Certified Tester

Mobile Application Testing Syllabus

CT-MAT

versão 1.0

International Software Testing Qualifications Board

Provided by:

International Software Quality Institute (iSQI)
International Software Testing Qualifications Board

BSTQB

Versão traduzida, adaptada e disponibilizada na Língua Portuguesa pelo *Grupo de Traduções do BSTQB*

Direitos autorais

Este documento pode ser copiado em sua totalidade, ou extratos feitos, se a fonte for confirmada.

Copyright Notice © International Software Testing Qualifications Board (hereinafter called ISTQB®)

ISTQB® é marca registrada do International Software Testing Qualifications Board.

Autores do syllabus *Certified Mobile Application Professional – Foundation Level (CMAP-FL)*: Jose Diaz, Rahul Verma, Tarun Banga, Vipul Kocher and Yaron Tsubery - transferred the copyright to ISTQB®. Este syllabus foi utilizado como base para a criação desse documento.

Copyright © 2019 pelos autores: Vipul Kocher (chair), Piotr Wicherski (vice-chair), José Díaz, Matthias Hamburg, Eran Kinsbruner, Björn Lemke, Samuel Ouko, Ralf Pichler, Nils Röttger, Yaron Tsubery.

Este documento foi produzido pela equipe central do *International Software Testing Qualifications Board Mobile Application Testing Working Group*: Vipul Kocher (chair), Piotr Wicherski (vice-chair), José Díaz, Matthias Hamburg, Eran Kinsbruner, Björn Lemke, Samuel Ouko, Tal Pe'er, Ralf Pichler, Lloyd Roden, Nils Röttger, Angelina Samaroo, Yaron Tsubery.

Os autores transferem os direitos autorais para o *International Software Testing Qualifications Board (ISTQB®)*. Os autores (como detentores atuais de direitos autorais) e ISTQB® (como futuro detentor dos direitos autorais) concordaram com as seguintes condições de uso:

Qualquer indivíduo ou empresa de formação pode utilizar este programa como base para um curso de formação se os autores e o ISTQB forem reconhecidos como os responsáveis pelos direitos de autor e fonte e desde que qualquer anúncio desse curso de formação apenas possa mencionar o plano de estudos após a submissão para credenciamento oficial dos materiais de treinamento para um Conselho de Membros reconhecido pelo ISTQB®.

Qualquer indivíduo ou grupo de indivíduos pode usar este programa como base para artigos, livros ou outros escritos derivados se os autores e o ISTQB® forem reconhecidos como os proprietários das fontes e dos direitos autorais do plano de estudos.

Qualquer Membro do Conselho de Administração reconhecido pelo ISTQB® pode traduzir este plano de estudos e licenciar o plano de estudos (ou sua tradução) para outras partes.

©International Software Testing Qualifications Board

Histórico da Revisão

Versão	Data	Observações
Alpha	11 maio 2018	Alpha Release
Beta	27 janeiro 2019	Beta Release
GA	28 março 2019	GA Release
V2019	3 maio 2019	ISTQB® Release

Histórico da versão BSTQB

Versão	Data	Observações
1	14/06/2023	Padronização do layout com o ISTQB

©International Software Testing Qualifications Board

Índice

Direitos autorais 2
Histórico da Revisão 3
Índice 4
Agradecimentos 7
0 Introdução 8
0.1 Finalidade deste documento 8
0.2 A certificação em teste de aplicativos para dispositivos móveis 8
0.3 Resultados de negócios 8
0.4 Objetivos de aprendizagem examináveis 9
0.5 Níveis práticos de competência 9
0.6 O exame 10
0.7 Tempo de estudo recomendado 10
0.8 Pré-requisitos 10
0.9 Fontes de informação 10
1 Mundo Mobile, diretrizes de negócio e tecnologia [175 min] 11
1.1 Dados do Mobile Analytics 12
1.2 Modelos de negócio 12
1.3 Tipos de dispositivos móveis 13
1.4 Tipos de aplicativos 14
1.5 Arquitetura de aplicativos 15
1.6 Estratégia de teste 17
1.7 Desafios do teste 18
1.8 Riscos no teste 19
2 Tipos de teste [265 min] 21
2.1 Teste de compatibilidade do dispositivo de hardware 23
2.1.1 Teste de recursos do dispositivo 23
2.1.2 Teste de diferentes exibições 23
2.1.3 Teste de temperatura do dispositivo 24
2.1.4 Teste de sensores de entrada do dispositivo 24
2.1.5 Teste de vários métodos de entrada 25
2.1.6 Teste de mudança de orientação da tela 25
2.1.7 Teste de interrupções 26
2.1.8 Teste de permissões de acesso para recursos do dispositivo 26
2.1.9 Teste de consumo de energia e estado 27
2.2 Teste de interações de aplicativos com o software do dispositivo 27
2.2.1 Teste de notificações 27

©International Software Testing Qualifications Board

2.2.2 Teste em links de acesso rápido ... 27
2.2.3 Teste de preferências do usuário fornecidas pelo sistema operacional ... 28
2.2.4 Teste em diferentes tipos de aplicativos ... 28
2.2.5 Teste de interoperabilidade com várias plataformas e versões do sistema operacional ... 28
2.2.6 Teste de interoperabilidade e coexistência com outros aplicativos no dispositivo ... 29
2.3 Teste de vários métodos de conectividade ... 29
3 Tipos e processos de teste [200 min] ... 31
3.1 Tipos comuns aplicáveis de teste ... 33
3.1.1 Testes de instalação ... 33
3.1.2 Teste de estresse ... 34
3.1.3 Testes de segurança ... 34
3.1.4 Teste de performance ... 35
3.1.5 Teste de usabilidade ... 35
3.1.6 Teste de banco de dados ... 36
3.1.7 Teste de internacionalização e regionalização ... 36
3.1.8 Teste de acessibilidade ... 36
3.2 Níveis de teste adicionais aplicáveis ... 37
3.2.1 Teste de campo ... 37
3.2.2 Teste de aprovação de armazenamento de aplicativos e teste pós-lançamento ... 37
3.3 Técnicas de teste baseadas na experiência ... 38
3.3.1 Personas e mnemônicos ... 38
3.3.2 Heurística ... 39
3.3.3 Roteiros ... 39
3.3.4 Gerenciamento de teste baseado em sessão (SBTM) ... 40
3.4 Processo e abordagem do teste ... 40
3.4.1 Processo de teste ... 40
3.4.2 Abordagens de teste ... 41
4 Plataformas de aplicativos, ferramentas e ambiente [80 min] ... 43
4.1 Plataformas de desenvolvimento de aplicativos ... 44
4.2 Ferramentas comuns da plataforma de desenvolvimento ... 44
4.3 Emuladores e simuladores ... 44
4.3.1 Visão geral de emuladores e simuladores ... 44
4.3.2 Usando emuladores e simuladores ... 45
4.4 Configurando um laboratório de teste ... 45
5 Automação da execução dos testes [55 min] ... 47
5.1 Abordagens de automação ... 48
5.2 Métodos de automação ... 48

©International Software Testing Qualifications Board

5.3 Avaliação das ferramentas de automação 49
5.4 Abordagens para configuração de um laboratório de automação 50
Referências 51
[Apêndice A] Objetivos de aprendizagem e níveis cognitivos 53
[Apêndice B] Glossário de termos específicos 55

©International Software Testing Qualifications Board

Agradecimentos

Esse documento foi produzido pelo *International Software Testing Qualifications Board Mobile Application Testing Working Group*.

Vipul Kocher (chair), Piotr Wicherski (vice-chair), José Díaz, Matthias Hamburg, Eran Kinsbruner, Björn Lemke, Samuel Ouko, Tal Pe'er, Ralf Pichler, Lloyd Roden, Nils Röttger, Angelina Samaroo, Yaron Tsubery.

A equipe principal agradece à equipe de revisão por suas sugestões e sugestões.

As seguintes pessoas participaram na revisão, comentários ou votação deste syllabus:

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

Esse documento foi formalmente oficializado em 3 de maio de 2019 pelo ISTQB®.

O BSTQB deseja agradecer a sua equipe do grupo de trabalho de traduções (WG Traduções) pelo empenho e esforço na tradução deste material: Irene Nagase, José Luiz Pereira, Paula B. de O. Fialkovitz, Paula Renata Z. de Souza, Rogério Athayde de Almeida, George Fialkovitz.

©International Software Testing Qualifications Board

0 Introdução

0.1 Finalidade deste documento

Este syllabus forma a base para o *ISTQB® CT-MAT Certified Tester (Foundation Level) – Mobile Application Testing*. O ISTQB® fornece este syllabus da seguinte forma:

1. Para os Conselhos Nacionais, para traduzirem ao seu idioma local e credenciar provedores de treinamento. Os Conselhos Nacionais podem adaptar o plano de estudos às suas necessidades linguísticas específicas e modificar as referências para se adaptarem às suas publicações locais.
2. Para os Conselhos de Exames, para derivar questões de exame em sua língua local adaptadas aos objetivos de aprendizagem de cada syllabus.
3. Aos Provedores de Treinamento, para produzir material didático e determinar os métodos de ensino apropriados.
4. Para os candidatos à certificação, para se prepararem para o exame (como parte de um curso de treinamento ou de forma independente).
5. Para a comunidade internacional de software e engenharia de sistemas, para promover a profissão de software e testes de sistemas, e como base para livros e artigos.

O ISTQB® pode permitir que outras entidades utilizem este programa para outros fins, desde que procurem e obtenham permissão prévia por escrito.

0.2 A certificação em teste de aplicativos para dispositivos móveis

A certificação *ISTQB® CT-MAT Certified Tester (Foundation Level) – Mobile Application Testing* é destinada a qualquer pessoa envolvida em teste de software que deseje ampliar seus conhecimentos em testes de aplicativos para dispositivos móveis ou qualquer pessoa que deseje iniciar uma carreira de especialista em testes de aplicativos para dispositivos móveis.

As informações sobre o *Mobile Application Testing* descritas no syllabus *ISTQB® CTFL Certified Tester Foundation Level* [ISTQB_FL_2018] foram consideradas na criação desse documento.

0.3 Resultados de negócios

Esta seção lista os resultados de negócios esperados de um candidato que tenha atingido a certificação *ISTQB® CT-MAT Certified Tester (Foundation Level) – Mobile Application Testing*.

MAT-01 Compreender e revisar os guias de negócios e tecnologia para aplicativos para dispositivos móveis para criar uma estratégia de teste.

MAT-02 Identificar e entender os principais desafios, riscos e expectativas associados ao teste de um aplicativo para dispositivo móvel.

MAT-03 Aplicar os tipos e níveis específicos de testes para aplicativos de dispositivos móveis.

MAT-04 Aplicar os tipos mais comuns de teste, como os mencionados no [ISTQB_FL_2018], no contexto específico para dispositivos móveis.

