Ziel, dieses Problem in einem folgenden Release zu beheben. Wenn Sie nicht in der Lage sind, dieses Verhalten zu reproduzieren, versuchen Sie dem Reviewer zu antworten und konkrete Fragen zu stellen, um weitere Informationen zu bekommen.

Allerdings bieten nicht alle App-Stores eine Antwortfunktion innerhalb des Reviewbereichs, um mit den Kunden zu sprechen. Wenn der App-Store ein solches Feature nicht anbietet, können Sie Ihr eigenes Review schreiben und darin erklären, dass Sie der Entwickler, Tester oder Produktmanager der App sind und gerne zusätzliche Informationen haben würden, oder Sie beschreiben irgendeine Art Lösung für das Problem. Sie müssen aber sicher sein, dass Sie Reviews innerhalb des App-Stores schreiben dürfen, um auf vorherige Reviews zu antworten. Hier verweise ich auf die Richtlinien zu App-Store-Reviews. Lesen Sie diese, um sicherzugehen, dass Sie keine Regeln verletzen. Wenn Sie in der Lage sind, auf

Reviews zu antworten, benutzen Sie diese Funktionalität, um mit Ihren Kunden zu interagieren und von Ihnen zu lernen.

Wenn Sie viel negatives Feedback aufgrund von Missverständnissen bezüglich der App oder ihrer Features bekommen, können Sie auch eine Art Troubleshooting Guide oder ein Tutorial innerhalb der App-Beschreibung anbieten. Sollten Sie allerdings sehr viel negatives Feedback über Ihre App bekommen, weil die Kunden die Features nicht verstehen, müssen Sie die ganze App oder das Feature überdenken, zusätzliche Benutzbarkeitstests ausführen und sobald wie möglich eine Aktualisierung anbieten.

### 7.2.5 Absturzberichte

Eine andere wertvolle Quelle für Informationen sind die Absturzberichte Ihrer App. Wenn Sie Werkzeuge wie HockeyApp<sup>4</sup>, crashlytics<sup>5</sup> oder TestFlight<sup>6</sup> benutzen, sollten Sie diese nach der Veröffentlichung überprüfen, um zu sehen, ob es dort irgendwelche Probleme mit Ihrer App gibt. Nicht jedes während der Laufzeit auftauchende Problem wird in einem Absturz der App enden. Wenn Ihre App ein gutes Exception Handling eingebaut hat, werden die Fehler dem Nutzer nicht gezeigt, sondern von den Werkzeugen und deren Absturzberichteroutinen erfasst. Diese Art von Informationen ist sehr wichtig, um Ihnen zu helfen, die App in zukünftigen Releases zu verbessern.

Die Werkzeuge bieten eine Webschnittstelle, auf der die meisten App-Abstürze aufgereiht, gruppiert und kategorisiert sind. Sie zeigen die absolute Anzahl an Abstürzen und die Anzahl der Nutzer, die von diesem Problem betroffen sind. Fast jedes Crash Reporting Werkzeug bietet schöne Grafiken, die zum Beispiel die App-Verteilung und App-Abstürze über die Zeit darstellen. Manche dieser Crash Reporting Werkzeuge bieten eine Möglichkeit, Feedback aus der App heraus zum Crash Reporting Backend zu senden. Außerdem bieten manche dieser Werkzeuge eine Integration zu Fehlerverfolgungswerkzeugen von Dritten.

Wenn Sie kein Crash Reporting Tool eingeführt haben, könnte es helfen, dass manche App-Store-Anbieter eine elementare Funktion für Absturzberichte zur Verfügung stellen, die als Ausgangspunkt genutzt werden kann.

#### **Wichtig:**

Führen Sie ein Crash Reporting Tool ein, da es Ihnen und Ihrem Team helfen wird, detailliertere Informationen zu den Problemen und Abstürzen innerhalb Ihrer App zu erhalten.

