Certified Tester Specialist

ISTQB® Mobile Application Testing Foundation Level Lehrplan

Version 2019

Zur Verfügung gestellt von International Software Quality Institute (iSQI)

Deutschsprachige Ausgabe
Herausgegeben durch das German Testing Board e.V.

Übersetzung des englischsprachigen Lehrplans des International Software Testing Qualifications Board (ISTQB®), Version 2019.

Urheberrecht der Autoren der englischen Originalausgabe von iSQI, Certified Mobile Application Professional – Foundation Level (CMAP-FL) – Jose Diaz, Rahul Verma, Tarun Banga, Vipul Kocher und Yaron Tsubery – haben das Urheberrecht auf das ISTQB® übertragen. Dieser Lehrplan diente als Grundlage für die Erstellung des vorliegenden Dokuments.

Urheberrecht der Autoren der englischen Ausgabe des ISTQB®, Certified Tester Specialist, Mobile Application Testing, Foundation Level Syllabus: Vipul Kocher (Leitung), Piotr Wicherski (stellvertretende Leitung), José Díaz, Matthias Hamburg, Eran Kinsbruner, Björn Lemke, Samuel Ouko, Ralf Pichler, Nils Röttger, Angelina Samaroo, Yaron Tsubery.

Urheberrecht der vorliegenden deutschen Übersetzung: Mitglieder der Arbeitsgruppe des German Testing Board e.V. – Matthias Hamburg (Leitung), Jürgen Beniermann, Florian Fieber, François Martin, Ralf Pichler, Nils Röttger.

Änderungsübersicht

Version	Datum	Bemerkung
ISTQB Alpha	11. Mai 2018	Alpha-Version
ISTQB Beta	27. Januar 2019	Beta-Version
ISTQB GA	28. März 2019	GA-Fassung (für die Generalversammlung)
ISTQB V2019	10. Mai 2019	Durch ISTQB® freigegebene Version
GTB V2019	01. Sep. 2019	Durch GTB® freigegebene deutschsprachige Version

Inhaltsverzeichnis

Änderungsübersicht 3
Inhaltsverzeichnis 4
Dank 7
0. Einführung 8
0.1 Zweck dieses Dokuments 8
0.2 Certified Foundation Level Mobile Application Testing 8
0.3 Geschäftlicher Nutzen 8
0.4 Prüfbare Lernziele 9
0.5 Praktische Kompetenzstufen 9
0.6 Prüfung 10
0.7 Empfohlene Schulungszeiten 10
0.8 Voraussetzung für die Prüfung 10
0.9 Informationsquellen 10
0.10 Geschlechtsneutrale Formulierungen 10
1. Mobilgerätewelt – Geschäftliche und technologische Treiber – 175 Minuten 11
1.1 Analyse mobiler Nutzungsdaten 12
1.2 Geschäftsmodelle für mobile Apps 12
1.3 Mobile Gerätetypen 13
1.4 Arten mobiler Applikationen 13
1.5 Mobile Applikationsarchitektur 15
1.6 Teststrategie für mobile Apps 16
1.7 Herausforderungen beim Testen mobiler Applikationen 18
1.8 Risiken beim Testen mobiler Applikationen 19
2. Testarten für mobile Applikationen – 265 Minuten 20
2.1 Testen der Kompatibilität mit der Gerätehardware 21
2.1.1 Testen von Gerätefunktionen 21
2.1.2 Testen unterschiedlicher Gerätedisplays 22
2.1.3 Testen der Gerätetemperatur 22
2.1.4 Testen der Geräteeingabesensoren 23
2.1.5 Testen von verschiedenen Eingabemethoden 23
2.1.6 Testen der Änderung der Bildschirmausrichtung 24
2.1.7 Testen von typischen Unterbrechungen 24
2.1.8 Testen von Zugriffsberechtigungen für Gerätefunktionen 24
2.1.9 Testen von Stromverbrauch und Ladezustand 25
2.2 Testen von App-Interaktionen mit Gerätesoftware 25
2.2.1 Testen von Benachrichtigungen 25

2.2.2 Testen von Schnellzugriffsverknüpfungen ... 25
2.2.3 Testen von vom Betriebssystem bereitgestellten Benutzereinstellungen ... 26
2.2.4 Testen von verschiedenen Arten von Apps ... 26
2.2.5 Testen der Interoperabilität mit mehreren Plattformen und Betriebssystemversionen ... 26
2.2.6 Testen der Interoperabilität und Koexistenz mit anderen Apps auf dem Gerät ... 27
2.3 Testen verschiedener Verbindungsmethoden ... 27
3. Häufig verwendete Testarten und Testprozesse für mobile Applikationen – 200 Minuten ... 29
3.1 Häufig verwendete Testarten für mobile Applikationen ... 30
3.1.1 Installierbarkeitstests ... 30
3.1.2 Stresstests ... 31
3.1.3 IT-Sicherheitstests ... 31
3.1.4 Performanztest ... 31
3.1.5 Gebrauchstauglichkeitstest ... 32
3.1.6 Datenbanktests ... 32
3.1.7 Globalisierungs- and Lokalisierungstests ... 33
3.1.8 Testen der Barrierefreiheit ... 33
3.2 Zusätzliche Teststufen beim Testen mobiler Applikationen ... 33
3.2.1 Feldtest ... 33
3.2.2 Testen der App-Store-Zulassung und Post-Release-Testen ... 34
3.3 Erfahrungsbasierte Testverfahren ... 34
3.3.1 Personas und Merkhilfen ... 34
3.3.2 Heuristiken ... 35
3.3.3 Touren ... 35
3.3.4 Sitzungsbasiertes Testmanagement (Session Based Test Management, SBTM) ... 36
3.4 Testprozess und Testvorgehensweisen beim Testen mobiler Applikationen ... 36
3.4.1 Testprozess ... 36
3.4.2 Testvorgehensweisen ... 37
4. Plattformen, Werkzeuge und Umgebung mobiler Applikationen – 80 Minuten ... 39
4.1 Entwicklungsplattformen für mobile Applikationen ... 39
4.2 Gängige Werkzeuge von Entwicklungsplattformen ... 39
4.3 Emulatoren & Simulatoren ... 40
4.3.1 Übersicht über Emulatoren & Simulatoren ... 40
4.3.2 Verwendung von Emulatoren und Simulatoren ... 40
4.4 Einrichtung eines Testlabors ... 41
5. Automatisierung der Testausführung – 55 Minuten ... 42
5.1 Automatisierungsansätze ... 42
5.2 Automatisierungsmethoden ... 43
5.3 Bewertung von Automatisierungswerkzeugen ... 44
5.4 Vorgehensweisen zum Einrichten eines Automatisierungstestlabors ... 44

6. Referenzen 46
6.1 ISTQB®-Dokumente 46
6.2 Referenzierte Bücher (englischsprachig) 46
6.3 Weitere Bücher und Artikel (englischsprachig) 46
6.4 Links (Web/Internet) 47
7. Anhang A – Lernziele/Kognitive Stufen des Wissens 48
7.1 Taxonomiestufe 1: Kennen (K1) 48
7.2 Taxonomiestufe 2: Verstehen (K2) 48
7.3 Taxonomiestufe 3: Anwenden (K3) 48
8. Anhang B – Glossar domänenspezifischer Begriffe 49
9. Index 55

Dank

Die folgenden Personen waren am Review, der Kommentierung und der Abstimmung der englischsprachigen Fassung des ISTQB® Lehrplans beteiligt:

Graham Bath, Veronica Belcher, Armin Born, Geza Bujdoso, YongKang Chen, Wim Decoutere, Frans Dijkman, Florian Fieber, David Frei, Péter Földházi Jr., Chaonian Guo, Attila Gyuri, Ma Haixia, Matthias Hamburg, Zsolt Hargitai, Hongbiao Liu, Ine Lutterman, Marton Matyas, Petr Neugebauer, Ingvar Nordström, Francisca Cano Ortiz, Nishan Portoyan, Meile Posthuma, Emilie Potin-Suau, Liang Ren, Lloyd Roden, Chaobo Shang, Mike Smith, Péter Sótér, Marco Sogliani, Michael Stahl, Chris Van Bael, Paul Weymouth, Salinda Wickramasinghe, Minghui Xu

Die GTB Arbeitsgruppe Mobil dankt GTB-Mitglied Anke Löwer für die Beteiligung an den Reviews der deutschen Übersetzung dieses Lehrplans.

0. Einführung

0.1 Zweck dieses Dokuments

Dieser Lehrplan bildet die Grundlage für das Softwaretest-Qualifizierungsprogramm Mobile Application Testing der Basisstufe (Foundation Level). Das German Testing Board (im Folgenden GTB® genannt) hat diesen Lehrplan in die deutsche Sprache übersetzt. Das GTB® und ISTQB® stellen den Lehrplan folgenden Adressaten zur Verfügung:

- Nationalen Boards des ISTQB® zur Akkreditierung von Trainingsanbietern und zur Erarbeitung von Prüfungsfragen in deutscher Sprache.
- Ausbildungsanbietern zur Erstellung ihrer Kursunterlagen und zur Bestimmung einer geeigneten Unterrichtsmethodik.
- Lernenden zur Vorbereitung auf die Prüfung (im Rahmen eines Schulungskurses oder des freien Lernens).
- Allen Personen, die im Bereich Software- und Systementwicklung tätig sind und ihre fachliche Kompetenz beim Testen von Software verbessern möchten, sowie als Grundlage für Bücher und Fachartikel.

GTB® und ISTQB® können die Nutzung dieses Lehrplans auch anderen Personenkreisen oder Institutionen für andere Zwecke genehmigen, wenn diese vorab eine entsprechende schriftliche Genehmigung einholen und erhalten.

0.2 Certified Foundation Level Mobile Application Testing

Dieses Ergänzungsmodul zur Grundstufe (Foundation Level) des Certified Tester-Ausbildungsprogramms richtet sich an alle in das Thema Softwaretesten involvierten Personen, die ihr Wissen über das Testen mobiler Applikationen vertiefen wollen, bzw. an Personen, die sich in ihrer beruflichen Laufbahn auf das Testen mobiler Applikationen spezialisieren wollen.

Informationen über das Testen mobiler Applikationen, die im ISTQB® Certified Tester Foundation Level-Lehrplan [ISTQB_CTFL_2018] enthalten sind, wurden bei der Erstellung dieses Lehrplans berücksichtigt.

0.3 Geschäftlicher Nutzen

In diesem Abschnitt wird der geschäftliche Nutzen (Business Outcomes nach ISTQB®) aufgelistet, den man von Kandidaten mit einer Zertifizierung als Certified Foundation Level Mobile Application Testing erwarten kann.

Ein CTFL Mobile Application Tester kann …

MAT-01 … die geschäftlichen und technologischen Treiber für mobile Applikationen verstehen und überprüfen, um eine Teststrategie zu erstellen.

MAT-02 … die wichtigsten Herausforderungen, Risiken und Erwartungen beim Testen einer mobilen Applikation identifizieren und verstehen.

MAT-03 … Testarten und Teststufen anwenden, die für mobile Applikationen spezifisch sind.

MAT-04 … allgemein gebräuchliche Testarten, wie die im Foundation Level Lehrplan [ISTQB_CTFL_2018] genannten, im spezifischen Kontext mobiler Applikationen anwenden.

MAT-05 … im Rahmen der im ISTQB®-Testprozess beschriebenen Hauptaktivitäten die Aktivitäten durchführen, die speziell für das Testen mobiler Applikationen erforderlich sind.

MAT-06 … geeignete Umgebungen und geeignete Werkzeuge für das Testen mobiler Applikationen identifizieren und verwenden.

MAT-07 … Methoden und Werkzeuge verstehen, die speziell zur Unterstützung des automatisierten Testens mobiler Applikationen verwendet werden.

0.4 Prüfbare Lernziele

Die Lernziele unterstützen diese geschäftlichen Ziele und dienen zur Ausarbeitung der Prüfung für die Zertifizierung in Certified Foundation Level Mobile Application Testing (CTFL-MAT). Den einzelnen Lernzielen ist jeweils eine kognitive Stufe des Wissens (K-Stufe) zugeordnet.

Die K-Stufen bzw. kognitive Stufen dienen dazu, Lernziele gemäß der überarbeiteten Taxonomie von Bloom [Anderson 2001] zu klassifizieren. Das ISTQB® verwendet diese Taxonomie bei der Erstellung der Prüfungen zu den Lehrplänen.

Dieser Lehrplan berücksichtigt drei verschiedene kognitive Stufen (K1 bis K3). Weitere Informationen finden Sie in Kapitel 7.

0.5 Praktische Kompetenzstufen

In der Grundstufe (Foundation Level) des Testens mobiler Applikationen wird das Konzept der praktischen Ziele eingeführt, die sich auf praktische Fähigkeiten und Kompetenzen konzentrieren.

Durch praktische Übungen können Kompetenzen erworben werden, wie sie in der folgenden nicht erschöpfenden Liste aufgeführt sind:

- Übungen für Lernziele der Stufe K3, die auf Papier oder mit Textverarbeitungssoftware durchgeführt werden (wie bereits bei verschiedenen bestehenden ISTQB®-Lehrplänen).
- Einrichten und Verwenden von Testumgebungen.
- Testen von Applikationen auf virtuellen und physischen Geräten.
- Verwenden von Werkzeugen auf Desktop-Rechnern und/oder mobilen Geräten zum Testen oder für Aufgaben zur Unterstützung des Testens, wie z.B. Installation, Abfragen, Protokollieren, Überwachen, Screenshots erfassen usw.

Für die praktischen Ziele gelten die folgenden praktischen Kompetenzstufen:

- H0: Dies kann eine Live-Demo einer Übung oder ein aufgezeichnetes Video beinhalten. Da dies nicht vom Kursteilnehmer selbst durchgeführt wird, ist es streng genommen keine Übung.
- H1: Geführte Übung. Die Kursteilnehmer befolgen eine Abfolge von Schritten, die vom Seminarleiter ausgeführt werden.
- H2: Übung mit Tipps. Die Kursteilnehmer erhalten eine Übung mit relevanten Hinweisen zur Lösung der Übung innerhalb eines vorgegebenen Zeitrahmens.
- H3: Freie Übungen ohne Tipps.

Empfehlungen:

- Lernziele der Stufe K1 verwenden normalerweise die Stufe H0, bzw. H1 oder H2, wenn es die Situation erfordert.
- Lernziele der Stufe K2 verwenden normalerweise die Stufen H1 oder H2, bzw. H0 oder H3, wenn es die Situation erfordert.
- Lernziele der Stufe K3 verwenden normalerweise die Stufen H2 oder H3, obwohl eine praktische Übung für ein Lernziel der Stufe K3 nicht immer notwendig ist. Wenn der Aufbau komplex oder zu zeitaufwendig ist, verwenden Sie die Stufe H0.

0.6 Prüfung

Die Prüfung für das Zertifikat CTFL Specialist Mobile Application Testing basiert auf diesem Lehrplan. Zur Beantwortung einer Prüfungsfrage kann Wissen aus mehreren Abschnitten dieses Lehrplans erforderlich sein. Alle Abschnitte dieses Lehrplans sind prüfungsrelevant, außer der Einführung und der Anhänge. Im Lehrplan sind Standards, Fachbücher und andere ISTQB®-Lehrpläne als Referenzen genannt; deren Inhalt ist jedoch nicht über das hinaus prüfungsrelevant, was im vorliegenden Lehrplan in zusammengefasster Form enthalten ist.

Das Format der Prüfung ist Multiple Choice. Es sind 40 Fragen zu beantworten. Zum Bestehen der Prüfung müssen mindestens 65% der Fragen (d.h. 26 Fragen) korrekt beantwortet werden. Praktische Ziele und Übungen werden nicht geprüft.

Prüfungen können als Teil eines akkreditierten Seminars oder unabhängig davon (z.B. bei einer Zertifizierungsstelle oder in einer öffentlichen Prüfung) abgelegt werden. Die Teilnahme an einem akkreditierten Seminar stellt keine Voraussetzung für das Ablegen der Prüfung dar.

Wer die Prüfung ohne Teilnahme an einem akkreditierten Seminar ablegen möchte, sollte die Kompetenzrichtlinien im Dokument [CTFL-MAT-2019-Akkreditierungs- und Kompetenzrichtlinien.pdf] lesen und versuchen, die praktischen Übungen selbst durchzuführen. Dies hilft, die Kompetenzen zu erlangen, die von akkreditierten Schulungsanbietern in den Seminaren vermittelt werden. Bitte beachten Sie, dass dies keinen Einfluss auf die Zertifizierungsprüfung CTFL Specialist Mobile Application Testing hat, da die Prüfung ausschließlich auf diesem Lehrplan und seinen Lernzielen basiert.

0.7 Empfohlene Schulungszeiten

Für jedes Lernziel dieses Lehrplans wurde die Mindestschulungszeit festgelegt. Die gesamte Schulungszeit für die einzelnen Kapitel ist in der Kapitelüberschrift angegeben.

Seminaranbieter werden darauf hingewiesen, dass in anderen Lehrplänen des ISTQB® Standardzeiten zur Anwendung kommen, die festgelegte Unterrichtszeiten je nach K-Stufe zuordnen. Der vorliegende Lehrplan wendet dieses Schema jedoch nicht strikt an. Daher erhalten die Schulungsanbieter flexiblere und realistischere Angaben hinsichtlich der jeweiligen Mindestschulungszeiten.

0.8 Voraussetzung für die Prüfung

Voraussetzung für die Prüfung zum CTFL Specialist Mobile Application Testing ist das erworbene Zertifikat zum ISTQB® Certified Tester Foundation Level (CTFL®).

0.9 Informationsquellen

Das offizielle Glossar des GTB definiert die deutschen Übersetzungen der Begriffe, die im ISTQB® Standardglossar [ISTQB_GLOSSARY] enthalten sind. Eine Version des Glossars ist erhältlich vom ISTQB® und von der GTB-Website.

Kapitel 6 enthält eine Liste der empfohlenen Bücher und Artikel zum Testen mobiler Applikationen.

0.10 Geschlechtsneutrale Formulierungen

Aus Gründen der einfacheren Lesbarkeit wird auf die geschlechtsneutrale Differenzierung, wie beispielsweise Benutzer/innen, verzichtet. Sämtliche Rollenbezeichnungen gelten im Sinne der Gleichbehandlung grundsätzlich für alle Geschlechter.

1. Mobilgerätewelt – Geschäftliche und technologische Treiber – 175 Minuten

Schlüsselbegriffe

Risikoanalyse, Risikominderung, risikoorientierter Test, Teststrategie

Lernziele für geschäftliche und technologische Treiber

1.1 Analyse mobiler Nutzungsdaten

MAT-1.1.1 (K2) Sie können beschreiben, wie die Analyse verfügbarer mobiler Nutzungsdaten als Eingabe für die Teststrategie und das Testkonzept verwendet werden kann.

HO-1.1.1 (H3) Sie können basierend auf Daten aus einer oder mehreren Nutzungsdatenquellen (geografischer Standort, Plattform, Betriebssystemversion und Verbreitung des Gerätetyps) die zu testenden Gerätetypen und deren entsprechende Priorisierung auswählen.

Hinweis: HO-1.1.1 und HO-1.7.1 (siehe unten) können kombiniert werden.

1.2 Geschäftsmodelle für mobile Apps

MAT-1.2.1 (K2) Sie können verschiedene Geschäftsmodelle für mobile Applikationen unterscheiden.