©International Software Testing Qualifications Board

MAT-05 Realizar as atividades necessárias, especificamente nos testes de aplicativos para dispositivos móveis como parte das principais atividades descritas no processo de teste do ISTQB®.

MAT-06 Identificar e utilizar ambientes adequados e ferramentas apropriadas para os testes de aplicativos para dispositivos móveis.

MAT-07 Compreender os métodos e ferramentas usados especificamente para suportar a automação de teste de aplicativo para dispositivo móvel.

0.4 Objetivos de aprendizagem examináveis

Os objetivos de aprendizagem dão suporte aos resultados de negócios e são usados na criação do exame que o candidato deve ser aprovado para alcançar a certificação CT-MAT. Os objetivos de aprendizagem são alocados em níveis cognitivos de conhecimento (nível K).

Um nível K, ou nível cognitivo, é usado para classificar os objetivos de aprendizagem de acordo com a taxonomia revisada de Bloom [Anderson 2001]. O ISTQB® usa essa taxonomia para projetar exames de seus syllabi.

Este syllabus considera três níveis-K diferentes (K1 a K3). Para mais informações, veja o capítulo 7

0.5 Níveis práticos de competência

O CT-MAT introduz o conceito de Objetivos Práticos, que se concentram em habilidades e competências práticas.

As competências podem ser alcançadas através da realização de exercícios práticos, como os mostrados na seguinte lista:

- Exercícios para os objetivos de aprendizado no nível K3 realizados usando papel e caneta ou software de processamento de texto, como é feito para os vários syllabus do ISTQB®.
- Configurando e usando ambientes de teste.
- Testando aplicativos em dispositivos virtuais e físicos.
- Usando ferramentas em desktops ou dispositivos móveis para testar ou auxiliar no teste de tarefas relacionadas, como instalação, consulta, registro, monitoramento, capturas de tela etc.

Os seguintes níveis aplicam-se a objetivos práticos:

- **H0**: Pode incluir uma demonstração ao vivo de um exercício ou vídeo gravado. Como isso não é realizado pelo aluno, não é estritamente um exercício.
- **H1**: Exercício guiado. O aluno segue uma sequência de etapas realizadas pelo instrutor.
- **H2**: Exercícios sugeridos. O aluno recebe um exercício com sugestões relevantes para permitir que seja resolvido dentro de um prazo determinado.
- **H3**: Exercícios não guiados sem dicas.

Recomendações:

- Os objetivos de aprendizado do K1 normalmente usam o nível H0 e H1 ou H2 quando a situação exige.

©International Software Testing Qualifications Board

- Os objetivos de aprendizado do K2 geralmente usam níveis H1 ou H2 e H0 ou H3 quando a situação exige.
- Os objetivos de aprendizagem do K3 normalmente usam os níveis de H2 ou H3, embora nem sempre seja necessário ter um exercício prático para um objetivo de aprendizado do K3. Se a configuração for complexa ou se consumir muito tempo, use o nível H0.

0.6 O exame

O exame do CT-MAT será baseado neste syllabus.

As respostas às questões do exame podem requerer conhecimento com base em mais de uma seção desse syllabus. Todas as seções são examináveis, com exceção da introdução e dos apêndices. Normas, livros e outros syllabus do ISTQB® são incluídos como referências, mas seu conteúdo não é passível de análise, além do que está resumido neste próprio syllabus.

O exame CT-MAT é de múltipla escolha. Existem 40 perguntas. Para passar no exame, pelo menos 65% das perguntas (ou seja, 26 perguntas) devem ser respondidas corretamente.

Os exames podem ser feitos como parte de um curso de treinamento credenciado ou realizado de forma independente (p. ex.: em um centro de exames ou em um exame público). A conclusão de um treinamento credenciado não é um pré-requisito para o exame.

0.7 Tempo de estudo recomendado

O tempo mínimo de treinamento foi definido para cada objetivo de aprendizado nesse syllabus. O tempo total para cada capítulo é indicado no título do capítulo.

Os provedores de treinamento devem observar que outros syllabi do ISTQB® aplicam uma abordagem de "tempo padrão" que aloca horários fixos de acordo com o K-Level. O programa do *Mobile Application Testing* não aplica estritamente este esquema. Como resultado, os provedores de treinamento recebem uma indicação mais flexível e realista dos tempos mínimos de treinamento para cada objetivo de aprendizado.

0.8 Pré-requisitos

A *Certificação ISTQB® CTFL Foundation Level* deve ser obtida antes de fazer este exame.

0.9 Fontes de informação

Os termos usados no plano de estudos são definidos no Glossário de termos do ISTQB® [ISTQB_GLOSSARY].

O Capítulo 6 contém uma lista de livros e artigos recomendados sobre o tema desse syllabus.

©International Software Testing Qualifications Board

1 Mundo Mobile, diretrizes de negócio e tecnologia [175 min]

Palavras-chave

análise de risco, mitigação de risco, teste baseado em risco, estratégia de teste

Objetivos de aprendizagem

1.1 Dados do Mobile Analytics

MAT-1.1.1: (K2) Descrever como os dados de análise móvel disponíveis podem ser usados como entrada para a estratégia de teste e o plano de teste.

HO-1.1.1: (H3) Com base nos dados coletados de uma ou mais fontes de dados analíticos (localização geográfica, plataforma, versão do sistema operacional e distribuição de tipo de dispositivo), selecionar os tipos de dispositivos a serem testados e sua priorização correspondente. Nota: HO-1.1.1 e HO-1.7.1 (abaixo) podem ser combinados.

1.2 Modelos de negócios

MAT-1.2.1: (K2) Distinguir entre vários modelos de negócios de aplicativos para dispositivos móveis.

1.3 Tipos de dispositivos móveis

MAT-1.3.1: (K1) Lembrar de diferentes tipos de dispositivos móveis.

1.4 Tipos de aplicativos

MAT-1.4.1: (K2) Distinguir entre diferentes tipos de aplicativos para dispositivos móveis.

1.5 Arquitetura de aplicativos

MAT-1.5.1: (K2) Distinguir entre tipos gerais de arquitetura de aplicativos para dispositivos móveis.

1.6 Estratégia de teste

MAT-1.6.1: (K3) Aplicar as características e especificidades do mercado móvel na preparação de uma estratégia de teste.

1.7 Desafios do teste

MAT-1.7.1: (K2) Dar exemplos dos desafios associados ao teste de aplicativos para dispositivos móveis.

HO-1.7.1: (H1) Reunir os dados de mercado, como compartilhamento de mercado de dispositivos ou sistemas operacionais, para uma região selecionada. Reunir dados para tamanhos de tela e densidade. Criar uma lista de cinco dispositivos e calcular a cobertura de mercado esperada para essa lista. Nota: HO-1.1.1 (veja anteriormente) e HO-1.7.1 podem ser combinados.

1.8 Riscos no teste

MAT-1.8.1: (K2) Descrever como os riscos específicos de aplicativos para dispositivos móveis podem ser mitigados.

©International Software Testing Qualifications Board

1.1 Dados do Mobile Analytics

Há muitos stakeholders no mundo móvel, incluindo fabricantes, provedores de plataforma, provedores de sistemas operacionais (OS), fornecedores de dados de mercado, provedores de ferramentas e, é claro, desenvolvedores de aplicativos e testadores.

Para contribuir efetivamente nas discussões de planejamento e análises de teste, um testador de aplicativo para dispositivo móvel deve estar ciente e familiarizado com os seguintes fatores:

- As implicações comerciais das plataformas de distribuição.
- Downloads de aplicativos por plataforma.
- A quantidade e distribuição de versões de sistemas operacionais.
- A distribuição no mercado de vários tipos de dispositivos, incluindo variações com base na localização geográfica.
- Tamanhos e resoluções diferentes de tela.
- Diversos métodos de entrada.
- Tipos de câmera.

Existem várias fontes de informação para esses fatores, tanto gratuitos quanto comerciais. Estes incluem o *StatCounter GlobalStats* [URL1], os próprios fornecedores do sistema operacional, e outras fontes de terceiros.

Os dados de *Mobile Analytics* são usados para selecionar um portfólio de dispositivos para execução de teste, que seja apropriado para o mercado-alvo. Os testes são executados sobre esse portfólio para testar o aplicativo em um dispositivo de acordo com a importância deste dispositivo. Os dados relacionados aos dispositivos e suas características especiais, se houver, também podem ser usados para projetar testes específicos para um tipo de dispositivo. Por exemplo, um dispositivo com sensor de batimento cardíaco pode precisar de casos de teste especiais.

1.2 Modelos de negócio

Existem vários modelos que podem ser usados para monetizar o trabalho realizado na criação de aplicativos para dispositivos móveis. Isso inclui, mas não se limita a: aplicativos *freemium*, baseados em propagandas, baseados em transações, com base em taxas, e corporativos. Além disso, as compras no aplicativo podem ser aplicadas a alguns desses modelos.

Há certas vantagens e desvantagens para cada uma dessas abordagens e o testador deve manter o modelo de negócios em mente enquanto testa o aplicativo para dispositivo móvel.

Em um modelo ***freemium***, os aplicativos geralmente são gratuitos, mas os usuários precisam pagar se precisarem de recursos adicionais. Os aplicativos precisam fornecer recursos suficientes para serem atraentes para os usuários, ao mesmo tempo em que fornecem recursos avançados para os quais muitos usuários estariam dispostos a pagar.

Os aplicativos **baseados em propagandas** exibem anúncios na tela à medida que os usuários interagem com os aplicativos. Essa estratégia para geração de receita é mais eficaz se os aplicativos forem usados por períodos relativamente longos. Os designers de interface do usuário devem ter

©International Software Testing Qualifications Board

cuidado ao exibir os anúncios. Eles devem ser proeminentes o suficiente sem ocultar partes essenciais do aplicativo e devem garantir que os usuários não se distraiam e não gostem de usar o aplicativo.

Os aplicativos **baseados em transações** cobram dos usuários por transação, uma taxa fixa ou uma porcentagem do valor da transação ou similar. Esse tipo de modelo de negócios é adequado apenas para um número limitado de aplicativos e geralmente é aplicado a aplicativos comerciais e financeiros, como carteiras móveis.

Aplicativos **baseados em taxas** exigem que os usuários paguem para baixar e instalar o aplicativo. A decisão sobre um modelo de negócios baseado em taxas deve ser bem considerada, já que existem muitas opções gratuitas ou livres para a maioria dos tipos de aplicativos. A probabilidade dos usuários comprarem um aplicativo desse tipo aumenta se lhe são oferecidos recursos ou usabilidades excepcionais ou quando aplicativos concorrentes não estão disponíveis.

Aplicativos **gratuitos** e **corporativos** não cobram seus usuários. Aplicativos corporativos são desenvolvidos para uso interno dentro da organização e fornecem uma interface para os serviços fornecidos. Existem muitos desses aplicativos disponíveis em organizações como bancos ou empresas de comércio eletrônico. Esses aplicativos geralmente não estão focados em monetizar o aplicativo em si, mas permitem que a receita seja gerada direcionando os usuários para os serviços fornecidos pelas organizações.