---

4. <http://hockeyapp.net/features/>

5. <http://try.crashlytics.com/>

6. [www.testflightapp.com](http://www.testflightapp.com)

### 7.2.6 Tracking und Statistiken

Um Informationen über Ihre Kunden und deren Nutzung Ihrer App zu sammeln, sollten Sie irgendeine Art Tracking-Mechanismus implementieren, um wichtige Daten zu sammeln. Diese Art der Information wird von Tracking-Werkzeugen zusammengefasst, um Statistiken über Ihre App und die Nutzung von Features zu erstellen. Wenn Ihre App einen Tracking-Mechanismus benutzt, überprüfen Sie diese Statistiken nach der Veröffentlichung.

Abhängig von der Tracking-Implementierung können Sie Informationen wie diese bekommen:

- Mobilbetriebssystemversion
- Mobilgerätehersteller
- Gerätemodell
- Bildschirmgröße
- Mobilbrowserversion
- Anzahl der Seitenaufrufe
- Wie oft ein bestimmtes Feature aufgerufen wurde
- Wie oft der Registrierungsprozess abgebrochen wurde

Versuchen Sie mithilfe dieser Statistiken und Zahlen das Verhalten Ihrer Nutzer zu verstehen, damit Sie die App und ihre Features verbessern können. Die folgende Liste enthält einige Mobile Tracking Tools:

- adjust  
([www.adjust.com/](http://www.adjust.com/))
- appsfire  
(<http://appsfire.com/>)
- AppsFlyer  
([www.appsflyer.com/](http://www.appsflyer.com/))
- Clicktale  
([www.clicktale.com/](http://www.clicktale.com/))
- iMobiTrax  
([www.imobitrax.com/](http://www.imobitrax.com/))
- MobileAppTracking  
([www.mobileapptracking.com/](http://www.mobileapptracking.com/))

Wie Sie sehen können, ist es nicht einfach, Test- und Markteinführungsstrategien für Apps zu erstellen, da beide viel Informationen über Testthemen und Pre- und Post-Launch-Aktivitäten benötigen und enthalten. Bitte denken Sie daran, dass solche Strategien nicht in Stein gemeißelt sind. Sie sollten beide Strategien überarbeiten und anpassen, wann immer Änderungen in Bezug auf Produkt, Risiko oder irgendwelche anderen Prioritäten eintreten. Dokumente zu Test- und Markt-

Einführungsstrategien für Apps unterliegen einem fortlaufenden Prozess und jedes Teammitglied sollte verantwortlich für die Aktualisierung und Ergänzung sein.

### 7.3 Zusammenfassung

Das Hauptthema von Kapitel 7 waren die Test- und Markteinführungsstrategien für Apps. Jedes App-Team benötigt Test- und Markteinführungsstrategien, um an wichtige Aufgaben vor und nach der Veröffentlichung der App zu denken.

Wenn Sie eine mobile Teststrategie erstellen, sollten die folgenden Themen abgedeckt und definiert sein:

- Anforderungsdefinitionen
- Testumfang
- Teststufen
- Testtechniken
- Testdaten
- Zielgeräte und Zielumgebung
- Testautomatisierung

Mit den im ersten Teil des Kapitels vorgestellten Fragen sind Sie nun in der Lage, Ihre eigene Teststrategie für Ihre App oder Ihr Unternehmen aufzustellen.

Der andere Teil dieses Kapitels deckte die Mobilmarkteinführungsstrategie ab. Die Wichtigkeit des Releasematerials, inklusive der Feature-Beschreibungen, Screenshots und irgendwelcher anderer Marketingmaterialien, wurde beschrieben. Mithilfe der vorgestellten Fragen sollte es für Sie sehr einfach sein, zu prüfen, ob alles für die Veröffentlichung vorhanden ist.

Nach der Veröffentlichung der App ist ein Community-Support sehr wichtig, bei dem Nutzer Fragen stellen können, wenn sie irgendeine Art Problem mit der App haben. Außerdem wurde die Bedeutsamkeit von Absturzberichten, Tracking und Nutzerstatistiken begründet.