1.3 Arten von Mobilgeräten

MAT-1.3.1 (K1) Sie können verschiedene Arten von Mobilgeräten aufzählen.

1.4 Arten mobiler Applikationen

MAT-1.4.1 (K2) Sie können verschiedene Arten mobiler Applikationen unterscheiden.

1.5 Mobile Applikationsarchitektur

MAT-1.5.1 (K2) Sie können zwischen den gängigen Architekturtypen mobiler Applikationen unterscheiden.

1.6 Teststrategie für mobile Apps

MAT-1.6.1 (K3) Sie können bei der Erstellung einer Teststrategie die Merkmale und Besonderheiten der Mobilgerätewelt anwenden.

1.7 Herausforderungen beim Testen mobiler Applikationen

MAT-1.7.1 (K2) Sie können Beispiele für die Herausforderungen beim Testen mobiler Applikationen nennen.

HO-1.7.1 (H1) Sie können für eine ausgewählte Region Marktdaten wie die Marktanteile von Geräten oder Betriebssystemen erheben. Sie können Daten zu Bildschirmgrößen und Bilddichte ermitteln. Sie können eine Liste mit fünf Geräten erstellen und die erwartete Marktabdeckung für diese Liste berechnen.

Hinweis: HO-1.1.1 (siehe oben) und HO-1.7.1 können kombiniert werden.

1.8 Risiken beim Testen mobiler Applikationen

MAT-1.8.1 (K2) Sie können beschreiben, wie sich die Risiken für mobile Applikationen mindern lassen.

1.1 Analyse mobiler Nutzungsdaten

In der Mobilgerätewelt gibt es viele Stakeholder, darunter Hersteller, Plattformanbieter, Betriebssystemanbieter, Marktdatenanbieter, Werkzeuganbieter und natürlich Applikationsentwickler und -tester.

Um effektiv an Testplanung und Testanalysen mitzuwirken, sollte ein Tester mobiler Applikationen die folgenden Faktoren kennen und mit diesen vertraut sein:

- Die geschäftlichen Auswirkungen der Verbreitung von Plattformen
- Heruntergeladene Apps je Plattform
- Die Anzahl und Verbreitung von Betriebssystemversionen
- Die Verbreitung der verschiedenen Gerätetypen im Markt, einschließlich Varianten basierend auf dem geografischen Standort
- Unterschiedliche Bildschirmgrößen und -auflösungen
- Die verschiedenen Eingabemethoden
- Kameratypen

Es gibt verschiedene Informationsquellen für die genannten Faktoren, sowohl kostenlose als auch kommerzielle. Dazu gehören StatCounter GlobalStats [URL1], die Betriebssystemanbieter selbst und andere Quellen von Drittanbietern.

Die mobilen Analysedaten werden verwendet, um ein Geräteportfolio für die Testausführung auszuwählen, das für den Zielmarkt geeignet ist. Über dieses Portfolio werden Tests ausgeführt, um die App auf einem Gerät entsprechend der Wichtigkeit des Geräts zu testen. Die Daten in Bezug auf bestimmte Geräte und ihre besonderen Merkmale, sofern vorhanden, können auch zum Entwerfen von Tests verwendet werden, die für einen Gerätetyp spezifisch sind. Beispielsweise benötigt ein Gerät mit Herzschlagsensor möglicherweise spezielle Testfälle.

1.2 Geschäftsmodelle für mobile Apps

Es gibt verschiedene Modelle, mit denen die bei der Erstellung mobiler Applikationen geleistete Arbeit in einen geldwertigen Vorteil umgewandelt werden kann. Hierzu zählen unter anderem: Freemium-, werbefinanzierte, transaktionsbasierte, kostenpflichtige und Unternehmens-Applikationen. Darüber hinaus können In-App-Käufe bei einigen dieser Modelle angewendet werden.

Für jeden dieser Ansätze gibt es bestimmte Vor- und Nachteile, und der Tester muss beim Testen der mobilen Applikation das jeweilige Geschäftsmodell berücksichtigen.

Bei einem Freemium-Modell sind die Applikationen im Allgemeinen kostenlos, Benutzer müssen jedoch bezahlen, wenn sie zusätzliche Funktionen benötigen. Die Applikationen müssen ausreichende Funktionen zur Verfügung stellen, um für die Benutzer attraktiv zu sein, und sie müssen darüber hinaus erweiterte Funktionen bereitstellen, für die eine große Anzahl von Benutzern bereit wäre, zu zahlen.

Bei werbefinanzierten Applikationen werden Anzeigen auf dem Bildschirm angezeigt, während Benutzer mit den Applikationen interagieren. Diese Strategie zur Umsatzgenerierung ist effektiver, wenn die Applikationen über einen längeren Zeitraum verwendet werden. Die Designer der Benutzungsschnittstelle müssen bei der Anzeige von Werbung darauf achten, dass die angezeigte Werbung auffällig genug ist, ohne jedoch wesentliche Teile der Applikation zu verbergen. Auch müssen sie sicherstellen, dass Benutzer nicht abgelenkt werden und die Applikation nicht mögen.

Transaktionsbasierte Applikationen berechnen dem Nutzer entweder eine Pauschale pro Transaktion oder einen Prozentsatz des Transaktionswertes oder ähnliches. Dieses Geschäftsmodell eignet sich nur für eine begrenzte Anzahl von Applikationen und wird normalerweise für Geschäfts- und Finanz-Apps wie mobile Geldbörsen (Wallets) angewendet.

Bei kostenpflichtigen Applikationen müssen die Benutzer für das Herunterladen und Installieren der App bezahlen. Die Entscheidung für ein kostenpflichtiges Geschäftsmodell sollte gut überlegt sein, da für die meisten Arten von Applikationen eine große Anzahl von kostenlosen oder Freemium-Optionen zur Verfügung steht. Die Wahrscheinlichkeit, dass Benutzer solche Apps kaufen, steigt, wenn sie herausragende Funktionen oder Benutzerfreundlichkeit bieten oder wenn konkurrierende Apps nicht verfügbar sind.

Gratis- und Unternehmens-Applikationen berechnen ihren Nutzern keine Gebühren. Unternehmens-Apps sind für den unternehmensinternen Gebrauch entwickelt und bieten eine Schnittstelle zu den bereitgestellten Diensten. Es gibt viele solcher Apps von Organisationen wie Banken oder E-Commerce-Unternehmen. Diese Apps bezwecken im Allgemeinen nicht die Wirtschaftlichkeit der App selbst, sondern unterstützen die Erzielung von Einnahmen, indem die Benutzer zu den von den Organisationen bereitgestellten Diensten geleitet werden.

1.3 Mobile Gerätetypen

Es gibt eine Vielzahl von Mobilgeräten, die verschiedene Arten von Applikationen unterstützen.

Typische Geräte sind:

- Basistelefone
- Feature-Phones
- Smartphones
- Tablets
- Begleitgeräte - einschließlich Wearables und einige IoT-Geräte (Internet of Things).

Beim Testen ist zu beachten, dass jeder Gerätetyp spezifische Funktionen für bestimmte Anforderungen aufweist.

Basistelefone werden nur für Telefon und SMS verwendet und bieten nur sehr wenige integrierte Apps und Spiele. Die Installation von Apps oder das Surfen im Internet sind nicht möglich.

Feature-Phones bieten eingeschränkten Support für Apps und deren Installation. Sie bieten Internetzugang über einen integrierten Browser und verfügen möglicherweise über zusätzliche Hardware wie Kameras.

Smartphones sind mit mehreren Sensoren ausgestattet. Das Betriebssystem unterstützt Funktionen wie die Installation von Apps, Multimedia-Unterstützung und das Surfen via Browser.

Tablets ähneln den Smartphones, sind jedoch größer. Sie werden normalerweise verwendet, wenn ein größeres Display benötigt oder gewünscht wird. Tablets können auch eine längere Akkulaufzeit bieten.

Begleitgeräte Smartphone oder Tablet zu ermöglichen....

Wearables sind Geräte, die von Benutzern getragen werden können. Diese können als Begleitgeräte zu vorhandenen Geräten fungieren oder unabhängig voneinander funktionieren. Uhren und Fitnessbänder sind Beispiele für beliebte Wearables.

1.4 Arten mobiler Applikationen

Es gibt drei Haupttypen von mobilen Applikationen:

- Native Apps
- Web-Apps
- Hybride Apps

Jede Art von Applikation hat bestimmte Vor- und Nachteile, sodass vor Beginn der Applikationsentwicklung eine Geschäftsentscheidung getroffen werden muss.

Native Applikationen werden mit plattformspezifischen Software Development Kits (SDKs), Entwicklungswerkzeugen und plattformspezifischen Sensoren und Funktionen entwickelt. Sie werden aus den App-Stores der Anbieter heruntergeladen, installiert und aktualisiert. Diese Apps müssen möglicherweise auf allen unterstützten Geräten getestet werden.

Native Applikationen bieten im Allgemeinen eine bessere Leistung, können die Plattformfunktionen uneingeschränkt nutzen und die Erwartungen bezüglich der Plattform erfüllen, für die sie entwickelt wurden. Die Entwicklungskosten sind in der Regel höher und es kann zusätzliche Herausforderungen geben, z.B. die Verwendung mehrerer Plattformen sowie die Installation und das Testen auf einer großen Anzahl von Geräten.

Auf Web-Applikationen wird über einen mobilen Browser zugegriffen. Da diese die typischen Webentwicklungstechnologien und Browser verwenden, ist die Unterstützung mehrerer Plattformen einfach und die Entwicklungskosten sind normalerweise niedriger.

Es gibt hauptsächlich vier Möglichkeiten, wie mobile Webapplikationen erstellt werden:

- Mobil-spezifische Versionen von Websites und Apps (diese werden auch als m(dot) Sites bezeichnet). In der Regel bedeutet dies, dass, wenn ein mobiler Browser die Applikation anspricht, eine mobile Version der Applikation bereitgestellt wird. Beispielsweise leitet facebook.com beim Zugriff von einem mobilen Gerät zu m.facebook.com um.
- Responsive Web-Apps sorgen dafür, dass sich das Design an Form und Bildschirmgröße (d.h. an den Anzeigebereich, auch View Port genannt) anpasst.
- Adaptive Web-Apps passen das Design an einige vordefinierte Größen an. Für die einzelnen Größen gibt es unterschiedliche Designs, und die dem Benutzer zur Verfügung stehenden Funktionen sind häufig einstellbar.
- Progressive Web-Apps ermöglichen die Verwendung von Kurzbefehlen für die Erstellung bestimmter Webseiten auf dem mobilen Startbildschirm. Sie sehen aus wie native Apps und können manchmal sogar offline arbeiten.

Mobile Web-Apps werden mit gängigen Webtechnologien erstellt, wodurch sie im Vergleich zu nativen und hybriden Apps in der Regel einfacher zu entwickeln und zu verwalten sind. Sie haben jedoch möglicherweise nicht so viele Funktionen wie native oder hybride Apps, und sie haben möglicherweise nur eingeschränkten Zugriff auf die nativen Anwendungsprogrammierschnittstellen (APIs) der Plattform. Auch der Zugriff auf mobile Sensoren ist eingeschränkt. Während Installationstests auf Geräten nicht benötigt werden, sind Browser-Kompatibilitätstests dagegen erforderlich.

Hybride Applikationen sind eine Kombination aus nativer und Web-App. Sie verwenden einen nativen App-Wrapper, der eine Webansicht enthält, um eine Webapplikation in einer nativen App auszuführen. Diese Apps werden von App-Stores der Anbieter heruntergeladen und können auf alle Gerätefunktionen zugreifen. Sie sind relativ einfach zu entwickeln, zu aktualisieren und zu warten, ohne die auf dem Gerät installierte App zu aktualisieren. Die für die Entwicklung dieser Apps erforderlichen Fähigkeiten sind fast die gleichen wie für die Webentwicklung. Zu den möglichen Schwachstellen dieser Apps zählen Leistungsprobleme aufgrund der Verwendung eines Wrappers und mögliche Abweichungen vom erwarteten Erscheinungsbild bzw. Look-and-Feel aufgrund plattformspezifischer Aspekte.

Native und hybride Apps werden physisch auf einem Gerät installiert und stehen dem Benutzer daher immer zur Verfügung, auch wenn das Gerät keine Internetverbindung hat. Im Vergleich dazu benötigen Web-Apps einen Internetzugang.

Einige Apps sind auf dem Mobilgerät vorinstalliert, andere können über verschiedene Vertriebskanäle wie z.B. Apple App Store, Google Play Store, Unternehmens-App Stores (nur im Unternehmensnetzwerk verfügbar) und über App-Marktplätze von Drittanbietern installiert werden.

Das Testen der verschiedenen Arten von Applikationen erfordert möglicherweise einen unterschiedlichen Ansatz. Die zu berücksichtigenden Parameter umfassen:

- Verschiedene zu unterstützende Gerätetypen
- Zu verwendende Sensor- und Gerätefunktionen
- Verfügbarkeit unter verschiedenen Netzwerkbedingungen
- Installierbarkeit, Kompatibilität, Performanz und Gebrauchstauglichkeit

1.5 Mobile Applikationsarchitektur

Es gibt mehrere Lösungen für das Entwerfen einer mobilen Applikation.

Einige Faktoren, die bei der Wahl einer bestimmten Architektur- oder Entwurfsentscheidung zu berücksichtigen sind:

- Zielgruppe
- Art der Applikation
- Unterstützung verschiedener mobiler und nicht-mobiler Plattformen
- Bedarf bezüglich der Verbindungsfähigkeit
- Datenspeicherbedarf
- Verbindungen zu anderen Geräten, einschließlich IoT-Geräten

Die Architekturentscheidungen werden beeinflusst durch:

- Clientseitige Architektur, wie z. B. Thin- oder Fat-Client
- Serverseitige Architektur, wie z. B. ein- oder mehrschichtig (Single- bzw. Multi-Tier)
- Verbindungstyp, wie z. B. WLAN, mobile Daten, Nahfeldkommunikation (NFC), Bluetooth
- Datensynchronisierungsmethoden, wie z. B. Teilstreckenverfahren (Store-and-Forward), Push & Pull, synchrone und asynchrone Kommunikationen

Thin-Client Apps enthalten keinen Anwendungscode, der an das Gerät angepasst ist und nutzen die Funktionen des mobilen Betriebssystems nur minimal. Diese Apps verwenden normalerweise den Webbrowser als Frontend und JavaScript als Sprache für die Implementierung der clientseitigen Logik.

Thick-/Fat-Client Apps können mehrere Schichten von Anwendungscode haben und mobile Betriebssystemfunktionen verwenden. Hierbei handelt es sich typischerweise um native oder hybride Apps.

Bei den serverseitigen Architekturen gibt es folgende Möglichkeiten:

- Einschichtige Architekturen (Single-Tier) sind monolithisch aufgebaut und haben alle Server auf demselben Rechner. Sie sind weniger skalierbar und schwerer abzusichern.
- Mehrschichtige verteilen die serverseitigen Komponenten auf verschiedene Schichten. Zweischichtige Architekturen (Two-Tier) beinhalten separate Web- und Datenbankserver, während dreischichtige Architekturen (Three-Tier) auch einen Anwendungsserver beinhalten. Mehrschichtige Architekturen ermöglichen die Aufteilung von Verantwortlichkeiten, die Spezialisierung von Datenbanken sowie eine bessere Flexibilität, Skalierbarkeit und Sicherheit. Sie können jedoch im Vergleich zu einschichtigen Architekturen erheblich teurer in Entwicklung, Verwaltung und Hosting.

Es gibt verschiedene Verbindungsmethoden. Ein mobiles Endgerät kann auf verschiedene Art und Weise mit dem Server verbunden werden, z.B. über WLAN oder über mobile Datenverbindungen wie 2G, 3G, 4G und 5G. Mobile Applikationen werden in einer der drei folgenden Modi betrieben:

- Nie verbundene Apps funktionieren offline und müssen nicht verbunden werden. Ein einfacher Taschenrechner ist ein Beispiel für eine solche App.
- Immer verbundene Apps benötigen während sie ausgeführt werden eine permanente Netzwerkverbindung. Alle mobilen Webapplikationen fallen in diese Kategorie, obwohl einige auch eingeschränkt funktionieren können, wenn sie teilweise verbunden sind.
- Teilweise verbundene Apps benötigen eine Verbindung für Aufgaben wie die Datenübertragung, können jedoch über einen längeren Zeitraum ohne Verbindung ausgeführt werden.

Zur Synchronisierung von Daten zwischen dem Client und dem Server können folgende Methoden gewählt werden:

- Im kontinuierlichen Modus werden die Daten übertragen, sobald sie bereitgestellt werden.
- Beim Teilstreckenverfahren können die Daten vor der Übertragung lokal gespeichert werden, insbesondere wenn gerade keine Netzverbindung vorhanden ist.

Die Datenübertragung kann auf zwei Arten erfolgen:

- Bei der synchronen Datenübertragung wartet die aufrufende Funktion, bis die aufgerufene Funktion ausgeführt wurde, bevor sie zurückkehrt.
- Bei der asynchronen Datenübertragung kehrt die aufgerufene Serverfunktion sofort zurück, verarbeitet die Daten im Hintergrund, und ruft die aufrufende Clientfunktion erst nach Abschluss der Aufgabe zurück. Dies gibt den Benutzern mehr Kontrolle. Die Implementierung des Handshake-Mechanismus erhöht jedoch die Komplexität hinsichtlich der Verfügbarkeit des Clients oder des Netzwerks, wenn der Server den Rückruf initiiert.

1.6 Teststrategie für mobile Apps

Um eine Teststrategie für mobile Geräte zu erstellen, muss der Tester alle in diesem Kapitel aufgeführten Punkte beachten. Darüber hinaus sind die in diesem Abschnitt beschriebenen Risiken und die in Abschnitt 1.7 beschriebenen Herausforderungen zu berücksichtigen.

Typische Risiken sind zum Beispiel:

- Unkenntnis der Verbreitungsdaten der mobilen Geräte an einem bestimmten geografischen Ort macht es unmöglich, auf nachhaltige Art und Weise die Geräte auszuwählen, auf denen die App getestet werden muss.
- Fehlende Informationen über den Typ des Geschäftsmodells verhindern eine Bewertung, ob das Verhalten der App dem Geschäftsmodell genügt,

Beim Erstellen einer Teststrategie für das Testen mobiler Applikationen müssen darüber hinaus die folgenden spezifischen Risiken und Herausforderungen berücksichtigt werden:

- Die Vielfalt mobiler Geräte, von denen einige gerätespezifische Mängel aufweisen.
- Die Verfügbarkeit von Geräten im Unternehmen selbst oder durch die Nutzung externer Testlabore.
- Die Einführung neuer Technologien, Geräte und/oder Plattformen während des Applikationslebenszyklus.

- Die Installation und Aktualisierung der App selbst über verschiedene Kanäle, einschließlich der Bereitstellung von App-Daten und -Einstellungen.
- Plattformprobleme, die sich auf die Applikation auswirken könnten.
- Die Netzwerkabdeckung und ihre Auswirkungen auf die App im globalen Kontext.
- Die Fähigkeit, die Netze verschiedener Dienstanbieter für den Test zu nutzen.
- Die Verwendung von Mobilgeräte-Emulatoren, -Simulatoren und/oder realen Geräten für bestimmte Teststufen und Testarten.

Diese Herausforderungen werden ausführlicher in Abschnitt 1.7 beschrieben.