1.3 Tipos de dispositivos móveis

Existe uma variedade de dispositivos móveis disponíveis que suportam diferentes tipos de aplicativos.

Dispositivos típicos incluem:

- Telefones básicos
- Telefones típicos
- Smartphones
- Tablets
- Dispositivos companheiros - incluindo *wearables* e alguns dispositivos IoT (*Internet of Things*).

Ao testar, deve-se ter em mente que cada tipo de dispositivo possui recursos específicos para necessidades específicas.

Os **telefones básicos** são usados apenas para telefone e SMS e fornecem pouquíssimos aplicativos e jogos internos. A instalação de aplicativos ou navegação não é possível.

Os **telefones típicos** fornecem suporte limitado para aplicativos e instalação de aplicativos. Eles fornecem acesso à Internet através de um navegador embutido e podem ter algum hardware adicional, como câmeras.

Os **smartphones** fornecem telefones com vários sensores. O sistema operacional suporta recursos como instalação de aplicativos, suporte a multimídia e navegação.

Os **tablets** são semelhantes aos smartphones, mas são fisicamente maiores. Eles são normalmente usados quando um display maior é necessário ou desejado, podendo também suportar uma vida útil mais longa da bateria.

©International Software Testing Qualifications Board

Os **dispositivos companheiros** e alguns dispositivos de IoT são dispositivos de computador comumente usados em conjunto com um *smartphone* ou *tablet* para estender uma funcionalidade disponível ou para dar acesso aos dados no telefone ou tablet de maneira mais conveniente.

Os **wearables** são dispositivos que podem ser usados pelos consumidores. Estes podem agir como um complemento aos dispositivos existentes ou funcionar de forma independente. Relógios e pulseiras de fitness são exemplos de *wearables* populares.

1.4 Tipos de aplicativos

Existem três tipos principais de aplicativos para dispositivos móveis:

- Nativo
- Baseado em navegador
- Híbrido

Cada tipo de aplicativo vantagens e desvantagens, exigindo que uma decisão de negócio seja tomada antes de iniciar o desenvolvimento do aplicativo.

As aplicações **nativas** são desenvolvidas usando kits de desenvolvimento de software (SDKs) específicos da plataforma, ferramentas de desenvolvimento, e sensores ou recursos específicos da plataforma. Eles são baixados, instalados e atualizados nas lojas de fornecedores. Esses aplicativos podem necessitar de testes em todos os dispositivos compatíveis.

Os aplicativos nativos geralmente oferecem melhor desempenho, podem utilizar totalmente os recursos da plataforma e atender às expectativas da plataforma para a qual são desenvolvidos. O custo de desenvolvimento é tipicamente maior e desafios adicionais podem ser aplicados, como o uso de múltiplas plataformas e a instalação e teste em muitos dispositivos.

Aplicativos **baseados em navegador** são acessados por meio de um navegador móvel. Como eles usam tecnologias e navegadores típicos de desenvolvimento da Web, o suporte a várias plataformas é fácil, e o custo de desenvolvimento geralmente é menor.

Existem quatro maneiras principais de criar aplicativos da Web para dispositivos móveis:

- Versões específicas para dispositivos móveis de sites e aplicativos (reconhecidos pelo domínio m.<url-do-site>). Geralmente, isso significa que, quando um navegador móvel aborda o aplicativo, lhe é fornecida uma versão móvel para navegação. Por exemplo, facebook.com redireciona para *m.facebook.com* quando acessado de um dispositivo móvel.
- Os aplicativos da web responsivos garantem que o desenho da tela se ajuste ao fator de forma e ao tamanho da tela, normalmente expressos como portas de exibição.
- Aplicativos da Web adaptativos ajustam o desenho da tela de acordo com alguns tamanhos predefinidos. Existem diferentes modelos para esses tamanhos e os recursos disponíveis para o usuário geralmente são ajustáveis.
- Aplicativos da Web progressivos permitem que atalhos de páginas da Web específicas sejam criados na tela inicial do dispositivo móvel. Eles aparecem como aplicativos nativos e às vezes até podem trabalhar off-line.

©International Software Testing Qualifications Board

Os aplicativos da Web para dispositivos móveis são criados usando tecnologias comuns da Web, o que geralmente facilita o desenvolvimento e o gerenciamento em comparação com aplicativos nativos e híbridos. No entanto, eles podem não ser tão ricos em recursos quanto os aplicativos nativos ou híbridos e podem ter acesso limitado às APIs (Interfaces de programação de aplicativo) nativas da plataforma. O acesso a sensores móveis também é limitado. O teste de instalação em dispositivos não é necessário, mas o teste de compatibilidade do navegador o é.

Os aplicativos **híbridos** são uma combinação de aplicativo nativo e aplicativo da web. Eles usam um empacotador de aplicativo que contém um visualizador nativo que executa o aplicativo da web internamente. Esses aplicativos são baixados de lojas de fornecedores e podem acessar todos os recursos do dispositivo. Eles são relativamente fáceis de desenvolver, atualizar e manter sem atualizar o aplicativo instalado no dispositivo. As habilidades necessárias para desenvolver esses aplicativos são quase as mesmas que para o desenvolvimento da web. Os possíveis pontos fracos para desses aplicativos incluem problemas de desempenho devido ao uso de um empacotador e possíveis divergências da aparência esperada devido à aspectos específicos da plataforma.

Os aplicativos nativos e híbridos são instalados fisicamente em um dispositivo e, portanto, estão sempre disponíveis para o usuário, mesmo quando o dispositivo não possui conexão com a Internet. Em comparação, os aplicativos baseados em navegador sempre exigem acesso à Internet.

Alguns aplicativos são pré-instalados no dispositivo móvel e outros podem ser instalados através de vários canais de distribuição, como *Apple App Store*, *Google Play Store*, lojas de aplicativos corporativos (disponíveis apenas dentro da rede corporativa) e mercados de aplicativos de terceiros.

O teste de cada um desses tipos de aplicativos pode exigir uma abordagem diferente. Os parâmetros a serem considerados incluem:

- Diferentes tipos de dispositivos a serem suportados.
- Recursos dos sensores e dispositivos a serem usados.
- Disponibilidade sob várias condições de rede.
- Capacidade de instalação, compatibilidade, eficiência de desempenho e usabilidade.

1.5 Arquitetura de aplicativos

Existem várias soluções para projetar um aplicativo para dispositivo móvel.

Algumas das considerações na escolha de uma arquitetura específica ou decisão de desenho incluem:

- Público-alvo.
- Tipo de aplicação.
- Suporte a várias plataformas móveis e não móveis.
- Necessidades de conectividade.
- Necessidades de armazenamento de dados.
- Conexões com outros dispositivos, incluindo dispositivos IoT.

Decisões de arquitetura incluem:

- Arquitetura *client-side*, tal como cliente pequeno ou grande.

©International Software Testing Qualifications Board

- Arquitetura *server-side*, tal como um ou vários níveis.
- Tipo de conexão, como *Wi-Fi*, dados de celular, NFC (*Near Field Communication*), *Bluetooth*
- Métodos de sincronização de dados, como armazenar e encaminhar, push e pull, comunicações síncronas e assíncronas

Os aplicativos *client-side* pequenos não contêm código personalizado para o dispositivo e fazem uso mínimo dos recursos do sistema operacional móvel. Esses aplicativos geralmente usam o navegador da web como *front-end* e *JavaScript* como o idioma para implementar a lógica do lado do cliente.

Os aplicativos *client-side* grandes (ou gordos) podem ter várias camadas de código de aplicativo e podem usar recursos do sistema operacional móvel. Estes são tipicamente aplicativos nativos ou híbridos.

As arquiteturas *server-side* incluem as seguintes possibilidades:

- As arquiteturas de camada única são monolíticas e possuem todos os servidores na mesma máquina. Eles são menos escaláveis e mais difíceis de proteger.
- As arquiteturas de várias camadas distribuem componentes do lado do servidor em várias unidades. As arquiteturas de duas camadas envolvem servidores da Web e de banco de dados separados, enquanto as arquiteturas de três camadas também incluem um servidor de aplicativos. As arquiteturas multicamadas permitem a separação de responsabilidades, fornecem especialização em banco de dados e proporcionam melhor flexibilidade, escalabilidade e segurança. No entanto, eles podem ser significativamente mais caros para desenvolver, gerenciar e hospedar, em comparação com arquiteturas de camada única.

Existem vários métodos de conexão. Um dispositivo móvel pode ser conectado ao servidor por meio diversos tipos de conexão, como *Wi-Fi* ou conexões de dados celulares como 2G, 3G, 4G e 5G. Os aplicativos para dispositivos móveis geralmente operam em um dos três modos a seguir:

- Aplicativos nunca conectados funcionam off-line e não precisam estar conectados. Uma calculadora simples é um exemplo de tal aplicativo.
- Aplicativos sempre conectados exigem uma conexão de rede permanente durante a operação. Todos os aplicativos da web para dispositivos móveis se enquadram nessa categoria, embora alguns possam operar de maneira limitada quando parcialmente conectados.
- Aplicativos parcialmente conectados exigem uma conexão para tarefas como transferência de dados, mas podem operar por longos períodos sem conexão.

A sincronização de dados entre o cliente e o servidor pode ser realizada nos seguintes modos:

- O modo contínuo é onde os dados são transferidos assim que são enviados.
- O modo de armazenamento e encaminhamento é onde os dados podem ser armazenados localmente antes de serem transferidos, especialmente quando não há conectividade disponível.

A transferência de dados pode ser realizada nas duas abordagens a seguir:

- A **transferência síncrona** de dados é executada quando a função de chamada aguarda que a função chamada seja concluída antes de retornar.

©International Software Testing Qualifications Board

- A **transferência assíncrona** de dados é executada quando a função de chamada do servidor retorna imediatamente, processa os dados em segundo plano e chama de volta a função do cliente que conclui a tarefa. Isso dá aos usuários mais controle. No entanto, a implementação do mecanismo de *handshake* aumenta a complexidade em relação à disponibilidade do cliente ou da rede quando o servidor inicia o retorno de chamada.

1.6 Estratégia de teste

Criar uma estratégia de teste para dispositivos móveis requer que o testador leve em consideração todos os parâmetros listados até agora neste capítulo. Além disso, os riscos discutidos nesta seção e os desafios descritos na seção 1.7 também devem ser considerados.

Riscos típicos são, por exemplo:

- Sem conhecer os dados de proliferação de dispositivos em uma determinada localização geográfica, não é possível escolher os dispositivos nos quais o aplicativo precisa ser testado de maneira sustentável.
- Sem conhecer o tipo de modelo de negócio, não é possível testar se o comportamento do aplicativo é adequado para esse modelo de negócio.