Die Teststrategie berücksichtigt die Risiken und Herausforderungen, wie sie in den folgenden Beispielen dargestellt sind:

- Die Teststrategie kann den Einsatz von Mobilgeräte-Emulatoren/-Simulatoren in frühen Entwicklungsphasen vorgeben, gefolgt von realen Geräten in späteren Phasen. Es gibt bestimmte Arten von Tests, die mit diesen Emulatoren/-Simulatoren durchgeführt werden können, aber nicht alle Testarten kommen dafür in Frage. Mehr dazu finden Sie in Abschnitt 4.3.
- Die Teststrategie kann die Herausforderung einer großen Anzahl verschiedener Geräte berücksichtigen, indem sie einen der folgenden Ansätze anwendet:
  - Einzelplattform-Ansatz: Reduzierung des Testumfangs auf einen einzigen Gerätetyp, eine Betriebssystemversion, einen Netzbetreiber und einen Netzwerktyp.
  - Multiplattform-Ansatz: Reduzierung des Testumfangs auf eine repräsentative Auswahl von Geräten und Betriebssystemen, die von der Mehrheit der Kunden auf dem Zielmarkt verwendet werden, basierend auf mobile Verbindungsdaten oder anderen Analysedaten.
  - Ansatz der maximalen Überdeckung: Deckt alle Betriebssystemversionen, Geräte, Hersteller, Betreiber und Netzwerktypen ab. Dies ist im Grunde genommen ein erschöpfender Test, der in der Regel nicht wirtschaftlich ist, insbesondere wenn die Vielzahl der auf dem Markt befindlichen Geräte und Betriebssystemversionen berücksichtigt wird.
- Die Teststrategie kann die Herausforderung berücksichtigen, die sich aus der Nichtverfügbarkeit von Geräten, Netzwerken oder realen Bedingungen ergibt, indem externe Ressourcen eingesetzt werden, wie z.B.:
  - Dienste für den Fernzugriff auf Geräte. Dies ist eine Möglichkeit, über das Web auf Geräte zuzugreifen, die auf andere Weise nicht zur Verfügung stehen.
  - Crowdtesting-Dienste. Dies ist eine Möglichkeit, auf eine große Gruppe von Freiwilligen und deren Geräte zuzugreifen.
  - Persönliche Netzwerke wie Freunde und Kollegen. Hier wird das eigene soziale Netzwerk genutzt.
  - Fehlerjagd (Bug Hunting). Dies ist eine als Spiel organisierte („gamified“) Testveranstaltung mit Freiwilligen aus dem Unternehmen oder der Öffentlichkeit.

Zusätzlich zu den im Foundation Level-Lehrplan [ISTQB_CTFL_2018] beschriebenen Teststufen werden in der Teststrategie auch die gängigen Testarten für mobile Applikationen (siehe Abschnitt 3.1) und eventuell zusätzliche benötigte Teststufen (siehe Abschnitt 3.2) berücksichtigt.

1.7 Herausforderungen beim Testen mobiler Applikationen

In der Mobilgerätewelt gibt es viele weitere Herausforderungen, die bei Desktop- oder Serversoftware ungewöhnlich oder unkritisch sind. Die Tester müssen sich dieser Herausforderungen bewusst sein und wissen, wie sie sich auf den Erfolg der Applikation auswirken können.

Typische Herausforderungen in der mobilen Welt sind:

- Mehrere Plattformen und Gerätefragmentierung: Mehrere unterschiedliche Betriebssystemtypen und -versionen, Bildschirmgrößen und Darstellungsqualitäten.
- Unterschiedliche Hardware bei verschiedenen Geräten: Verschiedene Sensortypen und Schwierigkeiten bei der Simulation von Testbedingungen für eingeschränkte CPU- und RAM-Ressourcen.
- Eine Vielzahl von Softwareentwicklungswerkzeugen, die für die Plattformen benötigt werden.
- Unterschiede im Design der Benutzungsschnittstellen und beim erwarteten Benutzererlebnis (UX) bei den verschiedenen Plattformen.
- Mehrere Netzwerktypen und -anbieter.
- Geräte mit stark limitierten Ressourcen.
- Verschiedene Vertriebskanäle für Apps.
- Verschiedene Benutzer und Benutzergruppen.
- Verschiedene Arten von Apps mit unterschiedlichen Verbindungsmethoden.
- Hohe Sichtbarkeit von Fehlern, die sich stark auf die Benutzer auswirken und leicht dazu führen können, dass diese Bewertungen in Online-Marktplätzen veröffentlichen.
- Marktplatzveröffentlichung, die zusätzliche Genehmigungszyklen für die Veröffentlichung durch App-Stores wie Google Play oder Apple App Store erfordern.
- Nichtverfügbarkeit von neu auf den Markt gebrachten Geräten, die den Einsatz von Mobilgeräte-Emulatoren/-Simulatoren erfordern.

Die Auswirkungen dieser Herausforderungen sind u.a.:

- Eine große Anzahl von zu testenden Kombinationen.
- Eine große Anzahl von erforderlichen Geräten für die Testausführung, was die Kosten erhöht.
- Die Notwendigkeit der Abwärtskompatibilität, um die Applikation auf älteren Versionen der Plattform auszuführen.
- Der Release von neuen Funktionen für jede Version des zugrunde liegenden Betriebssystems.
- Richtlinien, die für die verschiedenen Plattformen zu beachten sind.
- Eingeschränkte Verfügbarkeit von CPU-Ressourcen sowie Limitierung des Arbeitsspeichers und Speicherplatzes.

- Unterschiedliche Bandbreiten und Übertragungsschwankungen (Jitter) verschiedener Netzwerke.
- Änderungen der verfügbaren Upload- und Download-Geschwindigkeiten basierend auf Datentarifmodellen.

Die beiden folgenden Beispiele veranschaulichen typische Herausforderungen und ihre möglichen Auswirkungen:

- Unterschiedliche Geräte haben verschiedene Sensortypen, die beim Testen berücksichtigt werden müssen. Jeder neue Sensor, der zur Hardware hinzugefügt wird, erfordert möglicherweise zusätzliche Tests bzgl. Abwärtskompatibilität.
- Einige das Netzwerk betreffende Herausforderungen lassen sich auch unter unterschiedlichen Netzwerkbedingungen mit geeigneten Caching- und Prefetch-Strategien angemessen bewältigen. Dies ist jedoch mit Kosten verbunden. Eine große Anzahl offener Verbindungen kann die serverseitige Leistung beeinträchtigen, da bei den meisten Apps der Benutzer weiterhin auf dem Server angemeldet bleibt.

1.8 Risiken beim Testen mobiler Applikationen

Die in Abschnitt 1.7 erwähnten Herausforderungen können einzeln oder in Kombination mit anderen Herausforderungen auftreten. Dies kann zu zusätzlichen Risiken für eine mobile Applikation führen.

Ein Tester muss in der Lage sein, einen Beitrag zur Risikoanalyse des Produkts zu leisten. Gängige Risikoanalyse- und Risikominderungsmethoden, wie im Foundation Level-Lehrplan [ISTQB_CTFL_2018], Kapitel 5.5 beschrieben, können auch im mobilen Kontext angewendet werden. Darüber hinaus existieren die folgenden spezifisch mobilen Risiken und Strategien zu Risikominderung:

Risiko	Mögliche Risikominderung
Marktfragmentierung	Bestimmen einer geeigneten Geräteauswahl für die Testausführung, z.B. Testen der am häufigsten verwendeten Geräte.
Kosten für die Unterstützung mehrerer Plattformen	Durchführung einer Analyse, um zu verstehen, welche Plattformen am häufigsten verwendet werden, um dadurch das Testen nicht relevanter Plattformen zu vermeiden.
Einführung neuer Technologien, Plattformen und Geräte	Verwendung von Vorproduktionsversionen dieser Technologien für den Test.
Mangelnde Verfügbarkeit von Geräten für die Testausführung	Nutzung von Diensten für den Fernzugriff auf Geräte oder von Crowdtesting-Diensten.
Risiken aus den zu erwartenden Nutzungsmustern mobiler Applikationen für unterwegs	Anwenden geeigneter Testansätze wie z.B. den Feldtest.

2. Testarten für mobile Applikationen – 265 Minuten

Schlüsselbegriffe

browserübergreifende Kompatibilität, Gebrauchstauglichkeit, Interoperabilität, Koexistenz, Kompatibilität, System unter Test (SUT), Testart, Verbindungsfähigkeit

Lernziele für die Testarten für mobile Applikationen

2.1 Testen der Kompatibilität mit Gerätehardware

MAT-2.1.1 (K2) Sie können beschreiben, welche gerätespezifischen Funktionen und Hardware beim Testen berücksichtigt werden sollten.

HO-2.1.1 (H1) Sie können mehrere mobile Gerätefunktionen einer App testen, während das zu testende System (SUT) verwendet wird, um die korrekte Funktion des SUT zu überprüfen.

MAT-2.1.2 (K3) Sie können Tests für die Kompatibilität der App mit verschiedenen Bildschirmgrößen, Seitenverhältnissen und Bilddichten erstellen.

HO-2.1.2 (H3) Sie können eine App auf mehreren mobilen Geräten (virtuell oder physisch) testen, um die Auswirkungen der Auflösung und der Bildschirmgröße auf die Benutzungsschnittstelle der App zu ermitteln.

MAT-2.1.3 (K2) Sie können beschreiben, wie Tests die möglichen Auswirkungen einer Überhitzung des Geräts im SUT aufzeigen können.

MAT-2.1.4 (K1) Sie können verschiedene Testarten zum Testen der verschiedenen Eingabesensoren, die in Mobilgeräten verwendet werden, wiedergeben.

MAT-2.1.5 (K1) Sie können die Tests wiedergeben, die für verschiedene Eingabemethoden ausgeführt werden sollten.

HO-2.1.5 (H0) Sie können verschiedene Eingabemethoden von Apps testen, einschließlich tastaturbezogener Tests mit mehreren installierten Tastaturen, gestenbezogener Tests und (optional) kamerabezogener Tests.

MAT-2.1.6 (K2) Sie können beschreiben, wie Tests Probleme mit der Benutzungsschnittstelle aufdecken können, wenn sich die Bildschirmausrichtung ändert.

HO-2.1.6 (H3) Sie können eine Applikation testen, um die Auswirkung von Ausrichtungsänderungen auf die Funktionalität der App zu überprüfen, einschließlich der Auswirkung auf Datenhaltung und Richtigkeit der Benutzungsschnittstelle.

MAT-2.1.7 (K3) Sie können Tests für eine App mit typischen Unterbrechungen für Mobilgeräte erstellen.

HO-2.1.7 (H3) Sie können eine App auf mehrere Mobilgeräte-Unterbrechungen während der Ausführung der Applikation testen.

MAT-2.1.8 (K3) Sie können Tests zum Ändern der Zugriffsberechtigungen auf die von der App angeforderten Gerätefunktionen erstellen.

HO-2.1.8 (H3) Sie können die Berechtigungsverwaltung einer App testen, indem Sie die angeforderten Berechtigungen zulassen und ablehnen und das Verhalten beobachten, wenn Ordner und Sensoreinstellungen bei der Installation abgelehnt oder nach der Installation geändert werden.

MAT-2.1.9 (K3) Sie können Tests erstellen, um die Auswirkungen einer App auf den Stromverbrauch eines Geräts und die Auswirkungen von dessen Ladezustand auf die App zu verifizieren.

HO-2.1.9 (H3) Sie können eine App unter verschiedenen Batterieladezuständen testen, um Verbrauchsdaten zu ermitteln und die Leistung bei niedrigem bzw. leerem Batteriezustand zu ermitteln.

2.2 Testen der Interaktionen der App mit der Gerätesoftware

MAT-2.2.1 (K3) Sie können Tests für die Bearbeitung von Benachrichtigungen durch das zu testende System erstellen.
HO-2.2.1 (H2) Sie können die Wirkung des Empfangs von Benachrichtigungen testen, wenn eine App im Vordergrund und im Hintergrund läuft, und wie sich das Ändern der Benachrichtigungseinstellungen auf die Funktionalität der App auswirkt.
MAT-2.2.2 (K2) Sie können beschreiben, wie Tests die korrekte Funktionalität von Schnellzugriffsverknüpfungen verifizieren können.
HO-2.2.2 (H3) Sie können die Kurzbefehl-/Schnellzugriffsfunktionalität von Apps testen.
MAT-2.2.3 (K3) Sie können testen, wie sich die von einem Betriebssystem bereitgestellten Benutzereinstellungen auf eine App auswirken.
HO-2.2.3 (H3) Sie können eine laufende App testen, indem Sie die Eingabewertoptionen für die vom Betriebssystem bereitgestellten Einstellungen ändern.
MAT-2.2.4 (K2) Sie können zwischen verschiedenen Tests unterscheiden, die für native, Web- und hybride Applikationen erforderlich sind.
HO-2.2.4 (H0) (optional) Sie können die für Apps erforderlichen Tests je nach Typ der mobilen App identifizieren.
MAT-2.2.5 (K1) Sie können erforderliche Tests für Apps wiedergeben, die auf mehreren Plattformen oder Betriebssystemversionen verfügbar sind.
MAT-2.2.6 (K1) Sie können die für die Koexistenz und Interoperabilität mit anderen Apps erforderlichen Tests wiedergeben.

2.3 Testen verschiedener Verbindungsmethoden

MAT-2.3.1 (K2) Sie können die zum Testen der Verbindungsfähigkeit erforderlichen Tests zusammenfassen, einschließlich von Tests über Netzwerke hinweg, bei einer Verwendung von Bluetooth und beim Wechsel in den Flugmodus.
HO-2.3.1 (H0) (optional) Sie können Tests an einer Applikation durchführen, die Daten auf den Server überträgt, wenn das Telefon aufgrund der verfügbaren Signalstärken zwischen WLAN und mobiler Datenverbindung wechselt.

2.1 Testen der Kompatibilität mit der Gerätehardware

2.1.1 Testen von Gerätefunktionen

Unterschiedliche mobile Endgeräte mit unterschiedlichen Fähigkeiten bedeuten, dass Kompatibilitätstests an einer großen Anzahl von Geräten durchgeführt werden müssen. Dies erfordert eine Priorisierung der Zielgeräte für das Testen. Für die Priorisierung werden Marktdaten verwendet, wie in Abschnitt 1.1 erläutert, um ein für den Zielmarkt am besten geeignetes Geräteportfolio auszuwählen. Die Auswahl des Geräteportfolios ist in der Regel ein Kompromiss zwischen Marktabdeckung, Kosten und Risiko.

Applikationen können auf verschiedenen Gerätetypen mit den folgenden Funktionen installiert werden:

- Verschiedene Möglichkeiten das Gerät auszuschalten
- Verschiedene Navigationsmöglichkeiten
- Verwendung von mechanischen und von Bildschirmtastaturen
- Verschiedene Hardwarefunktionen wie zum Beispiel:
  - Radio
  - USB
  - Bluetooth
  - Kameras
  - Lautsprecher

- Mikrofone
- Kopfhöreranschluss

Keine dieser Funktionen sollte den Betrieb der Applikation beeinträchtigen.

Die Gerätefunktionen sind vielfältig und können sich sogar zwischen verschiedenen Gerätemodellen desselben Herstellers unterscheiden. Sie werden häufig zur Unterscheidung von Marktsegmenten verwendet und können sich im Laufe der Zeit schnell ändern. Beispielsweise ist es derzeit durchaus üblich, dass Geräte der oberen und mittleren Preisklasse mit Fingerabdrucksensoren ausgestattet sind, Geräten der unteren Preisklasse hingegen nicht. Dies ändert sich im Laufe der Zeit. Vor einigen Jahren waren Fingerabdrucksensoren in überhaupt keinem Mobilgerät enthalten. Aufgrund dieser Veränderbarkeit benötigen Tester ein klares Verständnis der mobilen Endgeräte und der von den Benutzern erwarteten Funktionen. Es ist Aufgabe der Tester, das Geräteportfolio zu erstellen und die entsprechenden Tests dafür zu entwerfen.

Im Allgemeinen reicht es nicht aus, zu testen, ob die Applikation mit den erwarteten Funktionen ordnungsgemäß funktioniert. Es muss außerdem getestet werden, ob die App auch dann erwartungsgemäß funktioniert, wenn eine bestimmte Funktion nicht vorhanden ist. Beispielsweise sollte eine App, die die Verwendung einer vorderen und einer hinteren Kamera unterstützt, nicht abstürzen, wenn sie auf einem Gerät mit mehreren Kameras, mit nur einer Kamera oder mit überhaupt keiner Kamera installiert und ausgeführt wird.

2.1.2 Testen unterschiedlicher Gerätedisplays

Gerätedisplays können verschiedene Bildschirmgrößen, unterschiedlich große Anzeigebereiche, Seitenverhältnisse und Auflösungen aufweisen, die in Pixel pro Zoll/Inch (ppi) und als Punktdichte in Punkten pro Zoll/Inch (DPI) gemessen werden. Die Gerätefragmentierung macht eine Priorisierung erforderlich. Es sollten Tests erstellt werden, bei denen die Benutzungsschnittstellen auf verschiedenen Geräten mit unterschiedlichen Bildschirmgrößen, Auflösungen und Seitenverhältnissen verwendet werden, die auf dem Zielmarkt am häufigsten vorkommen.

Beim Testen unterschiedlicher Gerätedisplays muss Folgendes überprüft werden:

- Die App skaliert alle Elemente der Benutzungsschnittstelle entsprechend der aktuellen Bilddichte und Bildschirmgröße.
- Die Elemente der Benutzungsschnittstelle überlappen sich nicht.
- Es treten keine Usability- oder Touch-Probleme auf.
- Es gibt kein problematisches Schrumpfen von Bildern aufgrund einer hohen Punktdichte (bzw. DPI/ppi).

2.1.3 Testen der Gerätetemperatur

Im Gegensatz zu Desktop-Computern reagieren mobile Geräte unterschiedlich auf einen Anstieg der Gerätetemperatur.

Mobile Geräte können aus verschiedenen Gründen überhitzen, z.B. durch Aufladen des Akkus, hohe Arbeitsbelastung, im Hintergrund ausgeführte Apps, ständige Nutzung von mobilen Daten, WLAN oder GPS.

Eine Überhitzung kann sich auf das Gerät auswirken, da versucht wird, die Erwärmung zu verringern und den Batteriestand zu erhalten. Dies kann einen Abfall der CPU-Frequenz, die Freigabe von Speicher und das Ausschalten von Teilen des Systems beinhalten.

Wenn es dazu kommt, kann sich dies auch auf die Funktionalität der App auswirken und muss daher bei der Planung von Tests berücksichtigt werden. Tests müssen so entworfen werden, dass sie viel Energie verbrauchen, was über einen langen ununterbrochenen Zeitraum zur Erzeugung von Wärme führt. Die zu testende Software darf dann kein unerwartetes Verhalten aufweisen.

2.1.4 Testen der Geräteeingabesensoren

Mobile Geräte empfangen viele verschiedene Arten von Sensoreingaben, die beispielsweise GPS, Beschleunigungsmesser, Gyroskope und 3-Achsen-Magnetometer verwenden, oder die auf Druck, Temperatur, Luftfeuchtigkeit, Herzschlag, Licht oder berührungslose Eingaben reagieren.

Beim Testen von verschiedenen Geräteeingabesensoren wird Folgendes überprüft:

- Die App funktioniert für jeden der verfügbaren Sensoren wie vorgesehen. Zum Beispiel muss die App auf verschiedene Bewegungsarten wie z.B. Kreisbewegung und Hin- und Herbewegung (wie beim Gehen) getestet werden.
- Funktionen, die auf externe Beleuchtung reagieren, reagieren unter verschiedenen Lichtbedingungen korrekt.
- Ein- und Ausgänge für Sound reagieren in Verbindung mit Softkeys und physischen Tasten für Lautstärke, Mikrofonen, kabelgebundenen und kabellosen Lautsprechern und bei verschiedenen Umgebungsgeräuschen korrekt.
- Die Standortposition ist unter folgenden Bedingungen genau:
    - Beim Ein- und Ausschalten des GPS.
    - Bei unterschiedlicher Empfangsqualität des GPS-Signals.
    - Wenn die App auf verschiedene andere Methoden zur Standortbestimmung zurückgreifen muss, z.B. WLAN, Mobilfunkstandort oder manuelle Standorteingabe.

2.1.5 Testen von verschiedenen Eingabemethoden

Beim Testen von verschiedenen Geräteeingabemethoden wird Folgendes überprüft:

- Da Mobiltelefone die Installation einer Vielzahl von Bildschirmtastaturen ermöglichen, funktioniert die App zumindest mit den von großen Geräteherstellern bereitgestellten und häufig verwendeten Bildschirmtastaturen.
- Die App stellt sicher, dass die Tastatur bei Bedarf standardmäßig mit dem entsprechenden Layout und den entsprechenden Tasten angezeigt wird.
- Wenn ein Benutzer den Touchscreen mit einem oder mehreren Fingern berührt, interpretiert die App dieses Muster als eine bestimmte Geste oder Befehl. Typische Gesten sind Berühren (Touch), Multitouch, Streichen bzw. Wischen, Tippen, Doppeltippen, Ziehen und Pinch zum Auf-/Zuziehen.
- Jeder Bildschirm der App muss korrekt auf die Gesten oder andere für diesen Bildschirm geeignete Eingabemethoden reagieren und alle nicht unterstützten Gesten oder Eingaben ignorieren.
- Von Apps verwendete Kameras können Bilder und Videos aufnehmen, Strichcodes, QR-Codes und Dokumente scannen und Entfernungen messen.
- Falls Front- und Rückkameras verfügbar sind, ist die entsprechende Kamera standardmäßig eingeschaltet. Falls für einen Video-Chat beispielsweise die Frontkamera standardmäßig eingeschaltet sein muss, müssen die Fälle getestet werden, in denen die App den Kameraeingang verwendet und in denen dies nicht der Fall ist. Darüber hinaus müssen Tests sicherstellen, dass die getestete Software ordnungsgemäß funktioniert, wenn nur eine Kamera (Front- oder Rückkamera) anstelle von zwei Kameras vorhanden ist. Dies gilt insbesondere dann, wenn die getestete Software eine bestimmte Kamera verwendet und genau diese fehlt.

2.1.6 Testen der Änderung der Bildschirmausrichtung

Bewegungssensoren werden verwendet, um Änderungen in der Ausrichtung zu erkennen und einen Wechsel zwischen Querformat und Hochformat (und umgekehrt) auszulösen, ggf. einhergehend mit einer Layoutänderung der Benutzungsschnittstelle.

Bei den Tests zur Änderung der Bildschirmausrichtung wird Folgendes überprüft:

- Korrekte Gebrauchstauglichkeit und Anwendungsverhalten, wenn auf Hochformat oder Querformat umgeschaltet wird.
- Die App behält ihren Zustand bei.
- Bereits erfasste Daten in Eingabedatenfeldern bleiben erhalten.
- Ausgabedatenfelder zeigen dieselben Daten an, solange die aktuelle Sitzung beibehalten wird.

Tests nach einer Änderung der Bildschirmausrichtung sollten sich nicht nur auf eine einzige Änderung der Bildschirmausrichtung konzentrieren, da Rendering- oder Statusprobleme möglicherweise nicht immer nach einer einzelnen Ausrichtungsänderung auftreten. Tests sollten daher mit mehreren unmittelbar aufeinanderfolgenden Änderungen zwischen Hoch- und Querformat durchgeführt werden.

Es sollten Tests entworfen werden, bei denen die Ausrichtung in den verschiedenen Zuständen einer Benutzungsschnittstelle - sowohl mit als auch ohne Daten - mehrmals geändert wird. Die App sollte sich wie erwartet verhalten und den Status beibehalten, ohne dass Daten verloren gehen oder sich ändern.

2.1.7 Testen von typischen Unterbrechungen

Häufige Arten von Geräteunterbrechungen sind Sprachanrufe, Nachrichten, eingeschaltete Ladegeräte, wenig Speicher und sonstige Arten von Benachrichtigungen. Benutzerinitiierte Unterbrechungen entstehen durch Aktionen wie das Wechseln der App oder das Versetzen des Geräts in den Standby-Modus, während die App ausgeführt wird.

Beim Testen von Unterbrechungen wird Folgendes überprüft:

- Die App verarbeitet alle oben genannten Unterbrechungen korrekt und das Verhalten der App wird durch diese nicht beeinträchtigt.
- Die App funktioniert weiterhin ordnungsgemäß und behält ihren Status, ihre Daten und Sitzungen bei, unabhängig davon, welche Art von Unterbrechung auftritt.
- Falls das Gerät über einen blockierenden Nicht-Stören-Modus verfügt, der die Benachrichtigungsfunktion unterdrückt, muss die App sicherstellen, dass die verschiedenen Bedingungen korrekt verwendet werden. Die Tests müssen auch überprüfen, ob sich die App korrekt verhält, wenn der Nicht-Stören-Modus ausgeschaltet wird, nachdem er über längere Zeit aktiviert war. Dies führt dazu, dass viele Benachrichtigungen gleichzeitig eingehen.
- Tests sollten für eingehende Unterbrechungen während der Nutzung der App ausgelegt sein, um sicherzustellen, dass die Unterbrechungen keine negativen Auswirkungen haben. Zum Beispiel: Testen, ob der Benutzer, der während der Verwendung der App einen Telefonanruf annimmt, danach wieder in den Zustand versetzt wird, in dem er sich zum Zeitpunkt der Unterbrechung befunden hat.

2.1.8 Testen von Zugriffsberechtigungen für Gerätefunktionen

Apps benötigen Zugriff auf verschiedene Ordner wie Kontakte und Bilder sowie auf Sensoren wie Kamera und Mikrofon. Wenn der Zugriff bei der Installation verweigert oder nach der Installation geändert wird, kann sich dies auf das Verhalten der App auswirken.

Beim Testen von Zugriffsberechtigungen wird Folgendes überprüft:

- Die App kann mit reduzierten Berechtigungen arbeiten; sie fordert den Benutzer auf, diese Berechtigungen zu erteilen, und schlägt nicht unerklärt fehl.
- Berechtigungen werden nur für die Ressourcen angefordert, die für die Funktionalität der App relevant sind. Pauschale Berechtigungen für Ressourcen, die in Zusammenhang mit der App nicht benötigt werden, sind nicht zulässig.
- Die App-Funktionalität reagiert korrekt, wenn eine Berechtigung während der Installation widerrufen oder abgelehnt wird.
- Jede von der App angeforderte Berechtigung ist korrekt und gerechtfertigt.

Um auf Zugriffsberechtigungen zu testen, müssen Tester wissen, warum die App die jeweilige Berechtigung benötigt und wie es sich auf die Funktionalität auswirken soll, wenn die Berechtigung während der Installation widerrufen oder abgelehnt wird. Der Testentwurf sollte vorsehen, dass Berechtigungen sowohl während der Installation abgelehnt also auch nach der Installation erteilt werden.

2.1.9 Testen von Stromverbrauch und Ladezustand

Beim Testen von Stromverbrauch und Ladezustand wird Folgendes überprüft:

- Akkuladezustand und entleerungsbedingte Fehler.
- Datenintegrität bei nahezu leerem und leerem Akku.
- Stromverbrauch, während die App aktiv ist und stark und wenig genutzt wird.
- Stromverbrauch, während die App im Hintergrund läuft.

Diese Tests müssen sorgfältig geplant werden, da sie über einen längeren Zeitraum ohne Unterbrechung ausgeführt werden müssen. Beispielsweise muss das Gerät mit der App im Hintergrund oder im Vordergrund unbeaufsichtigt gelassen werden, ohne dass das Gerät verwendet wird. Werkzeuge wie Protokoll-Analysatoren werden benötigt, um Informationen über die Batterieentladungsmuster zu gewinnen.

2.2 Testen von App-Interaktionen mit Gerätesoftware

2.2.1 Testen von Benachrichtigungen

Es gibt verschiedene Mechanismen, die vom Betriebssystem zur Anzeige von Benachrichtigungen verwendet werden. Manchmal verzögert das Betriebssystem die Anzeige von Benachrichtigungen oder zeigt sie überhaupt nicht an, um den Stromverbrauch zu optimieren. Beim Testen müssen die folgenden Testbedingungen berücksichtigt werden:

- Die korrekte Behandlung von empfangenen Benachrichtigungen, wenn sich die App im Vordergrund oder Hintergrund befindet, insbesondere bei schwachem Akku.
- Falls Benachrichtigungen eine direkte Interaktion mit dem App-Inhalt zulassen (d.h. ohne die App selbst zu öffnen), muss die Benutzerinteraktion zu einem späteren Zeitpunkt von der App bereitgestellt werden. Wenn der Benutzer beispielsweise auf eine Benachrichtigung antwortet, muss es zu einem späteren Zeitpunkt möglich sein, über die App auf diese Antwort zuzugreifen.
- Falls Benachrichtigungen den Zugriff auf die App ermöglichen, muss anstelle des Startbildschirms die entsprechende Seite der App geöffnet werden, sofern die Benachrichtigung einen Deeplink zu dieser Seite enthält.

2.2.2 Testen von Schnellzugriffsverknüpfungen

Möglicherweise werden Schnellzugriffsverknüpfungen wie App-Kurzbefehle in Android und Force Touch bzw. 3D Touch für iOS von der getesteten Software bereitgestellt. Diese Funktionen führen eine Teilmenge der Anwendungsfunktionen über den Startbildschirm aus, ohne die gesamte App zu starten.

Beim Testen müssen die folgenden Testbedingungen berücksichtigt werden:

- Wenn einige der Funktionen nur in einer bestimmten Version des Betriebssystems verfügbar sind, muss sich das zu testende System ordnungsgemäß verhalten, sofern es in Versionen des Betriebssystems mit oder ohne diese Funktionen installiert wird.
- Die in Schnellzugriffsverknüpfungen ausgeführten Aktionen werden beim Öffnen in der App korrekt wiedergegeben.

2.2.3 Testen von vom Betriebssystem bereitgestellten Benutzereinstellungen

Vom Betriebssystem bereitgestellte Einstellungen müssen getestet werden. Das Benutzererlebnis wird negativ beeinflusst, wenn eine bestimmte Voreinstellung von der App nicht beachtet wird. Wenn das Gerät beispielsweise stumm geschaltet ist, sollte die App keine Töne wiedergeben.

Beim Testen müssen die folgenden Testbedingungen berücksichtigt werden:

- Benutzer können typische Einstellungsoptionen wie Ton, Helligkeit, Netzwerk, Stromsparmodus, Datum und Uhrzeit, Zeitzone, Sprachen, Zugriffstyp und die Benachrichtigungsfunktion ändern.
- Die Apps halten sich an die voreingestellten Einstellungen und verhalten sich entsprechend.

2.2.4 Testen von verschiedenen Arten von Apps

Je nach Art der mobilen App (siehe Abschnitt 1.4) können spezifische Tests durchgeführt werden. Beim Testen müssen die folgenden Testbedingungen berücksichtigt werden:

- Für native Apps:
  - Gerätekompatibilität
  - Nutzung von Gerätefunktionen
- Für hybride Apps:
  - Interaktion der App mit den geräteeigenen Funktionen
  - Mögliche Leistungsprobleme aufgrund der Abstraktionsschicht
  - Gebrauchstauglichkeit (Look-and-Feel bzw. Erscheinungsbild) im Vergleich zu nativen Apps auf der jeweiligen Plattform
- Für Web-Apps:
  - Testen der browserübergreifenden Kompatibilität der App mit verschiedenen gängigen mobilen Browsern
  - Die Funktionalität wird durch verschiedene JavaScript-Engines nicht beeinträchtigt
  - Nutzung von Betriebssystemfunktionen (z.B. Datumsauswahl und Öffnen der entsprechenden Tastatur)
  - Gebrauchstauglichkeit (Look-and-Feel bzw. Erscheinungsbild) im Vergleich zu nativen Apps auf der jeweiligen Plattform

2.2.5 Testen der Interoperabilität mit mehreren Plattformen und Betriebssystemversionen

Softwareunternehmen unterstützen häufig Apps auf mehreren Betriebssystemen. Jedes mobile Betriebssystem hat seine eigenen Einschränkungen, die beim Testen von Apps berücksichtigt werden müssen. Die Tester müssen die Besonderheiten jeder getesteten Plattform kennen, um sicherzustellen, dass die App wie beabsichtigt funktioniert und dennoch dem Look-and-Feel der Plattform entspricht.

Beim Testen müssen die folgenden Testbedingungen berücksichtigt werden:

- Behandlung von Unterbrechungen, Benachrichtigungen und Optimierungen (z.B. zur Energieeinsparung).

- Korrekte Funktionalität, wenn Multiplattform-Applikationen Code gemeinsam nutzen oder mit plattformübergreifenden Entwicklungsframeworks erstellt wurden. Wenn die Applikationen keinen gemeinsamen Code haben, müssen zwei verschiedene Applikationen getestet werden; hierbei muss alles getestet werden.
- Testen der Abwärtskompatibilität, wenn eine Plattform unterschiedliche Betriebssystem-versionen verwendet.
- Testen neuer oder geänderter Funktionen von Plattformen. Beispielsweise erforderte die Einführung der Schlummerfunktion im Android-Framework das Testen der verschiedenen Versionen des Betriebssystems, die dieses Framework unterstützen, und derjenigen, die dies nicht unterstützen.

2.2.6 Testen der Interoperabilität und Koexistenz mit anderen Apps auf dem Gerät

Es ist durchaus üblich, dass Apps bei der Installation auf einem Gerät miteinander interagieren. Typische Beispiele sind die Kontakt- und E-Mail-Apps.

Beim Testen müssen die folgenden Testbedingungen berücksichtigt werden:

- Die Datenübertragung zwischen dem zu testenden System und der verwendeten App ist korrekt.
- Benutzerdaten, die in einer verwendeten App gespeichert sind, werden nicht beschädigt.
- Konfliktverhalten. Beispielsweise kann eine App GPS deaktivieren, um Energie zu sparen, während eine andere App GPS automatisch aktiviert.

Mit Millionen von Apps auf dem Markt kann die Koexistenz nicht für alle von ihnen realistisch getestet werden. Trotzdem sollten solche potenziellen Probleme berücksichtigt und auf ihr Risiko hin geprüft werden.

2.3 Testen verschiedener Verbindungsmethoden

Mobile Geräte können mit verschiedenen Methoden eine Verbindung zu Netzwerken herstellen (siehe Abschnitt 1.5). Hierzu zählen Mobilfunknetze wie 2G, 3G, 4G und 5G sowie WLAN und andere drahtlose Verbindungsmethoden wie die Nahfeldkommunikation (NFC) oder Bluetooth.

Beim Testen der Verbindungsfähigkeit sollten die folgenden Alternativen berücksichtigt werden:

- Geräteemulatoren bzw. -simulatoren können verschiedene Netzwerkverbindungen simulieren, und bei einigen Anbietern von Fernzugriffsdiensten ist dies in ihren Funktionen beinhaltet. Emulatoren/Simulatoren sind jedoch von begrenztem Wert.
- Einrichtung des eigenen Mobilfunknetzes, um verschiedene Verbindungsarten zu unterstützen, gefolgt von einer Bandbreitenmanipulation. Dies ist eine sehr kostspielige Alternative.
- Feldtests sind möglicherweise eine kostengünstigere Alternative; hierbei ist jedoch die Reproduzierbarkeit der Tests beschränkt.

In der Praxis gibt es unterschiedliche Verbindungsmethoden. Benutzer können ständig in einem bestimmten Modus verbunden sein oder zwischen verschiedenen Modi wechseln, z. B. vom WLAN zu Mobilfunk (z. B. wenn ein Benutzer während der Nutzung der App das Haus verlässt). Der Benutzer kann zwischen verschiedenen WLAN- bzw. Mobilfunknetzen und -versionen sowie zwischen Funkzellen wechseln. Unterwegs treffen die Benutzer möglicherweise sogar auf Funklöcher ohne Netzwerk. Darüber hinaus kann der Benutzer die Verbindung absichtlich trennen, indem er beispielsweise in den Flugmodus wechselt.

Das Testen der Verbindungsfähigkeit muss sicherstellen, dass die folgenden Testbedingungen berücksichtigt werden:

- Korrekte App-Funktionalität mit verschiedenen Verbindungsarten.
- Das Umschalten zwischen den Modi führt nicht zu unerwartetem Verhalten oder Datenverlust.
- Der Benutzer erhält klare Informationen, wenn die Funktionalität aufgrund einer eingeschränkten oder fehlenden Netzwerkverbindung eingeschränkt ist oder wenn die Bandbreite niedrig ist. In der Nachricht sollten die Einschränkungen und deren Gründe angegeben werden.

3. Häufig verwendete Testarten und Testprozesse für mobile Applikationen – 200 Minuten

Schlüsselbegriffe

Abbruch, Barrierefreiheit, Code-Einschleusung, exploratives Testen, Feldtest, Gebrauchstauglichkeits-labor, Gebrauchstauglichkeitstest, Heuristik, Installierbarkeit, IT-Sicherheitstest, Performanz, Performanztest, Post-Release-Testen, Tour, sitzungsbasiertes Testmanagement, Stresstest, Testprozess, Testpyramide, Teststufe

Lernziele für häufig verwendete Testarten und Testprozesse für mobile Applikationen

3.1 Häufig verwendete Testarten für mobile Applikationen

MAT-3.1.1 (K3) Sie können Installierbarkeitstests für mobile Apps erstellen.
MAT-3.1.2 (K3) Sie können Stresstests für mobile Apps erstellen.
MAT-3.1.3 (K2) Sie können Beispiele für Sicherheitsprobleme im Zusammenhang mit mobilen Apps nennen.
MAT-3.1.4 (K1) Sie können Überlegungen zum Zeit- und Ressourcenverhalten mobiler Apps wiedergeben.
MAT-3.1.5 (K3) Sie können Gebrauchstauglichkeitstests für mobile Apps erstellen.
HO-3.1.5 (H2) Auswahl einer Tour, einer Merkhilfe oder einer Heuristik zum Testen der Gebrauchstauglichkeit einer App innerhalb des sitzungsbasierten Testmanagements.
Hinweis: HO-3.1.5 und HO-3.3.1, HO-3.3.2 und HO-3.3.3 können kombiniert werden.
MAT-3.1.6 (K1) Sie können erkennen, welche Arten von Tests zum Testen von Datenbanken für mobile Apps erforderlich sind.
MAT-3.1.7 (K2) Sie können die für das Testen der Internationalisierung (Globalisierung) und Lokalisierung mobiler Apps erforderlichen Tests zusammenfassen.
MAT-3.1.8 (K2) Sie können die Notwendigkeit des Testens der Barrierefreiheit beim Testen mobiler Applikationen zusammenfassen.