A criação de uma estratégia de teste para testes de aplicativos para dispositivos móveis também precisa considerar os seguintes riscos e desafios específicos:

- A variedade de dispositivos móveis com defeitos internos específicos.
- A disponibilidade de dispositivos internamente ou através do uso de laboratórios de teste externos.
- A introdução de novas tecnologias, dispositivos, ou plataformas durante o ciclo de vida da aplicação.
- A instalação e atualização do próprio aplicativo através de vários canais, incluindo a preservação de dados e preferências do aplicativo.
- Problemas de plataforma que podem impactar o aplicativo.
- Cobertura de rede e seu impacto no aplicativo em um contexto global.
- A capacidade de testar usando as redes de vários provedores de serviços.
- O uso de emuladores móveis, simuladores, ou dispositivos reais para níveis e tipos de teste específicos.

Esses desafios são descritos em maior detalhe na seção 1.7.

A estratégia de teste leva a riscos e desafios em conta. Por exemplo:

- A estratégia de teste pode especificar o uso de emuladores ou simuladores móveis nos estágios iniciais de desenvolvimento, seguidos por dispositivos reais em estágios posteriores. Existem certos tipos de testes que podem ser executados nos emuladores ou simuladores móveis, mas não em todos os tipos de testes. Mais sobre isso é descrito na seção 4.3.
- A estratégia de teste pode considerar o desafio colocado por muitos dispositivos diferentes, adotando uma das seguintes abordagens:

©International Software Testing Qualifications Board

- *Abordagem de plataforma única*: reduzir o escopo para um único tipo de dispositivo, uma versão do sistema operacional, uma operadora e um tipo de rede.
- *Abordagem multiplataforma*: reduzir o escopo a uma seleção representativa de dispositivos e sistemas operacionais usados pela maioria dos clientes no mercado-alvo, com base no tráfego móvel ou em outros dados analíticos.
- *Abordagem de cobertura máxima*: abranger todas as versões do sistema operacional, dispositivos, fabricantes, operadoras e tipos de rede. Isso é basicamente um teste exaustivo, que geralmente não é economicamente viável, especialmente quando se considera a grande quantidade de dispositivos e versões de sistemas operacionais no mercado.

- A estratégia de teste pode considerar o desafio colocado pela não disponibilidade de dispositivos, redes ou condições da vida real usando recursos externos, tais como:
  - *Serviços de acesso remoto a dispositivos*. Esta é uma maneira de acessar dispositivos pela Web que não são de outra maneira.
  - *Serviços de teste de multidão*. Esta é uma maneira de acessar um enorme grupo de voluntários e seus dispositivos.
  - *Redes pessoais como amigos e colegas*. Isso faz uso da própria rede social privada.
  - *Caça aos bugs*. Este é um evento de teste gamificado usando voluntários da empresa ou do público em geral.

Além dos níveis de teste descritos em [ISTQB_FL_2018], a estratégia de teste também considera os tipos comuns de testes usados para aplicações móveis (consulte a seção 3.1) e qualquer nível adicional de teste necessário (consulte a seção 3.2).

1.7 Desafios do teste

No mundo móvel, existem muitos desafios adicionais que são incomuns ou pouco críticos no software de desktop ou servidor. Os testadores devem estar cientes desses desafios e de como eles podem afetar o sucesso do aplicativo.

Desafios típicos no mundo móvel incluem:

- Várias plataformas e fragmentação de dispositivos: vários tipos e versões de sistemas operacionais, tamanhos de tela e qualidade de exibição.
- Diferenças de hardware em vários dispositivos: vários tipos de sensores e dificuldade em simular condições de teste para recursos restritos de CPU e RAM.
- Variedade de ferramentas de desenvolvimento de software exigidas pelas plataformas.
- Diferença entre os desenhos de interface do usuário e as expectativas da experiência do usuário (UX) das plataformas.
- Vários tipos de rede e provedores.
- Dispositivos sem recursos.
- Vários canais de distribuição para aplicativos.
- Diversos usuários e grupos de usuários.
- Vários tipos de aplicativos com vários métodos de conexão.
- Alta visibilidade do *feedback* resultante de bugs que têm um alto impacto nos usuários, o que pode facilmente resultar na publicação de *feedback* em mercados on-line.

©International Software Testing Qualifications Board

- Publicação de mercado, que exige ciclos de aprovação adicionais para publicação por proprietários de mercado, como o *Google Play* ou a *Apple App Store*.
- Indisponibilidade de dispositivos recém-lançados que exigem o uso de emuladores ou simuladores móveis.

O impacto desses desafios inclui:

- Grande número de combinações a serem testadas.
- Grande número de dispositivos necessários para testes, o que eleva o custo.
- A necessidade de compatibilidade com versões anteriores para executar o aplicativo em versões mais antigas da plataforma.
- Novos recursos sendo lançados em todas as versões do sistema operacional subjacente.
- Diretrizes a serem consideradas para as diversas plataformas.
- CPUs com falta de recursos, bem como quantidade limitada de memória e espaço de armazenamento.
- Largura variável de banda e instabilidade de várias redes.
- Alterações nas velocidades de *upload* e *download* disponíveis com base nos planos de dados.

Os dois exemplos a seguir ilustram desafios típicos e seu impacto potencial:

- Diferentes dispositivos têm diferentes tipos de sensores e os testes precisam levar em conta essa diferença. Todos os novos sensores adicionados ao hardware podem exigir testes adicionais de compatibilidade com versões anteriores.
- Alguns dos desafios de rede podem ser tratados apropriadamente, mesmo sob condições variáveis, usando-se estratégias apropriadas de armazenamento em cache e pré-busca. No entanto, isso tem um custo: muitas conexões abertas podem afetar o desempenho do lado do servidor, pois a maioria dos aplicativos mantém o usuário logado no servidor.

1.8 Riscos no teste

Os desafios mencionados na seção 1.7 podem aparecer isoladamente ou em combinação com outros. Isso pode resultar em riscos adicionais para um aplicativo para dispositivo móvel.

Um testador deve ser capaz de contribuir para a análise de risco do produto. Análise de risco comum e métodos de mitigação, como discutido em [ISTQB_FL_2018], capítulo 5.5, também podem ser aplicados no contexto móvel. Além disso, existem os seguintes riscos e estratégias de mitigação específicos para dispositivos móveis:

Risco	Mitigação possível
Fragmentação do mercado.	Escolha uma seleção apropriada de dispositivos para a execução de testes, por exemplo, testando os dispositivos mais usados.
Custo de suporte a múltiplas plataformas.	Realizar análises para entender as plataformas mais utilizadas, evitando assim testes de quem não está mais no escopo.

©International Software Testing Qualifications Board

Introdução de novas tecnologias, plataformas e dispositivos.	Use versões de pré-produção dessas tecnologias.
Falta de disponibilidade de dispositivos para execução de teste.	Aplicar serviços de acesso a dispositivos remotos ou serviços de teste de multidão.
Riscos dos padrões de uso esperados de aplicativos para dispositivos móveis usados em trânsito.	Aplicando abordagens de teste apropriadas, como teste de campo.

©International Software Testing Qualifications Board

2 Tipos de teste [265 min]

Palavras-chave

coexistência, compatibilidade entre navegadores, compatibilidade, conectividade, interoperabilidade, sistema em teste (SUT), tipo de teste, usabilidade.

Objetivos de aprendizagem

2.1 Teste de compatibilidade do dispositivo de hardware

MAT-2.1.1: (K2) Descrever os recursos e hardware específicos do dispositivo, que devem ser considerados para testes.

HO-2.1.1: (H1) Testar um aplicativo para várias funcionalidades do dispositivo móvel enquanto o sistema em teste (SUT) está em uso para verificar o funcionamento correto do SUT.

MAT-2.1.2: (K3) Preparar testes para a compatibilidade do aplicativo com tamanhos, proporção e densidade de tela.

HO-2.1.2: (H3) Testar um aplicativo em vários dispositivos móveis (virtuais ou físicos) para mostrar o impacto da resolução e do tamanho da tela na interface do usuário do aplicativo.

MAT-2.1.3: (K2) Descrever como os testes podem mostrar os efeitos potenciais do superaquecimento do dispositivo no sistema em teste.

MAT-2.1.4: (K1) Relembrar os diferentes tipos de teste para testar os vários sensores de entrada usados em dispositivos móveis.

MAT-2.1.5: (K1) Relembrar os testes a serem executados para vários métodos de entrada.

HO-2.1.5: (H0) Testar um aplicativo para vários tipos de entradas, incluindo testes relacionados ao teclado, com vários modelos de teclados, testes relacionados a gestos e (opcionalmente) testes relacionados à câmera.

MAT-2.1.6: (K2) Descrever como os testes podem revelar problemas na interface do usuário ao mudar a orientação da tela.

HO-2.1.6: (H3) Testar um aplicativo para verificar o efeito da mudança de orientação na funcionalidade do aplicativo, incluindo retenção de dados e correção da interface do usuário.

MAT-2.1.7: (K3) Preparar testes para um aplicativo usando interrupções típicas de dispositivos móveis.

HO-2.1.7: (H3) Testar um aplicativo para várias interrupções de dispositivos móveis enquanto o aplicativo estiver em uso.

MAT-2.1.8: (K3) Preparar testes para alterar as permissões de acesso aos recursos do dispositivo solicitados pelo aplicativo.

HO-2.1.8: (H3) Testar o gerenciamento de permissões de um aplicativo concedendo e negando acessos solicitados e observando o comportamento quando as configurações de pastas e sensores são negadas na instalação ou alteradas após a instalação.

©International Software Testing Qualifications Board

MAT-2.1.9: (K3) Preparar testes para verificar o impacto de um aplicativo no consumo de energia de um dispositivo e o impacto de seu estado de energia no aplicativo.

HO-2.1.9: (H3) Testar um aplicativo sob vários níveis de energia da bateria para descobrir dados de consumo e estabelecer o desempenho sob estados de bateria baixa e inativa.

2.2 Teste de interações de aplicativos com o software do dispositivo

MAT-2.2.1: (K3) Preparar testes para o tratamento de notificações pelo sistema em teste.

HO-2.2.1: (H2) Testar o efeito de receber notificações quando um aplicativo está no primeiro plano e no segundo plano. Testar o efeito de alterar as configurações de notificação na funcionalidade do aplicativo.

MAT-2.2.2: (K2) Descrever como os testes podem verificar a funcionalidade correta dos links de acesso rápido.

HO-2.2.2: (H3) Testar um aplicativo para a funcionalidade de atalho ou acesso rápido.

MAT-2.2.3: (K3) Preparar testes para o impacto em um aplicativo das configurações de preferência do usuário fornecidas por um sistema operacional.

HO-2.2.3: (H3) Testar um aplicativo em execução alterando as opções de valor de entrada para as preferências fornecidas pelo sistema operacional.