3.2 Zusätzliche Teststufen beim Testen mobiler Applikationen

MAT-3.2.1 (K2) Sie können die zusätzlichen Teststufen wie z.B. den Feldtest und die damit verbundenen zusätzlichen Aktivitäten beschreiben, die für ein effektives Testen mobiler Applikationen erforderlich sind.
MAT-3.2.2 (K2) Sie können die Tests beschreiben, die für die Durchführung der App-Store-Zulassung für das Veröffentlichen von Apps erforderlich sind.

3.3 Erfahrungsbasierte Testverfahren

MAT-3.3.1 (K1) Sie können wiedergeben, was mit sitzungsbasiertem Testmanagement, Personas und Merkhilfen in Zusammenhang mit explorativem Testen mobiler Applikationen gemeint ist.
HO-3.3.1 (H2) Sie können eine Merkhilfe (oder einen Teil davon) auswählen, die für das Testen mobiler Applikationen spezifisch ist, um eine App mithilfe des sitzungsbasierten Testmanagements zu testen.
Hinweis: HO-3.1.5 und HO-3.3.1, HO-3.3.2 und HO-3.3.3 können kombiniert werden.
MAT-3.3.2 (K2) Sie können die Verwendung von Touren und Heuristiken als explorative Testverfahren für das Testen mobiler Applikationen beschreiben.
HO-3.3.2 (H2) Sie können eine spezifische Heuristik auswählen, um eine mobile Applikation zu testen.
Hinweis: HO-3.1.5 und HO-3.3.1, HO-3.3.2 und HO-3.3.3 können kombiniert werden.
MAT-3.3.3 (K3) Sie können eine spezifische Tour (wie z. B. eine Feature Tour) verwenden, um eine mobile App zu testen.
HO-3.3.3 (H2) Sie können eine spezifische Tour auswählen, um eine mobile App zu testen.

Hinweis: HO-3.1.5 und HO-3.3.1, HO-3.3.2 und HO-3.3.3 können kombiniert werden.

3.4 Testprozess und Testvorgehensweisen beim Testen mobiler Applikationen

MAT-3.4.1 (K2) Sie können den Testprozess, wie im Lehrplan [ISTQB_CTFL_2018] beschrieben, an die Anforderungen des Testens mobiler Applikationen anpassen.

MAT-3.4.2 (K2) Sie können die Testvorgehensweisen für jede Teststufe beschreiben, die für das Testen mobiler Applikationen spezifisch sind.

3.1 Häufig verwendete Testarten für mobile Applikationen

3.1.1 Installierbarkeitstests

Tester müssen im Test den Schwerpunkt auf die Installation, Aktualisierung und Deinstallation der App legen und dabei die folgenden Vorgehensweisen anwenden:

- App-Stores

  Der Installationsvorgang kann je nach Benutzer der App unterschiedlich sein. Die Nutzer können die App von Online-Marktplätzen wie dem Google Play Store oder dem App-Store von Apple installieren. Die Benutzer von Unternehmens-Apps müssen Installationstests über einen Link oder über Dienste wie HockeyApp oder App Center durchführen.

- Sideloading (App kopieren und installieren)

  Einige Betriebssysteme bieten die Möglichkeit, die App zu installieren, indem sie auf ein mobiles Gerät kopiert und aus der Datei heraus installiert wird.

- Desktop-Applikationen

  Für die Installation von Apps auf dem Smartphone stehen Desktop-Applikationen wie Apple iTunes (für iOS) oder der Android App Installer zur Verfügung. Der Tester muss die App in dieser Desktop-Applikation herunterladen und mit einem Kabel von dort auf dem Smartphone installieren. Die meisten dieser Desktop-Applikationen ermöglichen auch die Deinstallation der App.

Die Installation kann mit den folgenden Methoden durchgeführt werden:

- Per Funkübermittlung (OTA/Over-the-Air) über WLAN oder mobile Datenverbindung
- Über Datenkabel

Testbedingungen, die beim Testen berücksichtigt werden können, umfassen:

- Installation, Deinstallation und Aktualisierung auf internem und externem Speicher (falls unterstützt).
- Erneute Installation der App, wenn bei der vorherigen Deinstallation die Option „App-Daten beibehalten“ ausgewählt wurde.
- Erneute Installation der App, wenn bei der vorherigen Deinstallation die Option „App-Daten beibehalten“ nicht ausgewählt wurde.
- Abbrechen oder Unterbrechen der Installation oder Deinstallation, z.B. durch Herunterfahren des mobilen Endgeräts während des Vorgangs oder Trennen der Verbindung zum Internet.
- Wiederaufnahme der unterbrochenen Installation, Deinstallation und Aktualisierung nach Abbruch oder Unterbrechung.
- Berechtigungsbezogenes Testen. Einige Apps fordern beispielsweise die Berechtigung zur Verwendung des Adressbuchs an. Dieser wichtige Test muss das Verhalten der App überprüfen, wenn der Benutzer diese Berechtigung verweigert. Wird beispielsweise eine entsprechende Nachricht an den Benutzer gesendet?
- Beim Aktualisieren der App ist sicherzustellen, dass keine Daten verloren gehen.

Einige Apps erfordern Geräte mit Jailbreaking (iOS) oder Rooting (Android), die dem Benutzer Administratorrechte für das Gerät gewähren. Die meisten Plattformanbieter unterstützen Jailbreaking bzw. Rooting nicht, da dies rechtliche Konsequenzen haben kann. Eine App, die ohnehin kein Jailbreaking/Rooting erfordert, muss nicht zwingend auf Geräten mit Jailbreaking bzw. Rooting getestet werden.

3.1.2 Stresstests

Stresstests konzentrieren sich auf die Bestimmung der Performanz der Applikation unter Bedingungen, die über die normale Belastung hinausgehen. Der Stresstest ist in diesem Zusammenhang nur auf das Mobilgerät ausgerichtet. Stresstests des Backendsystems sind im ISTQB® Lehrplan Performanztester ([ISTQB_CTFL_PT_2019]) beschrieben, wo bei Bedarf weiterführende Informationen zu finden sind.

Testbedingungen, die bei den Stresstests berücksichtigt werden können, umfassen:

- hohe CPU-Auslastung
- zu wenig Speicher
- wenig Speicherplatz
- Batteriestress
- Ausfälle
- geringe Bandbreite
- sehr viele Benutzerinteraktionen (möglicherweise müssen dazu reale Netzwerkbedingungen simuliert werden)

Einige dieser Stressbedingungen können mit Werkzeugen wie Monkey erstellt werden. Dies ist ein Befehlszeilenwerkzeug, das über die ADB-Shell-Befehlszeile [URL3] ausgeführt wird. Falls möglich kann man diese Stressbedingungen auch manuell herstellen, z.B. mithilfe großer Dateien oder anderer Apps mit hoher CPU-Auslastung oder hohem Speicherverbrauch.

3.1.3 IT-Sicherheitstests

Da IT-Sicherheitstests ein komplexes Thema sind, trägt das ISTQB® mit dem Advanced Level Specialist-Lehrplan [ISTQB_CTAL_SEC_2016] diesem Thema Rechnung. Zu den wichtigsten Sicherheitsproblemen für mobile Apps gehören:

- Zugriff auf sensible Daten auf dem Gerät.
- Unverschlüsselte Datenübertragung oder unsichere Speicherung.

Testbedingungen, die bei den IT-Sicherheitstests berücksichtigt werden können, umfassen:

- Testeingaben für Code-Einschleusung und Pufferüberlauf.
- Verschlüsselung der übertragenen Daten.
- Verschlüsselung lokal gespeicherter Daten.
- Löschen temporärer Daten nach Verwendung oder nach einem Abbruch.
- Löschen von Text in Passwortfeldern.

Die 10 wichtigsten Sicherheitsschwachstellen im Zusammenhang mit Mobilgeräten gemäß OWASP-Liste (Open Web Application Security Project) sollten ebenfalls untersucht werden [URL2].

3.1.4 Performanztest

Wenn Benutzer die App installieren und sie nicht schnell genug gestartet wird (z.B. nach maximal 3 Sekunden), wird sie möglicherweise zugunsten einer anderen alternativen App deinstalliert. Aspekte

bzgl. Zeit- und Ressourcenverbrauch sind wichtige Erfolgsfaktoren für eine App. Um diese zu messen, werden Performanztests durchgeführt.

Die Performanz muss zusätzlich zur Interaktion mit dem Backendsystem und mit anderen Mobilgeräten auf dem Gerät selbst getestet werden.

Die Performanztests des gesamten Systems sollten gemäß der Teststrategie durchgeführt werden; diese sind nicht spezifisch für mobile Apps. Weitere Informationen finden Sie im ISTQB®-Lehrplan für Performanztester [ISTQB_CTFL_PT_2018].

Der Performanztest der App selbst sollte die Zeitmessung für die wichtigsten Abläufe enthalten. Einige Beispiele für die Abläufe einer Online-Banking-App sind: „Login“, „Adresse ändern“ oder „Überweisung mit PIN und TAN“. Der Tester sollte diese Zeitmessungen dann mit ähnlichen Apps vergleichen.

Neben den Zeitmessungen ist es wichtig, die vom Benutzer wahrgenommene Performanz zu berücksichtigen. Das Nutzungserlebnis kann einen großen Einfluss darauf haben, wie lange der Benutzer bereit ist, auf den Abschluss einer bestimmten Funktion zu warten.

3.1.5 Gebrauchstauglichkeitstest

Die Gebrauchstauglichkeit ist für mobile Apps sehr wichtig. Analysen zeigen, dass eine große Zahl von Benutzern Apps innerhalb weniger Minuten nach der Installation aufgrund schlechter Gebrauchstauglichkeit oder Performanz wieder deinstalliert [URL4].

Aus diesem Grund wird empfohlen, dass das Design der User Experience (UX) das Look-and-Feel der Plattform berücksichtigt, auf der die App verwendet werden soll. Wenn die User Experience nicht den Erwartungen des Benutzers an die Plattform seiner Wahl entspricht, kann dies sehr negative Auswirkungen haben. Daher sollten Tester das Look-and-Feel der verwendeten Plattform kennen.

Für die Durchführung der Gebrauchstauglichkeitstests können verschiedene verfügbare Heuristiken und Touren verwendet werden. Die Berücksichtigung von Personas ist bei Gebrauchstauglichkeitstests ebenfalls hilfreich. Bei Bedarf kann für die Tests zudem ein Gebrauchstauglichkeitslabor eingesetzt werden.

Bei den in Gebrauchstauglichkeitstests identifizierten Ergebnissen handelt es sich zumeist nur um Befunde und nicht um Fehler. Tester müssen in der Lage sein, die Befunde dem Team, dem Product Owner oder ähnlichen Stakeholdern zu erläutern. Um eine zufriedenstellende Gebrauchstauglichkeit zu erzielen, sollten Apps:

- selbsterklärend und intuitiv sein.
- Benutzerfehler zulassen.
- in der verwendeten Terminologie und im Verhalten konsistent sein.
- die Designrichtlinien der Plattformen beachten.
- die benötigten Informationen in jeder Bildschirmgröße und jedem Bildschirmtyp sichtbar und zugänglich machen.

Weitere Informationen finden Sie im ISTQB®-Lehrplan für Usability Testing [ISTQB_FLUT_2018].

3.1.6 Datenbanktests

Viele Apps müssen Daten mithilfe verschiedener Datenspeichermechanismen lokal speichern, z.B. in flachen Dateien oder in Datenbanken. Zu den Testbedingungen, die beim Testen von Datenbanken für mobile Apps zu berücksichtigen sind, gehören:

- Validierung von Aspekten in Zusammenhang mit der Datenspeicherung:
  - Synchronisierung
  - Konflikte beim Hochladen
  - Datensicherheit
  - Einschränkungen der Daten

- CRUD-Funktionalität, d.h. die vier grundlegenden Operationen **C**reate (Datensatz anlegen) **R**ead (Datensatz lesen) **U**pdate (Datensatz aktualisieren) und **D**elete (Datensatz löschen)
- Suchfunktion

- Testen der Datenintegration in Bezug auf Daten, die vom Gerät (z.B. Kontakte) oder von Apps von Drittanbietern (z.B. Bilder, Videos und Nachrichten) bereitgestellt werden.
- Performanz beim Speichern der Daten auf dem Gerät.

3.1.7 Globalisierungs- and Lokalisierungstests

Die Internationalisierungstests (I18n) bzw. Globalisierungstests von Applikationen umfassen das Testen von Apps für verschiedene Standorte, Formate für Datumsangaben, Zahlen und Währungen sowie das Ersetzen tatsächlicher Zeichenfolgen durch Pseudozeichenfolgen.

Das Testen der Lokalisierung (L10n) umfasst das Testen einer App mit lokalisierten Zeichenfolgen, Bildern und Bedienungsabläufen für eine bestimmte Region. Beispielsweise können russische und deutsche Wörter viel länger sein als in anderen Sprachen. Da mobile Geräte unterschiedliche Bildschirmgrößen und Auflösungen haben, können begrenzte Bildschirmgrößen zu Problemen mit übersetzten Zeichenfolgen führen. Diese Probleme sollten bei den Globalisierungs-/ Lokalisierungstests standardmäßig überprüft werden.

Ein sehr wichtiger Aspekt, der überprüft werden muss, ist das verwendete Datumsformat, z.B. JAHR - MONAT - TAG oder TAG - MONAT - JAHR.

3.1.8 Testen der Barrierefreiheit

Tests der Barrierefreiheit werden durchgeführt, um festzustellen, wie einfach Benutzer mit Behinderungen eine Komponente oder ein System verwenden können. Für mobile Apps erfolgt dies über die Geräteeinstellungen zur Zugänglichkeit, und durch Testen der App mit jeder dieser Einstellungen.

Richtlinien zur Barrierefreiheit sind bei Plattformanbietern erhältlich und sollten verwendet werden. Beispielsweise haben sowohl Google [URL5] als auch Apple [URL6] Richtlinien zur Barrierefreiheit für ihre jeweiligen Plattformen veröffentlicht. Hilfreich sind auch Rückmeldungen von Personen, die Barrierefreiheit benötigen.

Für das mobile Web wurde vom W3C ein Leitfaden zur Barrierefreiheit veröffentlicht, der berücksichtigt werden sollte [URL7].

3.2 Zusätzliche Teststufen beim Testen mobiler Applikationen

Zusätzlich zu den im ISTQB®-Lehrplan [ISTQB_CTFL_2018] beschriebenen üblichen Teststufen vom Komponenten- bis zum Abnahmetest sind für das Testen mobiler Applikationen zusätzliche Teststufen erforderlich.

3.2.1 Feldtest

Einige mobile Applikationen müssen in einem Feldtest getestet werden, um sicherzustellen, dass sie im erwarteten Nutzungsszenario realer Benutzer korrekt funktionieren. Dies kann das Testen in verschiedenen Netzwerken und mit verschiedenen Arten von Kommunikationstechnologien wie WLAN oder mobile Daten umfassen.

Feldtests sollten die Verwendung von Mobilfunkmasten, Netzwerken, WLAN und das Umschalten von Mobilfunkdaten während der Nutzung der App umfassen. Tests sollten mit unterschiedlichen Download-Geschwindigkeiten und Signalstärken durchgeführt werden und den Umgang mit Funklöchern einschließen.

Feldtests erfordern eine sorgfältige Planung und die Identifizierung aller für die Durchführung der Tests erforderlichen Elemente, wie z. B. geeignete Gerätetypen, WLAN, mobile Datentarife/Daten-

angebote verschiedener Betreiber und verschiedene Verkehrsmittel, um eine angemessene Abdeckung zu gewährleisten. Darüber hinaus müssen die Transportwege und -arten sowie die Uhrzeit, zu der die Tests durchgeführt werden sollen, geplant werden.

Ein weiterer wichtiger Aspekt, der bei der Durchführung von Feldtests berücksichtigt werden muss, ist die Gebrauchstauglichkeit der App. Die Tests sollten in Zusammenhang mit dem Nutzungsszenario Umgebungsfaktoren wie Temperatur und ähnliche Bedingungen berücksichtigen.

3.2.2 Testen der App-Store-Zulassung und Post-Release-Testen

Bevor eine App veröffentlicht wird, müssen einige Checklisten-basierte Tests bestanden werden, um die Zulassung der App-Stores zu gewährleisten. Wenn es sich bei der Version um eine Aktualisierung handelt, sollten auch Tests mit Bezug auf die Aktualisierung ausgeführt werden.

Checklisten basieren in der Regel auf Richtlinien, z.B. spezielle Richtlinien für Betriebssysteme, für das Design der Benutzungsoberfläche und für die Verwendung der von App-Stores bereitgestellten Bibliotheken und APIs.

Der Zulassungsprozess kann nach der Beantragung einige Zeit dauern. Falls während des Zulassungsprozesses Probleme festgestellt werden, muss möglicherweise eine neue Version eingereicht werden, was zusätzlich Zeit in Anspruch nimmt. Dies muss bei der Projektplanung und beim Testen sorgfältig abgewogen werden.

Eine weitere Teststufe ist das Post-Release-Testen. Das Testen auf dieser Teststufe beinhaltetet das Herunterladen und Installieren der Applikation aus App-Stores.

3.3 Erfahrungsbasierte Testverfahren

3.3.1 Personas und Merkhilfen

Personas sind fiktive Personen, die echte Kunden repräsentieren. Sie haben Motivationen, Erwartungen, Probleme, Gewohnheiten und Ziele. Außerdem ist es hilfreich Personas zu verwenden, wenn echtes Benutzerverhalten nachgeahmt werden muss.

Eine Persona kann einen Namen, ein Geschlecht, ein Alter, ein Einkommen, einen Bildungshintergrund und einen Standort haben. In einem mobilen Kontext können sie andere Apps verwenden, ihr mobiles Gerät x-mal pro Stunde überprüfen und auch andere Geräte und persönliche Merkmale aufweisen.

Eine Merkhilfe (oder Mnemonik) ist eine Gedächtnishilfe, um sich an etwas zu erinnern. Im Kontext des Testens steht jeder Buchstabe in einer Merkhilfe für eine Technik, eine Testmethode oder einen Testschwerpunkt. Ein Beispiel für eine Merkhilfe ist *SFiDPOT* [URL8]. Die Buchstaben in dieser Merkhilfe haben die folgenden Bedeutungen:

S - Structure (Strukturen, z.B. Elemente einer Benutzungsschnittstelle, andere Applikationselemente und deren Reihenfolge und Aufrufhierarchie)

F - Function (Funktionen, z.B. die gewünschten Features funktionieren, sind verfügbar, funktionieren gemäß den Anforderungen usw.)

i – Input (Eingaben, z.B. alle erforderlichen Eingaben (über Tastatur, Sensoren und Kamera) sind verfügbar und werden ordnungsgemäß verarbeitet)

D - Data (Daten, z.B. die Daten werden gespeichert (auch auf SD-Speicherkarte), geändert, hinzugefügt und gelöscht, wie in den Anforderungen definiert)

P - Platform (Plattform, z.B. die spezifischen Betriebssystemfunktionen sind abhängig von den Geräteeinstellungen verfügbar, einschließlich Speicher zum Herunterladen der App)

O - Operations (Operationen, z.B. die Aktivitäten des normalen Benutzers sind verfügbar, z.B. das Wechseln zwischen Mobilfunknetzen und WLAN)

T – Time (z.B. Behandlung und Anzeige von Zeitzonen, Uhrzeit und Datumsangaben)

Eine Merkhilfe bzw. Heuristik speziell in der mobilen Welt ist *I SLICED UP FUN* [URL9]. Die Buchstaben in dieser Merkhilfe haben folgende Bedeutung:

I – Inputs (Eingaben)

S – Store (App-Store)

L – Location (Standort)

I – Interactions and Interrupts (Aktionen und Unterbrechungen)

C – Communication (Kommunikation)

E – Ergonomics (Ergonomie)

D – Data (Daten)

U – Usability (Gebrauchstauglichkeit)

P – Plattform (Plattform)

F – Function (Funktion)

U – User Scenarios (Nutzerszenarien)

N – Network (Netzwerk)

3.3.2 Heuristiken

Ein heuristischer Ansatz ist eine Faustregel für das Lösen, Lernen und Erkennen von Problemen, bei dem eine praktische Methode angewendet wird. Dies garantiert nicht, dass dies optimal oder perfekt ist, kann jedoch als ausreichend angesehen werden, um die unmittelbaren Ziele zu erreichen.

Es gibt viele Heuristiken für das Testen mobiler Applikationen. Die meisten Merkhilfen können als Heuristik verwendet werden, aber nicht jede Heuristik ist eine Merkhilfe.

3.3.3 Touren

Touren werden beim explorativen Testen verwendet, um Applikationen aus einem bestimmten Blickwinkel und mit bestimmtem Fokus zu erkunden. Sie können durchgeführt werden, um die Funktionsweise einer Applikation zu verstehen und um Benutzerabläufe zu modellieren. Touren sind auch eine effektive Methode für Feldtests.

Das Beispiel einer Tour zu bekannten Sehenswürdigkeiten einer Stadt, die von einem Touristen nacheinander besucht werden, dient als Analogie für die Schritte, die beim Testen mobiler Applikationen ausgeführt werden. Diese Analogien sind in der folgenden Tabelle dargestellt:

Tour (Stadtbesichtigung)	Analogie für das Testen mobiler Applikationen
Die historische Altstadt	Legacy-Code
Das Geschäftsviertel Die Hauptverkehrszeit	Geschäftslogik der App Starten und Beenden der App
Das Touristenviertel	Teile der App, die von neuen Nutzern verwendet werden
Das Hotelviertel	Teile der App, die nur im Schlafmodus aktiv sind

Die Kombination von sitzungsbasiertem Testen (Session Based Test Management, siehe Abschnitt 3.3.4) mit Touren, einschließlich der Verwendung von Heuristiken und Merkhilfen, trägt zur Verbesserung der Effektivität beim Testen mobiler Applikationen bei.

Die folgende Tabelle zeigt einige gute Beispiele für Touren zum Testen von Apps und der abgedeckten Bereiche, und soll Ideen für das Testen liefern. Einige davon sind in [Kohl17] zu finden.

Tour zum Testen mobiler Apps	Abgedeckte Themen
Supermodell	Erscheinungsbild und Gebrauchstauglichkeit
Sehenswürdigkeit	Die wichtigsten Funktionen der App
Sabotage	Robustheit
Feature	Neue Features
Szenario	Gesamter Bedienungsablauf in der App in Kombination mit Szenarien
Verbindungsfähigkeit	Verbindungsfähigkeit, z.B. WLAN, GSM
Standort	Korrekte Sprache, Datumsformat, Zahlen
Licht	Sichtbarkeit bei unterschiedlichen Lichtverhältnissen wie Dunkelheit, im Freien, bei Rotlicht
Schwacher Akku	Datenverluste in der App aufgrund niedriger Ladezustände
Weitere Touren zum Testen mobiler Apps	Abgedeckte Themen (siehe [Kohl17])
Geste	Verwendung aller Gesten (wo immer möglich)
Ausrichtung	Änderung der Ausrichtung
Sinneswandel	Zurück
Bewegung	Verschiedene Bewegungen ausführen
Standort	Sich umherbewegen
Verbindungsfähigkeit	Die Verbindungsmethoden bzw. die Standorte wechseln
Vergleich	Vergleich mit anderen Gerätetypen
Konsistenz	Überprüfung der Konsistenz von Bildschirmen und grafischer Benutzungsoberfläche (GUI)

3.3.4 Sitzungsbasiertes Testmanagement (Session Based Test Management, SBTM)

Das sitzungsbasierte Testmanagement (Session Based Test Management, SBTM) ermöglicht eine zeitbegrenzte („time-boxed“) Ausführung des explorativen Testens. Eine Sitzung besteht aus drei Aufgaben:

- Einrichtung der Sitzung
- Testentwurf und -ausführung
- Problemanalyse und Auswertungsberichte

Beim SBTM werden normalerweise Sitzungsnotizen mit einer Testcharta verwendet, die die Testziele vorgibt. Außerdem dienen die Sitzungsnotizen dazu, die durchgeführten Testausführungsaktivitäten zu dokumentieren.

Das explorative Testen ist ein erfahrungsbasiertes Testverfahren, was ein effektiver Ansatz zum Testen mobiler Applikationen sein kann. Erfahrungsbasierte Testverfahren sind in [ISTQB_CTFL_2018] beschrieben.

3.4 Testprozess und Testvorgehensweisen beim Testen mobiler Applikationen

3.4.1 Testprozess

Die Hauptaktivitäten des ISTQB®-Testprozesses sind in [ISTQB_CTFL_2018] beschrieben und gelten auch für das Testen mobiler Applikationen.

Es gibt zusätzliche Aspekte, die für das Testen mobiler Applikationen spezifisch und grundsätzlich als Teil des ISTQB®-Testprozesses zu verstehen sind.

Hauptaktivitäten im Testprozesses	Typische Bereiche beim Testen mobiler Applikationen	Verweis auf Lehrplan
Testplanung	• Bestimmung der zu testenden Gerätekombinationen • Verwendung von Emulatoren und Simulatoren der mobilen Geräte als Teil der Testumgebung • Besondere Herausforderungen beim Testen mobiler Apps • Speziell für das Testen mobiler Applikationen erforderliche Testarten	• Abschnitt 4.3 • Abschnitt 1.7 • Abschnitt 3.2
Testanalyse und Testentwurf	• Testen der App-Store-Zulassung • Feldtest • Gerätekompatibilität • Art der Labore für den Test • Speziell für das Testen mobiler Applikationen erforderliche Testarten	• Abschnitt 3.2.2 • Abschnitt 3.2.1 • Abschnitt 3.2
Testrealisierung und Testausführung	• Feldtest • Herunterladen und Installierbarkeit im Rahmen von Post-Release-Tests • Erfahrungsbasierte Testverfahren	• Abschnitt 3.2.1 • Abschnitt 3.1.1 [ISTQB_CTFL_2018]
Testrealisierung und Testausführung	• Die Tests basieren auf Plattform-Richtlinien für die Benutzungsschnittstelle und die App-Stores. • Die auf Richtlinien basierenden Tests werden normalerweise von den Plattformanbietern für ihren App-Store-Zulassungsprozess durchgeführt. • Es wird Applikationsanbietern empfohlen, diese Tests vor der Übergabe der App an die Plattformanbieter auszuführen, um eine mögliche Ablehnung zu vermeiden.	

3.4.2 Testvorgehensweisen

Das Testen mobiler Applikationen umfass Aktivitäten, die von Entwicklern und Testern gleichermaßen durchgeführt werden müssen.

Die Bestimmung der geeigneten Testtiefe für die einzelnen Teststufen (d.h. für Komponententest, Integrationstest, Systemtest, Feldtest, App-Store-Zulassungstest, Post-Release- und Benutzerabnahmetest) ist wichtig für die Bereitstellung qualitativ hochwertiger Produkte. Die erforderliche Testtiefe einer Teststufe hängt von vielen Faktoren ab, z.B. von der Architektur der App, von ihrer Komplexität und von der Zielgruppe der Benutzer.

Entwicklungsplattformen für mobile Applikationen bieten eine Vielzahl von Werkzeugen, um Tests auf den verschiedenen Teststufen zu unterstützen. Es ist sehr wichtig, die Werkzeuge zu verstehen und wie diese auf einer bestimmten Teststufe angewendet werden können. Beispielsweise können Simulatoren und/oder Emulatoren eines Mobilgeräts auf der Komponententeststufe verwendet werden, wenn die von der Plattform bereitgestellten Framework- und Instrumentierungs-APIs genutzt werden müssen. Außerdem können Simulatoren und/oder Emulatoren eines Mobilgeräts auf der

Systemteststufe verwendet werden, wenn keine echten Geräte verfügbar sind. Dies ermöglicht das Testen der Funktionalität, eine begrenzte Auswahl an Aspekten der Gebrauchstauglichkeit sowie der Benutzungsschnittstelle.

Darüber hinaus kann eine frühzeitige Implementierung ein zentraler Gesichtspunkt sein, um sicherzustellen, dass die Geräte korrekt eingerichtet sind und alle Voraussetzungen für die Ausführung rechtzeitig erfüllt werden.

Unit- und Integrationstests sind ebenso wichtig wie manuelle Tests (insbesondere in der Feldtestphase). Beim Testen mobiler Applikationen wird die Testpyramide häufig umgedreht [Knott15]. Dies bedeutet, dass es viele manuelle Tests geben kann.

4. Plattformen, Werkzeuge und Umgebung mobiler Applikationen – 80 Minuten

Schlüsselbegriffe

Emulator, entferntes Testlabor, Feldtest, Simulator, umgebungsbasiertes Testen

Lernziele für Plattformen, Werkzeuge und Umgebung mobiler Applikationen

4.1 Entwicklungsplattformen für mobile Applikationen

MAT-4.1.1 (K1) Sie können die Entwicklungsumgebungen wiedergeben, die für die Entwicklung mobiler Applikationen verwendet werden.

4.2 Gängige Werkzeuge von Entwicklungsplattformen

MAT-4.2.1 (K1) Sie können einige der gängigen Werkzeuge wiedergeben, die von Plattformen zur Anwendungsentwicklung bereitgestellt werden.

HO-4.2.1 (H1) Sie können mit den vom Software Development Kit bereitgestellten Werkzeugen Screenshots erstellen, ein Protokoll extrahieren und eingehende Ereignisse simulieren.

4.3 Emulatoren & Simulatoren

MAT-4.3.1 (K2) Sie verstehen die Unterschiede zwischen Emulatoren und Simulatoren.

MAT-4.3.2 (K2) Sie können die Verwendung von Emulatoren und Simulatoren für das Testen mobiler Applikationen beschreiben.

HO-4.3.2 (H1) Sie können ein simuliertes/emuliertes Gerät einrichten und laufen lassen, eine App installieren und einige Tests ausführen.

4.4 Einrichten eines Testlabors

MAT-4.4.1 (K2) Sie können zwischen verschiedenen Ansätzen zum Einrichten eines Testlabors unterscheiden.

4.1 Entwicklungsplattformen für mobile Applikationen

Auf dem Markt sind integrierte Entwicklungsumgebungen (engl. Integrated Development Environments, IDEs) für die unterschiedliche Entwicklung mobiler Apps verfügbar. Diese IDEs verfügen über verschiedene Werkzeuge, mit deren Hilfe Apps entworfen, programmiert, kompiliert, installiert, deinstalliert, überwacht, emuliert, protokolliert und getestet werden können.

Für die Entwicklung mobiler Apps können beispielsweise Android Studio (für Android Apps) und Xcode (für iOS Apps) verwendet werden. Diese unterscheiden sich von gewöhnlichen IDEs durch die zusätzliche Unterstützung, die sie für die mobilen Plattformen bieten.

Es stehen auch einige plattformübergreifende Entwicklungs-Frameworks zur Verfügung, die bei der Entwicklung mobiler Applikationen helfen. Diese laufen auf mehreren Plattformen und erfordern nicht das Schreiben von plattformspezifischem Code.

4.2 Gängige Werkzeuge von Entwicklungsplattformen

Software Development Kits stellen üblicherweise verschiedene Hilfsprogramme zur Verfügung, die beim Entwickeln und Testen von Applikationen hilfreich sind. Diese Hilfsprogramme umfassen einen breiten Anwendungsbereich, wie z. B. das Erstellen von Screenshots, Extrahieren von Protokollen,

Senden von zufälligen Ereignissen und Benachrichtigungen an das Gerät, Überwachen verschiedener Parameter wie Speicher- und CPU-Auslastung und Einrichten virtueller Geräte.

Einige Beispiele für solche Werkzeuge sind Android Virtual Device (AVD) Manager, Android Debug Bridge (ADB) und Android Device Monitor für Android und Instruments für iOS.

4.3 Emulatoren & Simulatoren

4.3.1 Übersicht über Emulatoren & Simulatoren

Im Kontext dieses Lehrplans beziehen sich die Begriffe Emulator und Simulator auf Emulatoren und Simulatoren mobiler Endgeräte. Die Begriffe Simulator und Emulator werden manchmal synonym und somit falsch verwendet. Definitionen entnehmen Sie bitte dem Glossar in Kapitel 8.

Ein Simulator modelliert die Laufzeitumgebung, während ein Emulator die Hardware modelliert und dieselbe Laufzeitumgebung wie die physische Hardware verwendet. Auf einem Simulator getestete Applikationen werden zu einer eigenen (dedizierten) Version kompiliert, die auf dem Simulator, jedoch nicht auf einem realen Gerät funktioniert. Somit ist diese vom realen Betriebssystem unabhängig.

Im Gegensatz dazu werden Applikationen, die kompiliert wurden, um auf einem Emulator bereitgestellt und getestet zu werden, in den tatsächlichen Bytecode kompiliert, der auch vom echten Gerät verwendet werden kann.

Simulatoren und Emulatoren sind in der frühen Entwicklungsphase sehr nützlich, da sie normalerweise in die Entwicklungsumgebungen integriert sind und eine schnelle Bereitstellung, Testen und Überwachung von Applikationen ermöglichen.

Beim Testen werden Simulatoren manchmal auch als Ersatz für echte Geräte verwendet. Dies ist jedoch noch eingeschränkter als die Verwendung von Emulatoren, da sich die auf einem Simulator getestete Applikation auf Bytecode-Ebene von der Applikation unterscheidet, die real genutzt wird..

Emulatoren werden auch genutzt, um bei einigen Tests echte Geräte zu ersetzen und damit die Kosten für Testumgebungen zu senken. Ein Emulator kann ein Gerät nicht vollständig ersetzen, da sich der Emulator möglicherweise anders verhält als das Mobilgerät, das er versucht nachzuahmen. Darüber hinaus werden einige Funktionen wie (Multi-)Touchfunktionen, Beschleunigungsmesser und andere möglicherweise nicht unterstützt. Dies liegt zum Teil an den Einschränkungen der Plattform, auf welcher der Emulator ausgeführt wird.

4.3.2 Verwendung von Emulatoren und Simulatoren

Die Verwendung von Emulatoren und Simulatoren für das Testen mobiler Applikationen kann aus verschiedenen Gründen hilfreich sein.

Jede Entwicklungsumgebung für mobile Betriebssysteme enthält normalerweise ihren eigenen Emulator und Simulator. Darüber hinaus stehen Emulatoren und Simulatoren von Drittanbietern zur Verfügung.

Tester können den für ihren Zweck geeigneten Emulator oder Simulator verwenden. Um den Emulator oder Simulator zu verwenden, muss dieser gestartet werden, die erforderliche App muss auf ihnen installiert werden und wird dann so getestet, als ob sie sich auf dem echten Gerät befände.

In der Regel lassen sich verschiedene Nutzungsparameter auf den Emulatoren und Simulatoren einstellen. Diese Einstellungen können die Netzwerkemulation mit unterschiedlichen Geschwindigkeiten, Signalstärken und Paketverlusten, das Ändern der Ausrichtung, das Generieren von Unterbrechungen und die GPS-Standortdaten umfassen. Für einige dieser Einstellungen, beispielsweise bei GPS-Standortdaten oder Signalstärken, kann dies sehr nützlich sein, da diese auf realen Geräten schwierig oder kostspielig zu replizieren sind.

Das Herstellen einer Verbindung zu den Emulatoren zu Installationszwecken erfordert möglicherweise die Verwendung von Befehlszeilenwerkzeugen, wie z. B. Android Debug Bridge (ADB) für Android,

oder die Herstellung der Verbindung über die integrierte Entwicklungsumgebung, wie bei Xcode oder Android Studio.

4.4 Einrichtung eines Testlabors

Für die Einrichtung eines Testlabors für mobile Applikationen werden die folgenden Ansätze verwendet:

Vor-Ort-Labor

Bei einem Vor-Ort-Labor befinden sich alle Geräte, Emulatoren und Simulatoren vor Ort. Die Geräteauswahl kann auf der Grundlage verschiedener Faktoren erfolgen, wie z. B. Rang des Geräts in der Rangliste (die in Google oder anderen Analysen zu finden ist), Betriebssystem und Betriebssystemversionen, Bildschirmabmessungen und Pixeldichte, Verfügbarkeit und Kosten, Besonderheiten und Wichtigkeit des Geräts in der Zielgruppe.

Zu den Vorteilen eines Vor-Ort-Labors gehört die Verfügbarkeit von Geräten für spezifische umgebungsbasierte Tests und sensorspezifische Aspekte wie Batterie, Touch und verbesserte Sicherheit.

Für die Einrichtung dieser Art von Testlabor sind möglicherweise hohe Budgets erforderlich, je nachdem welche Geräte beschafft und gewartet werden müssen. Zusätzliche Herausforderungen sind die rechtzeitige Verfügbarkeit und Schwierigkeiten beim Testen an verschiedenen Standorten und in verschiedenen Umgebungen.

Entferntes Testlabor

Diese Testlabore sind wichtig und für das Testen hilfreich, wenn Geräte oder Netzwerke vor Ort physisch nicht verfügbar sind. Der Fernzugriff auf Geräte ermöglicht den Zugriff über eine Netzwerkverbindung auf verschiedene Geräte, die sich im Rechenzentrum des Anbieters befinden. Jeder potenzielle Anbieter muss im Hinblick auf die Einhaltung von Anforderungen, insbesondere im Hinblick auf die Sicherheit, evaluiert werden.

Einige entfernte Testlabore bieten zusätzlich Folgendes:

- Spezielle Versionen physischer Geräte (z.B. Samsung Testlabor für Mobilgeräte).
- Generische Geräte ausschließlich für ein bestimmtes Betriebssystem und Betriebssystem-version.
- Roboterarme zur Ausführung von berührungs- und gestenbezogenen Bedienfunktionen.
- VPN-Verbindungen (Virtual Private Network), um Zugriff auf das Gerät zu gewähren.
- Mobilfunkverbindungen mit verschiedenen Mobilfunknetzanbietern.
- Automatisierungswerkzeuge und -dienste.

Zu den Faktoren, die bei der Verwendung von entfernten Testlaboren berücksichtigt werden müssen, gehören eine langsame Reaktionszeit der Geräte und eingeschränkte Optionen für die Interaktion mit Geräten wie z. B. Multitouch- und Gestensteuerung. Dies kann bei einer sporadischen Verwendung kostengünstig sein, ist jedoch im Allgemeinen teurer, wenn das Testlabor über einen längeren Zeitraum für eine Vielzahl von Geräten verwendet wird.

Weitere Faktoren sind die Verfügbarkeit der Plattform bei Bedarf im Vergleich zur Notwendigkeit, auf fehlende Geräte im Vor-Ort-Labor zugreifen zu können, sowie die Skalierbarkeit des Labors, da es im Verlauf des Projekts wachsen und schrumpfen kann.