MAT-2.2.4: (K2) Distinguir entre diferentes testes exigidos para aplicações nativas, web e híbridas.

HO-2.2.4: (H0) Opcionalmente, identificar os testes necessários para aplicativos, dependendo de seu tipo.

MAT-2.2.5: (K1) Testar chamadas necessárias para aplicativos que estão disponíveis em várias plataformas ou versões do sistema operacional.

MAT-2.2.6: (K1) Testar a recuperação requerida para coexistência e interoperabilidade com outros aplicativos.

2.3 Teste de vários métodos de conectividade

MAT-2.3.1: (K2) Resumir os testes para teste de conectividade, incluindo aqueles entre redes, ao usar o *Bluetooth* e ao alternar para o modo de avião.

HO-2.3.1: (H0) Opcionalmente realizar testes em um aplicativo que está transferindo dados para o servidor quando o telefone alterna entre *Wi-Fi* e conectividade de dados com base na intensidade do sinal disponível.

©International Software Testing Qualifications Board

2.1 Teste de compatibilidade do dispositivo de hardware

2.1.1 Teste de recursos do dispositivo

Diferentes tipos de dispositivos com recursos diferentes significam que os testes de compatibilidade devem ser conduzidos em muitos dispositivos. Isso requer a priorização dos dispositivos destinados aos testes. Para dados de mercado de priorização, como discutido na seção 1.1, é usado para selecionar um portfólio de dispositivos mais apropriado para o mercado-alvo. A seleção do portfólio de dispositivos geralmente é um compromisso entre cobertura de mercado, custo e risco.

Os aplicativos podem ser instalados em diferentes tipos de dispositivos com os seguintes recursos:

- Diferentes métodos para desligar.
- Diferentes maneiras de navegar.
- Uso de diferentes tipos de teclado (físico e virtual).
- Vários recursos de hardware, como:
  - Rádio.
  - USB.
  - Bluetooth.
  - Câmeras.
  - Alto-falantes.
  - Microfone.
  - Acesso pelo fone de ouvido.

Nenhum desses recursos deve interferir negativamente nas operações do aplicativo.

Os recursos do dispositivo possuem muitas variações podendo ser diferentes mesmo entre modelos de dispositivos feitos pelo mesmo fabricante. Eles são comumente usados para diferenciar entre segmentos de mercado e podem mudar rapidamente ao longo do tempo. Por exemplo, atualmente é bastante comum que os dispositivos *top-de-linha* e medianos tenham sensores de impressão digital, enquanto os dispositivos de baixo custo não. Isso muda com o tempo. Alguns anos atrás, os sensores de impressões digitais não foram incluídos em nenhum dispositivo móvel. Devido a essa capacidade de mudança, o testador precisa ter um entendimento claro dos dispositivos e dos recursos esperados pelos usuários. O testador precisa criar o portfólio de dispositivos e projetar os testes correspondentes de acordo.

Geralmente, não é suficiente testar se o aplicativo funciona corretamente com os recursos esperados. Além disso, é necessário testar se o aplicativo ainda funciona como esperado se um determinado recurso estiver ausente. Por exemplo, um aplicativo que suporte o uso da câmera frontal e traseira não deve travar se for instalado e executado em um dispositivo com várias câmeras, apenas uma câmera ou nenhuma câmera.

2.1.2 Teste de diferentes exibições

As exibições de dispositivos podem ter vários tamanhos de tela, tamanhos de janelas de exibição, taxas de proporção e resoluções medidas em pixels por polegada (ppi) e pontos por polegada (dpi). A fragmentação do dispositivo requer que a priorização seja executada. Devem ser criados testes que

©International Software Testing Qualifications Board

exercitem a interface do usuário em vários dispositivos com diferentes tamanhos de tela, resoluções e relações de aspecto mais comuns no mercado-alvo.

O teste de diferentes monitores precisa ser realizado para verificar o seguinte:

- O aplicativo dimensiona todos os elementos da interface do usuário de acordo com a densidade e o tamanho atuais da tela.
- Os elementos da interface do usuário não se sobrepõem.
- Problemas de usabilidade ou toque não ocorrem.
- Não há encolhimento problemático de imagens por causa de alta dpi ou ppi.

2.1.3 Teste de temperatura do dispositivo

Ao contrário dos computadores de mesa, os dispositivos móveis reagem de maneira diferente ao aumento da temperatura do dispositivo.

Os dispositivos móveis podem ficar superaquecidos por vários motivos, como carga da bateria, carga de trabalho intensa, aplicativos em segundo plano, uso contínuo de dados de celular, *Wi-Fi* ou GPS.

O superaquecimento pode afetar um dispositivo, pois ele tenta reduzir o aquecimento e economizar os níveis de bateria. Isso pode incluir uma queda na frequência da CPU, a liberação de memória e o desligamento de partes do sistema.

Se isso acontecer, também pode afetar a funcionalidade do aplicativo e, portanto, deve ser considerado ao planejar o teste. Os testes devem ser projetados para consumir muita energia, o que leva à geração de calor durante um longo período ininterrupto. O software em teste deve, então, não mostrar nenhum comportamento inesperado.

2.1.4 Teste de sensores de entrada do dispositivo

Os dispositivos móveis recebem uma variedade de tipos de entradas de sensores que usam, por exemplo, GPS, acelerômetros, giroscópios e magnetômetros de 3 eixos, ou que reagem à pressão, temperatura, umidade, batimentos cardíacos, luz ou entradas sem toque.

O teste de diferentes sensores de entrada de dispositivo verifica o seguinte:

- O aplicativo funciona como planejado para cada um dos sensores disponíveis. Por exemplo, o aplicativo precisa ser testado para vários tipos de movimento, como movimento circular e movimento para trás e para frente (como em caminhada).
- Recursos que reagem à iluminação externa reagem corretamente sob várias condições de iluminação.
- As entradas e saídas de som respondem corretamente em conjunto com os botões de volume físico e de hardware, microfones, alto-falantes com e sem fio e em várias condições de som ambiente.
- A posição do local é precisa nas seguintes condições:
    - Ligar e desligar o GPS.
    - Qualidade de sinal GPS diferente.

©International Software Testing Qualifications Board

- Onde o aplicativo precisa de um retorno a vários outros métodos de determinação de localização, incluindo Wi-Fi, localização da torre de celular ou entrada de localização manual.

2.1.5 Teste de vários métodos de entrada

O teste de diferentes métodos de entrada de dispositivo verifica o seguinte:

- Dado que os telefones celulares permitem a instalação de uma variedade de teclados virtuais, o aplicativo pode funcionar com pelo menos aqueles fornecidos pelos principais fabricantes de dispositivos e aqueles que são amplamente usados.
- O aplicativo garante que o teclado seja exibido por padrão com o layout e teclas apropriadas, quando necessário.
- Quando um usuário coloca um ou mais dedos na tela de toque, o aplicativo interpreta esse padrão como um determinado gesto ou comando. Os gestos típicos incluem pressionar ou tocar, tocar duas vezes, tocar várias vezes, deslizar, arrastar e apertar para abrir ou fechar.
- Cada tela do aplicativo precisa responder corretamente aos gestos ou outros meios de entrada conforme apropriado, e ignorar todos os gestos ou entradas que não são suportados.
- As câmeras usadas pelos aplicativos podem capturar imagens e vídeos, digitalizar códigos de barras, *QR Codes,* documentos, e medir distâncias.
- Onde as câmeras frontal e traseira estão disponíveis, a câmera apropriada é ativada por padrão. Por exemplo, quando um bate-papo por vídeo exige que a câmera frontal seja ligada por padrão, os aplicativos precisam ser testados nos casos em que o aplicativo usa a entrada da câmera e onde ela não está. Além disso, os testes devem garantir que o software em teste funcione corretamente se houver apenas uma câmera (frontal ou traseira) em vez de duas. Isso é especialmente verdadeiro se o software em teste usa uma câmera específica e é esse que está faltando.

2.1.6 Teste de mudança de orientação da tela

Os sensores de movimento são usados para detectar mudanças na orientação e acionar uma alternância entre os modos paisagem e retrato (e vice-versa) com alterações de layout feitas na interface do usuário, conforme necessário.

Testes após uma mudança de orientação da tela, verificam:

- Corrigir a usabilidade e o comportamento funcional quando uma mudança para o modo retrato ou paisagem é executada.
- O aplicativo mantém seu estado.
- Os campos de dados de entrada retêm os dados já capturados.
- Os campos de dados de saída exibem os mesmos dados enquanto mantém a sessão atual.

Os testes após uma alteração na orientação da tela não devem se concentrar apenas em um único movimento, pois os problemas de renderização ou de estado podem nem sempre aparecer após uma única alteração. Portanto, os testes devem ser realizados com várias movimentações ininterruptas entre os modos retrato e paisagem.

©International Software Testing Qualifications Board

Testes que alternam a orientação várias vezes nos vários estados de uma interface de usuário, com e sem dados, devem ser modelados. O aplicativo deve se comportar como esperado, persistindo o estado sem qualquer perda ou alteração de dados.

2.1.7 Teste de interrupções

Tipos comuns de interrupções de dispositivos incluem chamadas de voz, mensagens, carregador ligado, pouca memória e outras notificações. As interrupções iniciadas pelo usuário resultam de ações como a alternância de aplicativos ou a configuração do dispositivo no modo de espera enquanto o aplicativo está sendo executado. Testes de interrupções verificam:

- Se o aplicativo lida com todas as interrupções mencionadas acima, sem impacto negativo no comportamento do aplicativo.
- Se o aplicativo continua a funcionar corretamente, preservando seu estado, dados e sessões, independentemente de qual interrupção ocorreu.
- Quando o dispositivo tem um modo de bloqueio de não perturbe que suprime as notificações, o aplicativo deve garantir que as várias condições sejam usadas corretamente. Estes testes também devem ser realizados quando o modo "não perturbe" é desativado após ter estado ativo por um longo período. Isso resulta em muitas notificações sendo recebidas de uma só vez.
- O teste deve ser projetado para receber interrupções durante o uso do aplicativo garantindo que elas não tenham impacto negativo. Por exemplo, responder a uma chamada telefônica enquanto usa o aplicativo e o usuário retorna ao estado em que ele estava no momento da interrupção.

2.1.8 Teste de permissões de acesso para recursos do dispositivo

Os aplicativos precisam de acesso a várias pastas, como contatos e imagens, e a sensores, como câmera e microfone. Quando o acesso é negado na instalação ou alterado após a instalação, isso pode afetar o comportamento do aplicativo.

Os testes para permissões de acesso verificam:

- Se o aplicativo é capaz de trabalhar com permissões reduzidas; ele pede ao usuário para conceder essas permissões e não falha de maneira inexplicável.
- Se as permissões só são solicitadas para os recursos relevantes para a funcionalidade do aplicativo. Nenhuma permissão ampla para recursos não relacionados é permitida.
- A funcionalidade do aplicativo responde corretamente se uma permissão for retirada ou rejeitada durante a instalação.
- Qualquer solicitação de permissão emitida pelo aplicativo está correta e justificada.