Testszenarien mit Sensoren für NFC/Bluetooth oder Batterieverbrauch lassen sich in der Cloud oft nur schwer testen. Die unterschiedlichen geografischen Standorte der entfernten Testlabore können jedoch bei Tests hilfreich sein, die Netzwerk- und GPS-Verbindungen erfordern.

Ein Testlabor kann je nach Art der durchzuführenden Tests entweder einen der Ansätze oder eine Kombination beider Ansätze verwenden.

5. Automatisierung der Testausführung – 55 Minuten

Schlüsselbegriffe

API, gerätebasiertes Testen, Testbericht, User-Agent-basiertes Testen

Lernziele für Automatisierung der Testausführung

5.1 Automatisierungsansätze

MAT-5.1.1 (K2) Sie können zwischen gängigen Automatisierungsansätzen und -Frameworks für das Testen mobiler Applikationen unterscheiden.

5.2 Automatisierungsmethoden

MAT-5.2.1 (K2) Sie können verschiedene Automatisierungsmethoden für das Testen mobiler Applikationen beschreiben.

5.3 Bewertung von Automatisierungswerkzeugen

MAT-5.3.1 (K1) Sie können die verschiedenen Parameter wiedergeben, die bei der Bewertung von Automatisierungswerkzeugen für das Testen mobiler Applikationen berücksichtigt werden müssen.

5.4 Vorgehensweisen zum Einrichten eines Automatisierungstestlabors

MAT-5.4.1 (K2) Sie können zwischen gängigen Vorgehensweisen zur Einrichtung von Testlaboren mit den jeweiligen Vor- und Nachteilen in Bezug auf die Testautomatisierung unterscheiden.

5.1 Automatisierungsansätze

Es gibt verschiedene Automatisierungsansätze und -Frameworks, die für das Testen mobiler Applikationen verwendet werden können. Die Wahl des Ansatzes wird zum Teil von der Art der Applikation bestimmt.

Zwei gebräuchliche Testautomatisierungsansätze sind:

- User-Agent-basiertes Testen
- Gerätebasiertes Testen

Beim User-Agent-basierten Testen wird die vom Browser gesendete User-Agent-Kennung verwendet, um einem bestimmten Browser auf einem bestimmten Gerät vorzutäuschen. Dieser Ansatz kann zum Ausführen von mobilen Web-Applikationen verwendet werden. Beim gerätebasierten Testen wird die getestete Applikation direkt auf dem Gerät ausgeführt. Dieser Ansatz kann für alle Arten von mobilen Applikationen verwendet werden.

Der Applikationstyp kann auch das Testautomatisierungs-Framework bestimmen, das für die Applikation geeignet wäre. Mobile Webapplikationen können mit den üblichen Automatisierungswerkzeugen für Desktop Webapplikationen getestet werden, während für das Testen nativer Apps möglicherweise spezielle Werkzeuge erforderlich sind. Auch Plattformanbieter können spezielle Automatisierungswerkzeuge für ihre Plattform bereitstellen.

Automatisierungsansätze, die für herkömmliche Anwendungen verwendet werden, sind häufig auch auf mobile Applikationen anwendbar. Hierzu zählen Mitschnitt-, datengetriebenes, schlüsselwortgetriebenes und verhaltensgetriebenes Testen, wie im ISTQB® Foundation Level Lehrplan

[ISTQB_CTFL_2018] und im ISTQB® Advanced Level Specialist Lehrplan Test Automation Engineer [ISTQB_CTAL_TAE_2019] beschrieben.

Die wichtigsten Fähigkeiten, die ein Testframework für mobile Applikationen normalerweise umfassen sollte, sind:

- Objektidentifizierung
- Objektoperationen
- Testberichte
- Application Programming Interfaces (APIs) und erweiterbare Funktionen
- Angemessene Dokumentation
- Integration mit anderen Werkzeugen
- Unabhängigkeit von der Testentwicklung

5.2 Automatisierungsmethoden

Um automatisierte Tests zu entwickeln, müssen Tester den Mechanismus zur Aufzeichnung oder Erstellung von Automatisierungsskripten verstehen und wissen, wie der Zugriff auf Objekte der Applikation wie z. B. Schaltflächen, Tabellen und Eingabefelder und die Interaktion mit diesen erfolgt.

Es gibt verschiedene Methoden, um ein Objekt zu identifizieren, das für die Testautomatisierung mobiler Apps verwendet wird. Dazu gehören Bilderkennung, optische Zeichenerkennung (OCR), sowie die Objekterkennung (web oder nativ, abhängig von der Art der Applikation).

Tester mobiler Applikationen müssen nicht nur die Objekterkennung und -identifizierung üben, sondern sie müssen auch wissen, welche Objektidentifizierungsmethode am besten geeignet ist, damit die Tests, die auf einer Vielzahl von Mobilgeräten parallel und kontinuierlich ausgeführt werden, erfolgreich sein werden.

Die wichtigsten Unterschiede zwischen den Methoden zur Skripterstellung sind:

Vergleichs-gegenstand	Objektidentifizierung	Bildvergleich
Zuverlässigkeit	Solange die Kennung konstant ist, kann das Bildschirmlayout geändert werden. Das Risiko besteht darin, dass Objekte im Code identifiziert und bearbeitet werden können, obwohl sie für den Benutzer nicht sichtbar sind. Dies kann zu falsch negativen Testergebnissen führen.	Bilder können entsprechend der Bildschirmgröße skaliert werden, die Tests schlagen jedoch fehl, sobald sich das Layout ändert.
Benutzererlebnis	In der Regel ist manuelles Skripten erforderlich, zumindest um die Lesbarkeit und Wartbarkeit aufgezeichneter Skripte zu verbessern.	Vollständiges GUI-Testen, kein Skripten erforderlich.
Ausführungs-geschwindigkeit der Skripte	Ist tendenziell schneller als der Bildvergleich insbesondere wenn native Werkzeuge des Systemherstellers verwendet werden.	Ist tendenziell langsamer, da der Bildschirm Pixel für Pixel mit einem Musterbild verglichen werden muss.
Wartung der Skripte	Ist von der Qualität der Testskripte abhängig.	Hauptsächlich für die Bereitstellung geänderter Vergleichsbilder.
Herausforderung bei	Kenntnisse der Skriptsprache und	Generierung von Vergleichsbildern,

der Skripterstellung	von Softwareentwicklungsmethoden zur Erstellung einer nachhaltigen Automatisierungslösung sind erforderlich.	insbesondere wenn sich die App häufig ändert.

5.3 Bewertung von Automatisierungswerkzeugen

Für die Erstellung erfolgreicher Testautomatisierungslösungen müssen die Testautomatisierungsteams geeignete Werkzeuge auswählen. Für den Auswahlprozess ist ein Verständnis der wichtigsten Unterschiede zwischen den verfügbaren Werkzeugen und deren Eignung für die spezifischen Projektanforderungen erforderlich (siehe auch ISTQB® Advanced Level Specialist Lehrplan Test Automation Engineer [ISTQB_CTAL_TAE_2019]).

Die Bewertungsparameter für Testautomatisierungswerkzeuge lassen sich in zwei Kategorien unterteilen:

- Eignung der Organisation
- Eignung der Technik

Die Bewertungsparameter, die die Organisation betreffen, sind in Kapitel 6.2 des ISTQB® Foundation Level Lehrplans [ISTQB_CTFL_2018] beschrieben.

Die Bewertungsparameter, die die Technik betreffen, umfassen:

- Anforderungen und Herausforderungen bzgl. der Testautomatisierung, z.B. die Verwendung neuer Funktionen wie FaceID, Fingerabdruck und Chatbots durch die App.
- Anforderungen der Testumgebung, wie z.B. unterschiedliche Netzwerkbedingungen, Importieren oder Erstellen von Testdaten und serverseitige Virtualisierung.
- Funktionen für die Erstellung von Testberichten und Feedbackschleifen.
- Die Fähigkeit des Frameworks, die Ausführung in großem Umfang lokal oder in einem Testlabor in der Cloud zu verwalten und zu steuern.
- Integration des Testframeworks mit anderen in der Organisation verwendeten Werkzeugen.
- Verfügbarkeit von Support und Dokumentation für aktuelle und zukünftige Upgrades.

5.4 Vorgehensweisen zum Einrichten eines Automatisierungstestlabors

Beim Testen mobiler Applikationen haben Entwickler und Tester Wahlmöglichkeiten in Bezug auf das Testlabor mit Mobilgeräten, das für ihre Testautomatisierung zum Einsatz kommt.

- Vor-Ort-Labor (On-Promise Lab)
- Entferntes Testlabor (sog. Remote Testlab)

Es ist auch eine Kombination dieser beiden Ansätze möglich. Ihre Hauptmerkmale werden in Abschnitt 4.4 dieses Lehrplans beschrieben und verglichen.

Vor-Ort-Labore sind im Allgemeinen schwierig und zeitaufwendig zu warten. Für die frühen Entwicklungs- und Testphasen von mobilen Apps ist es am besten, die Geräte vor Ort zu haben und parallel zu Emulatoren und Simulatoren zu verwenden.

Im fortgeschritteneren Stadium der App-Entwicklung müssen die Teams vollständige Regressionstests, sowie funktionale Tests und nicht-funktionale Tests durchführen. Diese Tests werden am besten in einem Testlabor mit umfassender Mobilgeräteausstattung durchgeführt. Hierfür wird ein entferntes Testlabor für Mobilgeräte in der Cloud verwaltet, kontinuierlich aktualisiert und gewartet. Solche entfernten Testlabore für Mobilgeräte sind eine ideale Ergänzung zum Vor-Ort-Labor und stellen sicher, dass ausreichend viele Kombinationen von Mobilgeräten und Betriebssystemen verfügbar sind, und dass diese auf dem aktuellsten Stand sind. Durch die Nutzung solcher allgemein

verfügbarer Testlabore für Mobilgeräte haben Teams Zugriff auf eine größere Anzahl unterstützter Funktionen, einschließlich umfangreicherer Testberichtsmöglichkeiten und erweiterter Funktionen zur Testautomatisierung.

Letztendlich ist die Stabilität des gesamten Testlabors entscheidend für die Effizienz und Zuverlässigkeit von Tests, wenn diese mithilfe eines Testautomatisierungs-Frameworks oder mittels kontinuierlicher Integration in großem Maßstab ausgeführt werden. Solche Labore sind in der Regel so konzipiert, dass Geräte und Betriebssysteme immer verfügbar und stabil sind.

In späteren Entwicklungsphasen der App sind entfernte Mobilgeräte-Testlabore nicht immer erforderlich. Gut gestaltete und gewartete Vor-Ort-Labore für Mobilgeräte können genauso gut oder besser sein als entfernteTestlabore.

6. Referenzen

6.1 ISTQB®-Dokumente

Alle hier referenzierten ISTQB-Dokumente gibt es sowohl in Englisch (Original) als auch in Deutsch (Übersetzung und Herausgabe durch die ISTQB-Mitgliedsbords GTB, ATB und STB). Sie sind zum Beispiel über die Webseite des German Testing Board,

url: https://www.german-testing-board.info/

kostenlos zu beziehen.

- [ISTQB_CTFL_2018]: ISTQB® Certified Tester – Foundation Level Syllabus – Version 2018
- [ISTQB_FLAT_2017]: ISTQB® Certified Tester – Foundation Level Extension Syllabus – Agile Tester – Version 2017 (deutschsprachig)
- [ISTQB_FLUT_2018]: ISTQB® Certified Tester – Foundation Level Specialist Syllabus – Usability Testing – Version 2018
- [ISTQB_CTFL_PT_2018]: ISTQB® Certified Tester – Foundation Level Specialist Syllabus – Performance Testing – Version 2018
- [ISTQB_CTAL_SEC_2019]: ISTQB® Certified Tester – Advanced Level Specialist Syllabus – Security Testing –
  Version 2019
- [ISTQB_CTAL_TAE_2019]: ISTQB® Certified Tester – Advanced Level Specialist Syllabus - Test Automation Engineer - Version 2019
- [ISTQB_GLOSSARY]: ISTQB® Glossary of Terms used in Software Testing, Version 3.2

6.2 Referenzierte Bücher (englischsprachig)

- [Knott15] Knott, D., “Hands-On Mobile App Testing”, Addison-Wesley Professional, 2015, ISBN 978-3-86490-379-3
- [Kohl17] Kohl, J. , “Tap into mobile application testing”, leanpub.com, 2017, ISBN 978-0-9959823-2-1

6.3 Weitere Bücher und Artikel (englischsprachig)

- Boris Beizer, “Black-box Testing”, John Wiley & Sons, 1995, ISBN 0-471-12094-4
- Rex Black, “Agile Testing Foundations”, BCS Learning & Development Ltd: Swindon UK, 2017, ISBN 978-1-78017-33-68
- Rex Black, “Managing the Testing Process”(3e), John Wiley & Sons: New York NY, 2009, ISBN 978-0-470-40415-7
- Hans Buwalda, “Integrated Test Design and Automation”, Addison-Wesley Longman, 2001, ISBN 0-201-73725-6
- Lee Copeland, “A Practitioner's Guide to Software Test Design”, Artech House, 2003, ISBN 1-58053-791-X
- Rick David Craig, Stefan P. Jaskiel, “Systematic Software Testing”, Artech House, 2002, ISBN 1-580-53508-9

6.4 Links (Web/Internet)

Haftungsausschluss für Verlinkungen (Links geprüft am 01. Juli 2019)

- [URL1] http://gs.statcounter.com/
- [URL2] www.owasp.org
- [URL3] https://developer.android.com/studio/test/monkey
- [URL4] https://www.google.de/amp/s/techcrunch.com/2016/05/31/nearly-1-in-4-people-abandon-mobile-apps-after-only-one-use/amp/
- [URL5] https://www.google.com/accessibility/
- [URL6] https://www.apple.com/accessibility/
- [URL7] https://www.w3.org/WAI/standards-guidelines/mobile/
- [URL8] https://www.slideshare.net/karennjohnson/kn-johnson-2012-heuristics-mnemonics
- [URL9] http://www.kohl.ca/articles/ISLICEDUPFUN.pdf

7. Anhang A – Lernziele/Kognitive Stufen des Wissens

Die folgende Taxonomie für Lernziele bildet die Grundlage des Lehrplans. Jeder Inhalt wird entsprechend der zugeordneten Lernziele geprüft.

7.1 Taxonomiestufe 1: Kennen (K1)

Der Lernende kann einen Begriff oder ein Konzept erkennen, abrufen und sich daran erinnern.

Schlüsselbegriffe: identifizieren, erinnern, abfragen, abrufen, wiedergeben, erkennen, kennen.

Beispiele:

MAT-1.3.1 (K1) Sie können verschiedene Arten von Mobilgeräten aufzählen.

7.2 Taxonomiestufe 2: Verstehen (K2)

Der Lernende begründet oder erläutert Aussagen zum Thema. Typische beobachtbare Leistungen sind beschreiben, zusammenfassen, vergleichen, klassifizieren, begründen, erklären, Beispiele nennen.

Schlüsselbegriffe: zusammenfassen, verallgemeinern, abstrahieren, klassifizieren, vergleichen, auf etwas übertragen, etwas gegenüberstellen, erläutern, Beispiele geben, interpretieren, übersetzen, darstellen, rückschließen, folgern, kategorisieren, Modelle bilden.

Beispiele:

MAT-1.1.1 (K2) Sie können beschreiben, wie die Analyse verfügbarer mobiler Nutzungsdaten als Eingabe für die Teststrategie und das Testkonzept verwendet werden kann.

7.3 Taxonomiestufe 3: Anwenden (K3)

Der Lernende überträgt erworbenes Wissen auf gegebene neue Situationen oder wendet es zur Problemlösung an.

Schlüsselbegriffe: anwenden, einsetzen, nutzen, nach einem Verfahren vorgehen, ein Verfahren anwenden

Beispiele:

MAT-1.6.1 (K3) Sie können bei der Erstellung einer Teststrategie die Merkmale und Besonderheiten der Mobilgerätewelt anwenden.

Referenz (für die kognitiven Stufen der Lernziele):

Anderson, L. W. and Krathwohl, D. R. (eds) (2001) A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Allyn& Bacon: Boston MA

8. Anhang B – Glossar domänenspezifischer Begriffe

Glossarbegriff	Definition
2G	Mobilfunktechnologie der 2. Generation.
3D Touch	siehe “Force Touch”
3G	Mobilfunktechnologie der 3. Generation.
4G	Mobilfunktechnologie der 4. Generation.
5G	Mobilfunktechnologie der 5. Generation.
Abwärtskompatibilität	Die Fähigkeit von Apps, auf früheren Versionen einer Plattform zu laufen.
ADB	Android Debug Bridge (ADB) - Befehlszeilenwerkzeug, das die Kommunikation mit einem Gerät ermöglicht.
Android Device Monitor (ADM)	Ein eigenständiges Werkzeug, das eine Benutzungsschnittstelle für Werkzeuge zum Debuggen und Analysieren von Android-Apps bietet.
Android Studio	Die offizielle integrierte Entwicklungsumgebung (IDE) für Android. Android Studio bietet Werkzeuge zum Erstellen von Apps auf allen Arten von Android-Geräten.
anzeigengestützte Apps	Ein Finanzierungsmodell für Apps, bei dem die Entwicklungsorganisationen durch in der App angezeigte Werbung Geld verdienen.
App Kurzbefehl	Eine Verknüpfung zu einem bestimmten Satz von Aktionen, die von Anwendungsentwicklern in Apps mit Android-Version 7.1 oder höher definiert wurde.
App-Daten beibehalten	Vom Benutzer generierte Daten und/oder Inhalte der App bleiben auf dem Gerät erhalten, wenn die App deinstalliert wird, damit andere Apps darauf zugreifen können, oder falls dieselbe App erneut installiert wird.
App-Store	Eine Vertriebsplattform für Applikationen, auf der Entwickler ihre Apps hochladen und Benutzer nach Apps suchen können, um sie herunterzuladen und auf ihrer Plattform zu installieren.
asynchrone Kommunikation	Eine Art der Kommunikation, bei der Daten nicht in einem stetigen Strom, sondern zeitlich versetzt übertragen werden können.
Ausrichtung	Die Platzierung eines Objekts auf dem Mobiltelefon, die angibt, wie das Gerät verwendet wird. Die Ausrichtung kann entweder im Querformat oder Hochformat sein.
AVD	Akronym für Android Virtual Device.
Backendsystem	Ein Serversystem, das Funktionen für andere Systeme bereitstellt.
Basistelefon	Ein Mobiltelefon mit minimalen Funktionen wie Telefonieren, Speichern von Telefonnummern, Senden von SMS, Uhrzeit und Wecker.
Begleitgerät	Ein Rechner oder Gerät, das für die Zusammenarbeit mit einem dazugehörigen Smart Device entwickelt wurde.
Benachrichtigung	Eine Mitteilung, die vom Gerät ausgesendet wird.
Bibliothek	Eine Sammlung von Ressourcen, die von Computerprogrammen verwendet werden, d.h. von Funktionen der App.
Bildschirmtastatur	Eine als Software realisierte virtuelle Tastatur, die dem Benutzer auf einem Display angezeigt wird.
Bluetooth	Eine drahtlose Kommunikationstechnologie für den Nahbereich.
Bytecode	Ein Befehlssatz, der zur effizienten Ausführung durch einen Software-Interpreter entwickelt wurde. Wird auch als portabler Code bzw. P-Code bezeichnet.
CPU-Frequenz	Die Taktfrequenz des Prozessors.