Para testar as permissões de acesso, um testador precisa saber por que o aplicativo precisa de cada permissão e como a funcionalidade deve ser afetada se a permissão for retirada ou rejeitada durante a instalação. O teste deve ser projetado para rejeitar permissões durante a instalação, bem como conceder permissões após a instalação.

©International Software Testing Qualifications Board

2.1.9 Teste de consumo de energia e estado

Os testes para consumo de energia e estado verificam:

- O estado da bateria e defeitos relacionados à drenagem.
- A integridade dos dados em condições de baixa potência e bateria descarregada.
- O consumo de energia enquanto o aplicativo está ativo e está sob uso pesado e baixo.
- O consumo de energia enquanto o aplicativo está em segundo plano.

Esses testes precisam ser planejados com cuidado, pois necessitam de execuções ininterruptas durante um longo período. Por exemplo, o dispositivo pode precisar ser deixado desacompanhado com o aplicativo em segundo plano ou em primeiro plano, mas o dispositivo não é usado. Ferramentas como analisadores de log são necessárias para extrair informações sobre os padrões de drenagem de bateria.

2.2 Teste de interações de aplicativos com o software do dispositivo

2.2.1 Teste de notificações

Existem vários mecanismos usados pelo sistema operacional para exibir notificações. Às vezes, o sistema operacional atrasará a exibição das notificações ou deixará de exibi-las em um lance para otimizar o consumo de energia. As seguintes condições de teste devem ser consideradas:

- O tratamento correto das notificações recebidas quando o aplicativo está em primeiro ou segundo plano, especialmente sob condições de bateria fraca.
- Se as notificações permitirem interação direta com o conteúdo do aplicativo (ou seja, sem abrir o aplicativo em si), a interação do usuário deverá ser fornecida pelo aplicativo posteriormente. Se, por exemplo, o usuário responder a uma notificação, deverá ser possível acessar essa resposta posteriormente de dentro do aplicativo.
- Se as notificações permitirem o acesso ao aplicativo, a página correspondente deverá ser aberta ao invés da tela inicial quando a notificação contiver um link direto para essa página.

2.2.2 Teste em links de acesso rápido

Links de acesso rápido, como atalhos de aplicativos no Android e *force-touch* ou *3D-touch* para iOS, podem ser fornecidos pelo software em teste. Esses recursos executam um subconjunto da funcionalidade do aplicativo na tela inicial sem realmente iniciar o aplicativo inteiro.

As seguintes condições de teste devem ser consideradas:

- Quando alguns dos recursos estão disponíveis apenas em uma versão específica do sistema operacional, o sistema em teste deve se comportar corretamente se estiver instalado em versões do sistema operacional que ofereçam ou não tais recursos.
- As ações realizadas em links de acesso rápido são refletidas corretamente no aplicativo quando abertas.

©International Software Testing Qualifications Board

2.2.3 Teste de preferências do usuário fornecidas pelo sistema operacional

Quaisquer preferências (configurações) fornecidas aos usuários pelo sistema operacional devem ser testadas. Isso cria uma experiência negativa para os usuários se uma determinada configuração de preferência não for respeitada pelo aplicativo. Por exemplo, se o dispositivo estiver definido como mudo, o aplicativo não deve reproduzir sons.

As seguintes condições de teste devem ser consideradas:

- Os usuários podem alterar as opções típicas de preferências, como som, brilho, rede, modo de economia de energia, data e hora, fuso horário, idiomas, tipo de acesso e notificações.
- Os aplicativos aderem às preferências do conjunto, comportando-se de acordo.

2.2.4 Teste em diferentes tipos de aplicativos

Testes específicos podem ser realizados dependendo do tipo de aplicativo para dispositivo móvel (consulte a seção 1.4). As seguintes condições de teste devem ser consideradas:

Para aplicativos nativos:
- Compatibilidade de dispositivos.
- Utilização de recursos do dispositivo.

Para aplicativos híbridos:
- Interação do aplicativo com os recursos nativos do dispositivo.
- Potenciais problemas de desempenho devido à camada de abstração.
- Usabilidade (aparência e comportamento) em comparação com aplicativos nativos na plataforma em questão.

Para aplicativos da web:
- Teste para determinar a compatibilidade entre navegadores do aplicativo para os navegadores móveis mais comuns.
- Funcionalidade não é afetada devido a vários mecanismos *JavaScript*.
- Utilização de recursos do sistema operacional (p. ex.: selecionador de data e abertura do teclado apropriado).
- Usabilidade (aparência e comportamento) em comparação com aplicativos nativos na plataforma em questão.

2.2.5 Teste de interoperabilidade com várias plataformas e versões do sistema operacional

As empresas de software geralmente oferecem suporte a aplicativos em vários sistemas operacionais. Cada sistema operacional móvel tem suas próprias limitações, que precisam ser levadas em conta ao testar aplicativos. Os testadores devem estar cientes das especificidades de cada plataforma testada para garantir que o aplicativo funcione como pretendido, enquanto ainda esteja em conformidade com a aparência e comportamento da plataforma.

As seguintes condições de teste devem ser consideradas:

©International Software Testing Qualifications Board

- Manipulação de interrupções, notificações e otimizações (p. ex.: para economia de energia).
- Funcionalidade correta em que aplicativos multiplataforma compartilham algum código ou foram criados usando estruturas de plataformas cruzadas de desenvolvimento. Observe que, se os aplicativos não compartilham o código, é como testar dois aplicativos diferentes e tudo precisa ser testado.
- Teste de compatibilidade com versões anteriores se uma plataforma usar versões diferentes do sistema operacional.
- Teste de novos recursos ou alterados, construídos em plataformas. Por exemplo, no Android, a introdução do *Framework Doze* exigiu testes nas várias versões do sistema operacional que suportam ou não este framework.

2.2.6 Teste de interoperabilidade e coexistência com outros aplicativos no dispositivo

É bastante comum que os aplicativos interajam uns com os outros quando instalados em um dispositivo. Exemplos típicos são os aplicativos de contato e e-mail.

As seguintes condições de teste devem ser consideradas:

- Se a transferência de dados entre o sistema em teste e o aplicativo utilizado está correta.
- Se não há nenhum dano em nenhum dado do usuário armazenado em um aplicativo utilizado.
- Se não há comportamentos conflitantes. Por exemplo, um aplicativo pode desativar o GPS para economizar energia, enquanto outro aplicativo ativa o GPS automaticamente.

Com milhões de aplicativos no mercado, a coexistência não pode ser testada realisticamente para todos eles. No entanto, tais questões potenciais devem ser consideradas e testadas de acordo com seu risco.

2.3 Teste de vários métodos de conectividade

Os dispositivos móveis podem usar vários métodos para se conectar a redes (consulte a seção 1.5). Estes incluem redes celulares como 2G, 3G, 4G e 5G, bem como *Wi-Fi* e outros tipos de conexão sem fio, como NFC ou *Bluetooth*.

As seguintes alternativas devem ser consideradas ao realizar os testes de conectividade:

- Emuladores ou simuladores de dispositivos podem simular várias conexões de rede, e alguns provedores de serviços de acesso à dispositivos remotos os incluem dentro de seus recursos. Emuladores ou simuladores são, no entanto, de valor limitado.
- Configurar sua própria rede móvel para suportar vários tipos de conexão e, em seguida, aplicar a manipulação de banda larga. Esta é uma alternativa muito cara.
- O teste de campo é uma alternativa potencialmente mais econômica, mas é limitado no que diz respeito à reprodução de testes.

No mundo real, os métodos de conectividade são diferentes. Os usuários podem ser conectados continuamente usando um modo específico ou podem alternar entre os modos, como de *Wi-Fi* para celular (p. ex.: quando um usuário sai de casa usando o aplicativo). O usuário pode alternar entre várias

©International Software Testing Qualifications Board

redes e versões de *Wi-Fi* em celulares, bem como entre células GSM. Enquanto estão em movimento, eles podem até atingir pontos mortos sem rede. Além disso, o usuário pode deliberadamente desconectar, por exemplo, alternando para o modo avião.

O teste de conectividade deve garantir que as seguintes condições de teste sejam consideradas:

- Corrigir a funcionalidade do aplicativo com diferentes modos de conectividade.
- Alternar entre os modos não causando nenhum comportamento inesperado ou perda de dados.
- Informar claramente ao usuário se a funcionalidade estiver restrita devido à conexão de rede limitada ou nula ou se a largura de banda for baixa. A mensagem deve indicar as limitações e suas razões.

©International Software Testing Qualifications Board

3 Tipos e processos de teste [200 min]

Palavras-chave

final anormal, acessibilidade, injeção de código, teste exploratório, teste de campo, heurística, instalação, eficiência de desempenho, teste de desempenho, teste pós-lançamento, teste de segurança, gerenciamento de teste baseado em sessão, teste de estresse, nível de teste, processo de teste, pirâmide de teste, roteiro, laboratório de usabilidade, teste de usabilidade

Objetivos de aprendizado

3.1 Tipos comuns e aplicáveis de teste

MAT-3.1.1: (K3) Preparar testes de instalação para aplicativos para dispositivos móveis.

MAT-3.1.2: (K3) Preparar testes de estresse para aplicativos para dispositivos móveis.

MAT-3.1.3: (K2) Dar exemplos de problemas de segurança relacionados a aplicativos para dispositivos móveis.

MAT-3.1.4: (K1) Lembrar-se de considerações de comportamento de recursos e tempo para aplicativos para dispositivos móveis.

MAT-3.1.5: (K3) Preparar testes de usabilidade para aplicativos para dispositivos móveis.

HO-3.1.5: (H2) Escolher um roteiro, um mnemônico ou uma heurística para testar a usabilidade de um aplicativo usando o gerenciamento de teste baseado em sessão.

Nota: HO-3.1.5 e HO-3.3.1, HO-3.3.2 e HO-3.3.3 podem ser combinados.

MAT-3.1.6: (K1) Reconhecer os tipos de testes necessários para o teste de banco de dados de aplicativos para dispositivos móveis.

MAT-3.1.7: (K2) Resumir os testes necessários para internacionalização (globalização) e testes de regionalização de aplicativos para dispositivos móveis.

MAT-3.1.8: (K2) Resumir a necessidade dos testes de acessibilidade em testes de aplicativos para dispositivos móveis.

3.2 Níveis de testes adicionais aplicáveis

MAT-3.2.1: (K2) Descrever os níveis de teste adicionais, como testes de campo, e as atividades extras necessárias associadas para um teste eficaz de aplicativos para dispositivos móveis.

MAT-3.2.2: (K2) Descrever os testes necessários para executar a aprovação da loja de aplicativos para aplicativos que serão publicados.

3.3 Técnicas de teste baseadas na experiência

MAT-3.3.1: (K1) Relembrar o gerenciamento de testes baseado em sessão, personas e mnemônicos no contexto de testes móveis exploratórios.

HO-3.3.1: (H2) Escolher um mnemônico (ou parte dele) que seja específico para o teste de aplicativo para dispositivo móvel para testar um aplicativo usando o gerenciamento de teste baseado em sessão.

©International Software Testing Qualifications Board

Nota: HO-3.1.5 e HO-3.3.1, HO-3.3.2 e HO-3.3.3 podem ser executadas juntas.

MAT-3.3.2: (K2) Descrever o uso de roteiros e heurísticas como técnicas exploratórias para testes de aplicativos para dispositivos móveis.

HO-3.3.2: (H2) Escolher uma heurística específica para dispositivos móveis para testar o aplicativo para dispositivos móveis.

Nota: HO-3.1.5 e HO-3.3.1, HO-3.3.2 e HO-3.3.3 podem ser combinados.

MAT-3.3.3 app. (K3) Fazer uso de um roteiro específico para celular (como o roteiro por recursos) para testar um dispositivo móvel.

HO-3.3.3: (H2) Escolher um roteiro específico para dispositivos móveis para testar um aplicativo para dispositivos móveis.

Nota: HO-3.1.5 e HO-3.3.1, HO-3.3.2 e HO-3.3.3 podem ser combinados.

3.4 Processo e abordagem do teste

MAT-3.4.1: (K2) Corresponder o processo de teste, conforme descrito em [ISTQB_FL_2018], às necessidades dos testes de aplicativos para dispositivos móveis.

MAT-3.4.2: (K2) Descrever as abordagens para testes em cada nível de teste, especificamente para testes de aplicativos para dispositivos móveis.

©International Software Testing Qualifications Board

3.1 Tipos comuns aplicáveis de teste

3.1.1 Testes de instalação

Os testadores precisam se concentrar na instalação, atualização e desinstalação do aplicativo usando as seguintes abordagens:

Lojas de aplicativos

O processo de instalação pode ser diferente dependendo dos usuários do aplicativo. Os usuários podem instalar o aplicativo em lojas de mercado, como a *Google Play Store* ou a *App Store* da Apple. Os usuários de aplicativos corporativos precisarão realizar testes de instalação por meio de um link ou de um serviço de distribuição, como o *HockeyApp* ou o *App Center*.

***Sideloading* (copiando e instalando o aplicativo)**

Alguns sistemas operacionais oferecem a opção de instalar o aplicativo copiando-o em um dispositivo móvel e instalando-o a partir do arquivo.

Aplicativos de desktop

Aplicativos de área de trabalho, como o *Apple iTunes* (para iOS) ou o *Android App Installer*, estão disponíveis para a instalação de aplicativos no smartphone. O testador precisa baixar o aplicativo neste aplicativo e usar um cabo para instalá-lo no smartphone. A maioria desses aplicativos de desktop também permite a desinstalação do aplicativo.

A instalação pode ser executada usando os seguintes métodos:

- OTA (*Over-the-Air*) via *Wi-Fi* ou dados de celular
- Cabo de dados

Algumas das condições de teste que podem ser consideradas incluem:

- Instalação, desinstalação e atualização na memória interna e externa (se suportado).
- Reinstalação do aplicativo quando a opção "*reter dados do aplicativo*" foi escolhida durante a desinstalação anterior.
- Reinstalação do aplicativo quando a opção "*reter dados do aplicativo*" não foi escolhida durante a desinstalação anterior.
- Cancelar ou interromper a instalação ou desinstalação, por exemplo, desligando o dispositivo móvel durante o processo ou desconectando-se da Internet.
- Retomando a instalação interrompida, desinstalação e atualização após o cancelamento ou interrupção.
- Testes relacionados a permissões. Por exemplo, alguns aplicativos solicitam permissão para usar o catálogo de endereços. Esse teste importante deve verificar o comportamento do aplicativo se o usuário negar permissão. Por exemplo, há uma mensagem correspondente enviada ao usuário?
- Atualizar o aplicativo e verificar se nenhum dado foi perdido.

©International Software Testing Qualifications Board

Alguns aplicativos exigem dispositivos *jailbroken* (iOS) ou *rooted* (Android) que dão ao usuário os direitos administrativos sobre o dispositivo. A maioria dos provedores de plataforma não oferece suporte a ambos, pois isso pode ter consequências legais. Um aplicativo que não requer essas permissões talvez não precise ser testado para os dispositivos de *jailbreak* ou *rooted*.

3.1.2 Teste de estresse

O teste de estresse está focado em determinar a eficiência de desempenho do aplicativo quando sujeito a condições além da carga normal. O teste de estresse neste contexto é direcionado apenas para o dispositivo móvel. Os testes de estresse do *backend* são descritos no programa *ISTQB® Performance Testing* ([ISTQB_FLPT_2018]) para que informações adicionais possam ser referenciadas conforme necessário.

Algumas das condições de teste que podem ser consideradas para testes de estresse incluem:

- Alto uso da CPU.
- Despejo de memória.
- Pouco espaço em disco.
- Estresse da bateria.
- Falhas.
- Largura de banda ruim
- Número muito alto de interações do usuário (as condições da rede do mundo real talvez precisem ser simuladas para isso)

Algumas dessas condições estressantes podem ser criadas usando ferramentas como o *Monkey*. Essa é uma ferramenta de linha de comando que é executada na linha de comando do *shell* do ADB [URL3] ou, se possível, manualmente, por exemplo, usando arquivos grandes ou outros aplicativos com alto uso da CPU ou consumo de memória.

3.1.3 Testes de segurança

Como o teste de segurança é um tópico complexo, o ISTQB® tem um syllabus especializado separado sobre este assunto [ISTQB_ALSEC_2016]. Os principais problemas de segurança para aplicativos para dispositivos móveis incluem:

- Acesso a dados confidenciais no dispositivo.
- Transferência de informações não criptografada ou armazenamento inseguro.

Algumas das condições de teste que podem ser consideradas para testes de segurança incluem:

- Teste de entradas para injeção e estouro de código.
- Criptografia de dados transferidos.
- Criptografia de dados armazenados localmente.
- Exclusão de dados temporários após o uso ou após um término anormal.
- Limpeza dos dados dos campos de senha.

©International Software Testing Qualifications Board

As 10 principais vulnerabilidades relacionadas a dispositivos móveis do *Open Web Application Security Project* (OWASP) também devem ser exploradas [URL2].

3.1.4 Teste de performance

Se o usuário instalar o aplicativo e ele não aparecer rápido o suficiente (p. ex.: menor ou igual a 3 segundos), ele poderá ser desinstalado em favor de outro aplicativo alternativo. Os aspectos de tempo e consumo de recursos são fatores importantes de sucesso para um aplicativo e testes de performance são realizados para medir esses aspectos.

A eficiência de performance precisa ser testada no próprio dispositivo, além da interação com o sistema *backend* e outros dispositivos móveis.

O teste de performance de todo o sistema deve ser executado conforme definido na estratégia de teste e não é específico para celular. Por favor, consulte o programa *ISTQB® Performance Testing* [ISTQB_FLPT_2018] para mais detalhes.

O teste de performance do aplicativo em si deve conter cronometria para os fluxos de trabalho mais importantes. Alguns exemplos para os fluxos de trabalho de um aplicativo bancário on-line são: “Login”, “Alterar endereço” ou “Transferência bancária com PIN e TAN”. O testador deve então comparar essa cronometria com aplicativos semelhantes.

Além de medidas cronométricas, é importante considerar a performance percebida pelo usuário. A experiência do usuário pode ter um grande impacto em quanto tempo o usuário está disposto a esperar até que uma determinada função seja concluída.

3.1.5 Teste de usabilidade

A usabilidade é muito importante para aplicativos para dispositivos móveis porque os dados mostram que muitos usuários desinstalam seus aplicativos em poucos minutos após a instalação devido à má usabilidade ou performance, consulte [URL4].

Devido a isso, é recomendável que o desenho da experiência do usuário (UX) considere a aparência da plataforma na qual o aplicativo será usado. Se o UX não estiver de acordo com as expectativas do usuário em relação à plataforma escolhida, ele poderá ter um forte impacto negativo. Assim, um testador deve estar ciente da aparência e comportamento da plataforma usada.

Os testes de usabilidade podem ser realizados por um testador usando várias heurísticas disponíveis e roteiros de teste. Considerar pessoas também é um suporte útil para testes de usabilidade. Se necessário, um laboratório de usabilidade também pode ser usado para essa finalidade.

Nos projetos, as descobertas identificadas durante o teste de usabilidade são, na maioria das vezes, apenas descobertas e não defeitos. O testador deve ter a capacidade de explicar as descobertas à equipe, ao dono do produto ou a *Stakeholders* semelhantes. Para alcançar uma usabilidade satisfatória, um aplicativo deve:

- Ser autoexplicativo e intuitivo.
- Permitir erros do usuário.
- Ser coerente na redação e no comportamento.

©International Software Testing Qualifications Board

- Respeitar as diretrizes de desenho das plataformas.
- Tornar as informações necessárias visíveis e acessíveis em cada tamanho e tipo de tela.

Por favor, consulte o programa *ISTQB® Usability Testing* [ISTQB_FLUT_2018] para mais detalhes.

3.1.6 Teste de banco de dados

Muitos aplicativos precisam armazenar dados localmente usando vários mecanismos de armazenamento, como arquivos simples ou bancos de dados. Algumas das condições de teste a serem consideradas para o teste de banco de dados de aplicativos para dispositivos móveis incluem:

- Validação de problemas de armazenamento de dados:
  - Sincronização.
  - Conflitos de carga.
  - Segurança de dados.
  - Restrições nos dados.
  - Funcionalidade CRUD (Criar / Ler / Atualizar / Apagar).
- Pesquisa.
- Teste de integração de dados para dados fornecidos pelo dispositivo (p. ex.: contatos) ou por aplicativos de terceiros (p. ex.: fotos, vídeos e mensagens).
- Performance em armazenar os dados no dispositivo.

3.1.7 Teste de internacionalização e regionalização

A internacionalização (I18N) ou teste de globalização do aplicativo inclui o teste de um aplicativo para diferentes locais, formatos de datas, números e moeda e a substituição de cadeias reais por *pseudo-strings*.

O teste de regionalização (L10N) inclui o teste de um aplicativo com *strings*, imagens e fluxos de trabalho localizados para uma região específica. Por exemplo, palavras em russo e alemão podem ser muito mais longas do que em outros idiomas. Como os dispositivos móveis têm tamanhos e resoluções de tela diferentes, tamanhos de tela limitados podem causar problemas com as sequências traduzidas. Esses problemas devem ser verificados como testes padrão de internacionalização e regionalização.