Glossarbegriff	Definition
CRUD	Merkhilfe für „Create/Read/Update/Delete“. Die Begriffe beziehen sich auf die Art des Datenzugriffs (Erstellen/Lesen/Ändern/Löschen).
Datenintegrität	Die Genauigkeit und Konsistenz von Daten über den gesamten Lebenszyklus, einschließlich der Speicherung, Verarbeitung und Abruf der Daten.
Datensynchronisierung	Der Vorgang, bei dem Daten aus zwei oder mehr Quellen in denselben Zustand gebracht werden.
Datenvalidierung	Die Überprüfung, ob Daten korrekt, genau, konsistent und verwendbar sind.
DPI / PPI	Akronym für Dots per Inch / Pixels per Inch (Punkte pro Zoll / Pixel pro Zoll) - eine Zahl, die die Punkt- bzw. Pixeldichte einer Anzeige angibt.
Drittanbietermarktplatz	Eine App-Vertriebsplattform, die nicht von einem Plattformanbieter betrieben wird.
einschichtig	Ein Entwicklungsansatz für Backend-Applikationen, bei dem ein einziger Server alle für eine Applikation benötigten Dienste bereitstellt.
Einstellungen	Die allgemeinen Konfigurationsparameter von Geräten oder Applikationen, die vom Benutzer geändert werden können.
Emulator	Eine Softwareanwendung, die das Verhalten von Hardware nachahmt.
externer Speicher	Ein zusätzlicher Speicher, der einem Gerät über eine Standardschnittstelle hinzugefügt wird. Derzeit sind für Mobiltelefone SD-Karten am weitesten verbreitet.
Fat-Client	Ein Client in Client-Server-Applikationen, der dafür entwickelt wurde, einen Teil oder den größten Teil der Datenverarbeitung zu übernehmen.
Feature-Phone	Eine Klasse von Mobiltelefonen, die zwar mehr Funktionen als ein Basistelefon bieten, wie z. B. einen Browser, jedoch nicht die volle Funktionalität eines Smartphones.
Fernzugriff auf Geräte	Interaktion mit einem Gerät, das sich physisch an einem anderen Ort als der Benutzer befindet, normalerweise über das Internet.
flache Datei	Eine Datei ohne interne Hierarchie.
Flugmodus	Ein spezieller Betriebsmodus für mobile Endgeräte, bei dem die Funksender deaktiviert sind, um Interferenzen mit Flugbetriebs-/Kommunikationssystemen zu vermeiden.
Force Touch	Eine von Apple Inc. entwickelte Technologie, mit der Trackpads und Touchscreens unterscheiden können, wie viel Kraft auf ihre Oberflächen ausgeübt wird.
Freemium-App	Ein Geschäftsmodell, bei dem Benutzer nichts für das Herunterladen der App bezahlen und optionale In-App-Käufe angeboten werden.
Funkloch	Ein Ort ohne Funkempfang.
Funkzelle	Ein Teil eines GSM-Netzwerks, der durch seine eindeutige Standortkennung identifiziert werden kann.
Gerätefragmentierung	Die Vielfalt verfügbarer Geräte mit ihrer geräteeigenen Hardwarekonfiguration sowie der Auswirkungen auf die Apps und das Benutzererlebnis.
Geste	Ein bestimmtes Interaktionsmuster, z. B. ein Wischen oder ein Pinch (Auf-/Zuziehen), um bestimmte Funktionen des Geräts zu aktivieren. Beispielsweise wird üblicherweise “pinch-to-zoom” verwendet, um die angezeigten Inhalte zu vergrößern oder zu verkleinern.
Globalisierung	Siehe Internationalisierung.

Glossarbegriff	Definition
GPS	Akronym für Global Positioning System - ein Netzwerk von Satelliten, die weltweit Zeitsignale aussenden. Durch Einbeziehen des Signals von mindestens 3 Satelliten kann ein Empfänger seine relative Position zu den Satelliten durch Triangulation (Dreieckmessung) berechnen.
Größe des Anzeigebereichs	Eine virtuelle Bildschirmgröße, mit welcher der Browser das Layout an den Bildschirm anpasst.
GSM	GSM ist die Abkürzung für Global System for Mobile Communications (ursprünglich Groupe Spécial Mobile), ein Standard, der vom European Telecommunications Standards Institute (ETSI) entwickelt wurde, um die Protokolle für digitale Mobilfunknetze der zweiten Generation zu beschreiben, die von Mobilgeräten verwendet werden. Derzeit der weltweit gebräuchlichste Standard für mobile Kommunikation.
Hochformat	Die Geräteausrichtung, in der die Anzeigehöhe größer als die -breite ist.
hybride App	Eine Applikation, die native und Webtechnologien kombiniert verwendet. Normalerweise verwendet eine hybride App ein natives Framework, das auf dem Gerät installiert ist, um mit den Gerätebibliotheken und Ähnlichem zu interagieren. Zusätzlich werden Inhalte angezeigt, die von einem Webserver empfangen werden.
I18N	Numeronym (Zahlenwort) für “internationalization” (Internationalisierung).
IDE	Integrierte Entwicklungsumgebung - Eine Softwareanwendung, die Programmierern umfassende Funktionen für die Softwareentwicklung bietet.
In-App-Kauf	Zusätzliche Inhalte und Funktionen, die direkt in einer App gekauft werden können.
Instruments	Ein Werkzeug für Leistungsanalysen und Performanztests, das im Entwicklungswerkzeug Xcode enthalten ist.
Internationalisierung	Der Prozess der Vorbereitung einer Applikation für verschiedene lokalisierte Versionen.
interner Speicher	Ein Speicher, der in der Gerätehardware integriert ist.
IoT-Gerät	Internet of Things (Internet der Dinge). Ein mit dem Internet verbundenes Gerät oder beispielsweise ein Sensor, der von Interesse ist.
Jailbreaking	Eine Previlegienerweiterung zum Entfernen von Softwarebeschränkungen, die vom Betriebssystem vorgegeben sind. Der Begriff wird normalerweise für iOS verwendet. Ähnlich zum Rooten bei Android.
Konflikt beim Hochladen	Ein Fehler beim Versuch, eine Datei hochzuladen, die bereits am Ziel vorhanden ist.
kostenpflichtige App	Eine App, die sich durch den Verkauf in App-Stores finanziert.
L10N	Numeronym (Zahlenwort) für “localization” (Lokalisierung).
Ladezustand	Ein vom Benutzer definiertes oder vordefiniertes Profil in Bezug auf den Stromverbrauch, das auf einem mobilen Gerät aktiviert werden kann.
Laufzeitumgebung	Implementierung des Ausführungsmodells. Auch als Laufzeitsystem bekannt.
Lokalisierung	Der Prozess zur Anpassung einer App oder eines Produkts an eine bestimmte Region durch Aktionen wie Übersetzung und Angleichung an bestimmte Formate.

Glossarbegriff	Definition
Look-and-Feel/Erscheinungsbild	Das Erscheinungsbild bzw. der visuelle und emotionale Eindruck von etwas.
mehrschichtig	Ein Entwicklungsansatz für Backend-Applikationen, bei dem zwei oder mehr Server spezielle Funktionen bereitstellen.
Merkhilfe	Eine Gedächtnisstütze, Mnemonik.
mobile Plattform	Ein Ökosystem rund um ein mobiles Betriebssystem, das normalerweise Entwicklungswerkzeuge, das Betriebssystem selbst und einen Vertriebskanal für die App umfasst.
mobiler Gerätetyp	Eine Klassifizierung mobiler Geräte nach ihren Grundfunktionen. Zu den gängigen Klassen gehören Basistelefone, Feature-Phones, Smartphones, Phablets, Tablets und Wearables.
mobiles Betriebssystem	Ein Betriebssystem, das speziell für mobile Geräte entwickelt wurde.
Mobilfunkdaten	Daten, die über ein Mobilfunknetz übertragen werden.
Mobilfunknetz	Ein Mobilfunknetz ist ein Netzwerk, das aus mehreren unabhängigen, aber verbundenen Zellen besteht.
Mobilgeräte-Emulator	Virtuelle Abbildung einer Hardwareplattform. Beispielsweise ist der Android-Emulator eine virtuelle Hardware, auf der ein reales Android-Betriebssystem-Abbild ausgeführt wird. Genau dasselbe Betriebssystem-Abbild kann auf der Hardware bereitgestellt werden und wird lauffähig sein, da es sich um das echte Betriebssystem handelt.
Mobilgeräte-Simulator	Eine virtuelle Laufzeitumgebung. Beispielsweise gibt der iOS-Simulator vor, iOS zu sein, ist jedoch kein echtes iOS.
Mobilgerätewelt	Ein umfassender Begriff, der alles in Bezug auf die Mobilgeräte-Technologie umfasst, vom Markt und seinen Playern bis zu den Geräten und Apps.
Multiplattformapplikation	Applikationen, die für die Ausführung auf mehreren Plattformen bestimmt sind und mit einer einheitlichen Codebasis für alle Plattformen entworfen und entwickelt wurden.
Multitouch	Eine Art der Interaktion mit einem Gerät, bei der mehrere Berührungen gleichzeitig ausgeführt werden
native App	Eine speziell für eine bestimmte Plattform entwickelte Applikation, für die normalerweise von der Plattform bereitgestellte APIs und Entwicklungswerkzeuge verwendet werden.
NFC	Akronym für Near Field Communication (Nahfeldkommunikation) – eine Kommunikationstechnologie, die den Nahbereichsfunk nutzt.
Nicht-Stören-Modus	Ein Betriebsmodus für mobile Endgeräte, der vom Benutzer aktiviert werden kann, um bestimmte Funktionen (allgemeine Benachrichtigungen und Sprachanrufe) zu blockieren.
OCR	Akronym für Optical Character Recognition (Optische Zeichenerkennung). Die Erkennung von Textbildern in einem elektronischen Bild und deren Umwandlung in maschinencodierten Text.
OTA	Akronym für Over the air (Funkübermittlung). Datenübertragung über Funksignale, die üblicherweise in Zusammenhang mit der Installation von Apps auf Geräten verwendet wird, die direkt von einer Quelle erfolgt, die nicht über ein Kabel angeschlossen ist.
Persona	Ein Modell / Archetyp für eine bestimmte Benutzergruppe.
plattformübergreifendes Entwicklungsframework	Ein Framework zum Entwickeln einer App für verschiedene Plattformen mit derselben Codebasis.
QR-Code	QR-Code (Abkürzung für Quick Response Code) ist das Markenzeichen für eine Art Matrix-Strichcode (oder zweidimensionalen Strichcode).

Glossarbegriff	Definition
Querformat	Die Geräteausrichtung, in der die Anzeigebreite größer als die -höhe ist.
Rooten	Der Vorgang, der darauf abzielt, Root-Zugriff auf das Betriebssystem des Geräts zu erlangen. Der Begriff wird normalerweise im Zusammenhang mit der Android-Plattform verwendet. Ähnlich zum Jailbreaking bei iOS.
Seitenverhältnis	Das Verhältnis von Breite zu Höhe einer Anzeige oder eines Bildes.
sensible Daten	Daten, die besonders schützenswert sind, wie z. B. Passwörter und personenbezogene Daten.
Sensor	Ein Gerät, Modul oder Subsystem, dessen Zweck darin besteht, Ereignisse oder Änderungen in seiner Umgebung zu erkennen und die Informationen an andere Elektronik, häufig einen Computerprozessor, zu senden.
Sideloading	Das Laden bzw. Installieren einer Applikation nicht über einen App-Store, sondern auf andere Art und Weise.
Sitzung	Ein Ereignis mit einem bestimmten Zeitrahmen (timeboxed).
Sitzungsnotizen	Ein Dokument zum Beobachten und Aufzeichnen einer Testsitzung
Smartphone	Ein handlicher Personal Computer mit einem mobilen Betriebssystem und einer integrierten mobilen Breitband-Mobilfunknetzverbindung für Sprache, SMS (Short Message Service) und Datenkommunikation über das Internet.
Software Development Kit	Eine Reihe von Werkzeugen und Bibliotheken zum Entwickeln von Software für eine bestimmte Plattform.
Strichcode	Eine optische, maschinenlesbare Darstellung von Daten.
Stromsparmodus	Ein Betriebsmodus für mobile Geräte, der vom Benutzer oder vom Gerät selbst aktiviert werden kann, um Energie zu sparen.
Stromverbrauch	Die Menge der verbrauchten Energie.
synchrone Kommunikation	Ein Datenübertragungsverfahren, bei dem Datenströme mit derselben Geschwindigkeit gesendet (Upstream) und empfangen (Downstream) werden und durch Zeitsignale synchronisiert sind.
Tablet	Ein mobiler Gerätetyp, bei dem es sich üblicherweise um ein Gerät mit einem Bildschirm von 7 Zoll und mehr handelt.
Teilstreckenverfahren	Ein Verfahren zur Datensynchronisierung, bei dem die Daten lokal gespeichert und bei verfügbarer Netzwerkverbindung an den Server weitergeleitet werden.
Thin-Client	Ein Client in Client-Server-Applikationen, der besonders klein ausgelegt ist, sodass der Großteil der Datenverarbeitung auf dem Server erfolgt.
transaktionsbasierte Apps	Eine Applikation, in der Benutzer pro Transaktion bezahlen.
Überlauf	Eine Situation, in welcher der Umfang der eingehenden Daten das verfügbare Volumen überschreitet.
Unterbrechung	Ein Ereignis, das während eines anderen Ereignisses auftritt
Unternehmens-App	Eine Applikation, die für die interne Verwendung in einer Organisation erstellt wurde und nicht für die öffentliche Verwendung bestimmt ist.
Virtual Private Network (VPN)	Ein verschlüsselter privater Kanal über ein öffentliches Netzwerk.
vorinstallierte App	Eine mobile Applikation, die vom Gerätehersteller installiert wird. Normalerweise kann der Benutzer diese Applikationen nicht deinstallieren.
Vor-Ort-Labor	Ein Labor, das sich physisch am selben Ort befindet wie der Benutzer des Labors.
Wearable	Ein am Körper getragenes Computergerät wie z. B. eine Uhr oder eine Brille.

Glossarbegriff	Definition
Web-App	Eine im Internet zur Verfügung gestellte Applikation, auf die via einem Browser zugegriffen wird.
Xcode	Eine integrierte Entwicklungsumgebung, die von Apple Inc. zur Entwicklung von Applikationen für Apple-Geräte bereitgestellt wird.

9. Index

2G 50
3D Touch 50
3G 50
4G 50
5G 50
Abbruch 29
Abwärtskompatibilität 50
ADB 50
ADM 50
Android Device Monitor 50
Android Studio 39, 50
anzeigengestützte App 12
anzeigengestützte Applikation 50
API 42
App-Daten beibehalten 50
App-Store 30, 50
App-Store-Zulassung 34
asynchrone Datenübertragung 16
asynchrone Kommunikation 50
Ausrichtung 24, 50
Automatisierungsansatz 42
AVD 50
Backendsystem 50
Barrierefreiheit 29
Basistelefon 13, 50
Begleitgerät 50
Benachrichtigungen 25, 50
Benutzereinstellungen 26
Berechtigungen 24
Bibliothek 50
Bildschirmausrichtung 24
Bildschirmtastatur 50
Bluetooth 50
browserübergreifende Kompatibilität 20
Bytecode 50
Code-Einschleusung 29
CPU-Frequenz 50
CRUD 33, 51
Datenintegrität 51
Datensynchronisierung 51
Datenvalidierung 51
Displays 22
dpi 51
Drittanbietermarktplatz 51
einschichtig 15, 51
Einstellungen 26, 51
Emulator 39, 40, 51
entferntes Testlabor 39, 41
exploratives Testen 29, 36
externer Speicher 51
Fat-Client 15, 51
Feature-Phone 13, 51
Fehlerjagd 17
Feldtest 27, 29, 33, 39
Fernzugriff auf Geräte 51
flache Datei 51
Flugmodus 51
Force Touch 51
Freemium App 12
Freemium-App 51
Funkloch 51
Funkzelle 51
Gebrauchstauglichkeit 20
Gebrauchstauglichkeitslabor 29
Gebrauchstauglichkeitstest 29, 32
gerätebasiertes Testen 42
Gerätefragmentierung 51
Gerätefunktionen 22
Geste 51
Globalisierung 33, 51
GPS 52
Gratis App 13
Größe des Anzeigebereichs 52
GSM 52
Heuristik 29, 35
Hochformat 52
hybride App 14, 26, 52
I18N 52
IDE 39, 52
immer verbundene Apps 16
In-App-Kauf 52
Installierbarkeit 29, 30
Instruments 52
integrierte Entwicklungsumgebung 52
Internationalisierung 33, 52
interner Speicher 52
Interoperabilität 20, 26
IoT-Gerät 13, 52
IT-Sicherheitstest 29
IT-Sicherheitstests 31
Jailbreaking 52
Koexistenz 20, 27
Kompatibilität 20
Konflikt beim Hochladen 52
kontinuierlicher Modus 16
kostenpflichtige App 13
kostenpflichtige App 52
L10N 52
Laufzeitumgebung 52
Lernziele 9
Lokalisierung 33, 52
Look-and-Feel/ 52
mehrschichtig 15, 53
Merkhilfe 34, 53
mobile Plattform 53
mobiler Gerätetyp 53

mobiles Betriebssystem 53
Mobilfunkdaten 53
Mobilfunknetz 53
Mobilgeräte-Emulator 53
Mobilgeräte-Simulator 53
Mobilgerätewelt 53
Multiplattformapplikation 53
Multitouch 53
native App 14, 26, 53
NFC 53
Nicht-Stören-Modus 53
nie verbundene Apps 16
OCR 53
OTA 53
Performanz 29
Performanztest 29, 32
Persona 34, 53
plattformübergreifendes Entwicklungsframework 53
Post-Release-Testen 29, 34
ppi 51
praktische Ziele 9
Prüfung 10
Punktdichte 51
QR-Code 53
Querformat 53
Richtlinien 33
Risikoanalyse 11, 19
Risikominderung 11, 19
risikoorientierter Test 11
Rooten 54
SBTM 36
Schnellzugriffsverknüpfungen 25
Schulungszeit 10
Seitenverhältnis 54
sensible Daten 54
Sensoren 23, 54
Sideloading 30, 54
Simulator 39, 40, 53
Sitzung 54
sitzungsbasiertes Testmanagement 29
Sitzungsnotizen 54
Skripterstellung 43
Smartphone 13, 54
Software Development Kit 54
Stresstest 29, 31
Strichcode 54
Stromsparmodus 54
Stromverbrauch 54
synchrone Datenübertragung 16
synchrone Kommunikation 54
System unter Test 20
Tablet 13, 54
Teilstreckenverfahren 16, 54
teilweise verbundene Apps 16
Temperatur 22
Testart 20
Testbericht 42
Testen der Barrierefreiheit 33
Testlabor 41
Testprozess 29, 37
Testpyramide 29, 38
Teststrategie 11, 16
Teststufe 29, 39
Thin-Client 15, 54
Tour 29, 35
transaktionsbasierte App 12, 54
Überlauf 54
umgebungsbasiertes Testen 39
Unterbrechungen 24, 54
Unternehmens-App 13, 54
User-Agent-basiertes Testen 42
Verbindungsfähigkeit 20, 27
Virtual Private Network (VPN) 54
vorinstallierte App 54
Vor-Ort-Labor 41
VPN 54
Wearable 13, 54
Web-App 26, 54
Web-Applikation 14
Xcode 39, 55