Um aspecto muito importante a ser verificado é o formato de data utilizado, como ANO - MES- DIA ou DIA - MES - ANO.

3.1.8 Teste de acessibilidade

O teste de acessibilidade é realizado para determinar a facilidade com a qual os usuários com deficiências podem usar um componente ou sistema. Para aplicativos para dispositivos móveis, isso pode ser feito usando as configurações de acessibilidade do dispositivo e testando o aplicativo para cada configuração.

As diretrizes de acessibilidade estão disponíveis nos fornecedores de plataformas e devem ser usadas. Por exemplo, o Google [URL5] e a Apple [URL6] publicaram diretrizes de acessibilidade para suas respectivas plataformas. Receber feedback de pessoas que precisam de acessibilidade também é útil.

©International Software Testing Qualifications Board

Para web móvel, um guia de acessibilidade foi publicado pelo W3C, que deve ser considerado [URL7].

3.2 Níveis de teste adicionais aplicáveis

Além dos níveis usuais de teste do componente até o teste de aceite descrito em [ISTQB_FL_2018], há também a necessidade de níveis de teste adicionais para o teste de aplicativos para dispositivos móveis.

3.2.1 Teste de campo

Alguns aplicativos para dispositivos móveis precisam de testes de campo para garantir que funcionem corretamente no cenário de uso esperado de usuários reais. Isso pode incluir testes em várias redes e em diferentes tipos de tecnologias de comunicação, como dados de *Wi-Fi* ou celular.

Os testes de campo devem incluir o uso de torres móveis, redes, *Wi-Fi* e troca de dados de celular enquanto o aplicativo estiver em uso. Os testes devem ser realizados com velocidades de *download* e intensidade de sinal variáveis e incluem o manuseio de pontos cegos.

O teste de campo requer um planejamento cuidadoso e a identificação de todos os itens necessários para realizar os testes, como tipos de dispositivos apropriados, *Wi-Fi*, planos de dados de celulares em várias operadoras e acesso a vários modos de transporte necessários para fornecer cobertura adequada. Além disso, as rotas e modos de transporte e a hora do dia em que os testes devem ser executados precisam ser programados.

A usabilidade de um aplicativo é outro aspecto importante que precisa ser coberto durante a realização de testes de campo. Os testes devem incorporar fatores ambientais, como temperatura e condições similares relacionadas ao cenário de uso.

3.2.2 Teste de aprovação de armazenamento de aplicativos e teste pós-lançamento

Antes de um aplicativo ser enviado para publicação, alguns testes baseados em lista de verificação devem ser passados para garantir a aprovação dos armazenamentos de aplicativos. Se a liberação for uma atualização, os testes relacionados à atualização também deverão ser executados.

As listas de verificação geralmente são baseadas em diretrizes, como aquelas específicas dos sistemas operacionais, no desenho da interface do usuário e no uso de bibliotecas e APIs fornecidas pelos armazenamentos de aplicativos.

O processo de aprovação pode levar algum tempo após o envio. Se algum problema for encontrado durante o processo de aprovação, talvez seja necessário enviar uma nova versão, o que exigirá mais tempo para ser resolvido. Essa situação requer consideração cuidadosa durante o planejamento e o teste do projeto.

Um outro nível de teste é o teste "*pós-lançamento*". O teste nesse nível inclui o download e a instalação do aplicativo nos armazenamentos de aplicativos.

©International Software Testing Qualifications Board

3.3 Técnicas de teste baseadas na experiência

3.3.1 Personas e mnemônicos

As **personas** são personagens fictícios que representam clientes reais. Eles têm motivações, expectativas, problemas, hábitos e objetivos, e ajuda a usá-los quando o comportamento real do usuário precisa ser imitado.

Uma pessoa pode ter um nome, sexo, idade, renda, formação educacional e localização. Em um contexto móvel, eles podem usar outros aplicativos, verificar seu dispositivo móvel "x" vezes por hora e ter outros dispositivos e características pessoais.

Um **mnemônico** é uma ajuda de memória para lembrar de algo. No contexto do teste, cada letra em um mnemônico significa uma técnica, um método de teste ou um ponto focal para teste. Um exemplo de um mnemônico é o SFiDPOT [URL8]. As letras no mnemônico têm os seguintes significados:

- `[S]` **Structure** (Estrutura, por exemplo, elementos da interface do usuário, outros elementos do aplicativo e sua ordem e hierarquia de chamadas)
- `[F]` **Function** (Função, por exemplo, os recursos desejados estão funcionando, disponíveis, e trabalhando de acordo com os requisitos etc.)
- `[i]` **input** (Entrada, por exemplo, todas as entradas necessárias estão disponíveis e processadas como deveriam, como as entradas de teclado, sensores e câmera)
- `[D]` **Data** (Dados, por exemplo, os dados são armazenados (também no cartão SD), modificados, adicionados e excluídos conforme definido nos requisitos)
- `[P]` **Platform** (Plataforma, por exemplo, as funções específicas do sistema operacional estão disponíveis dependendo das configurações do dispositivo, incluindo armazenamento para baixar o aplicativo)
- `[O]` **Operations** (Operações, por exemplo, as atividades do usuário normal estão disponíveis, como a movimentação entre redes de operadoras de celular e Wi-Fi)
- `[T]` **Time** (Tempo, por exemplo, manuseio e exibição de fusos horários, hora e datas)

Um mnemônico e heurístico especificamente relacionado a dispositivos móveis é I SLICED UP FUN [URL9]. Letras no mnemônico têm os seguintes significados

- `[I]` **Inputs** (Entradas)
- `[S]` **Store** (Loja)
- `[L]` **Location** (Localização)
- `[I]` **Interactions and interruptions** (Interações e interrupções)
- `[C]` **Communication** (Comunicação)
- `[E]` **Ergonomics** (Ergonomia)
- `[D]` **Data** (Dados)
- `[U]` **Usability** (Usabilidade)
- `[P]` **Platform** (Plataforma)
- `[F]` **Function** (Função)
- `[U]` **User scenarios** (Cenários do usuário)

©International Software Testing Qualifications Board

- **[N] Network** (Rede)

3.3.2 Heurística

Uma abordagem heurística é uma abordagem "regra geral" para a resolução de problemas, aprendizagem e descoberta que emprega um método prático. Isso não garante ser ótimo ou perfeito, mas pode ser considerado suficiente para atingir os objetivos imediatos.

Existem muitas heurísticas para testes móveis. A maioria dos mnemônicos pode ser usada como heurística, mas nem toda heurística é um mnemônico.

3.3.3 Roteiros

Os roteiros são usados em testes exploratórios para permitir que um aplicativo seja explorado de um ponto de vista e foco específicos. Eles podem ser executados para entender como um aplicativo funciona e para criar modelos para o fluxo de trabalho. Os roteiros fornecem um método eficaz para testes de campo.

Um exemplo de um roteiro é o ***Landmark***, onde um usuário imita as visitas de um turista a uma cidade, indo para pontos de referência bem conhecidos. A tabela a seguir mostra como as visitas feitas no passeio podem ser usadas como analogias para as etapas a seguir nos testes para dispositivos móveis.

Visitas no Roteiro Landmark	**Analogia para testes móveis**
O bairro histórico	Código legado
O distrito comercial Hora do rush	Lógica de negócios do aplicativo Inicialização e desligamento de aplicativos
O bairro turístico	Parte do aplicativo usado pelos recém-chegados
O quarto do hotel	Partes do aplicativo que estão ativas somente no modo de suspensão

A combinação de testes baseados em sessão (consulte a seção 3.3.4) com roteiros, incluindo o uso de heurísticas e mnemônicos, ajuda a aprimorar a eficácia dos testes de aplicativos para dispositivos móveis.

A tabela a seguir mostra alguns bons exemplos de roteiros para testes de aplicativos e as áreas que eles cobrem para dar ideias de teste. Alguns deles podem ser encontrados em [Kohl17].

Roteiro por teste de aplicativos	**Assunto coberto**
Supermodelo	Aparência e usabilidade
Ponto de referência	Os recursos mais importantes do aplicativo
Sabotagem	Robustez
Característica	Novas características
Cenário	Fluxo de trabalho inteiro no aplicativo em combinação com histórias de usuários
Conectividade	Conectividade usada, como Wi-Fi, GSM
Regionalização	Linguagem correta, datas, números
Luz	Visibilidade em diferentes condições de iluminação, como escuro, exterior, luz vermelha.
Bateria Fraca	Perdas de dados no aplicativo causadas por baixos níveis de energia.

©International Software Testing Qualifications Board

Roteiro por teste de aplicativos	Assunto coberto
Gesto	Use todos os gestos sempre que possível
Orientação	Alterar orientação
Mude sua opinião	Volte
Movimento	Faça vários tipos de movimentos
Regionalização	Mover-se
Conectividade	Alterar tipos de conexão ou locais movendo
Comparação	Compare com outro tipo de dispositivos
Consistência	Verificar a consistência das telas, GUI

3.3.4 Gerenciamento de teste baseado em sessão (SBTM)

O *Session-Based Test Management* (SBTM), ou gerenciamento de teste baseado em sessão, permite que os testes exploratórios sejam gerenciados de forma casuística. Uma sessão consiste em três tarefas:

- Configuração da sessão
- Projeto e execução de testes
- Emissão de investigação e relatórios

O SBTM geralmente usa uma folha de sessão contendo um termo de teste que fornece objetivos de teste.

Além disso, a folha de sessão é usada para documentar as atividades de execução de teste realizadas.

O teste exploratório é uma técnica de teste baseada em experiência que pode ser uma abordagem eficaz para testar aplicativos para dispositivos móveis. Técnicas de teste baseadas em experiência são descritas em [ISTQB_FL_2018].

3.4 Processo e abordagem do teste

3.4.1 Processo de teste

As principais atividades do processo de teste do ISTQB® estão descritas no [ISTQB_FL_2018], sendo também aplicáveis aos testes de aplicativos para dispositivos móveis.

Existem aspectos adicionais que são específicos para testes móveis, que devem sempre ser considerados como parte do processo de teste do ISTQB®.

Principais de atividades no processo de teste	Áreas típicas a serem consideradas para testes móveis
Planejamento de teste	• Combinações de dispositivos que precisam ser testadas. • Uso de emuladores móveis e simuladores móveis como parte do ambiente de teste (**seção 4.3**).

©International Software Testing Qualifications Board

Principais de atividades no processo de teste	Áreas típicas a serem consideradas para testes móveis
	• Desafios especiais no teste de aplicativos para dispositivos móveis (**seção 1.7**). • Tipos de teste especificamente necessários para testes de aplicativos para dispositivos móveis (**seção 3.2**).
Análise e Modelagem	• Teste de aprovação do App Stores (**seção 3.2.2**). • Teste de Campo (**seção 3.2.1**). • Compatibilidade de dispositivos. • Tipo de laboratório a ser usado. • Tipos de teste especificamente necessários para testes de aplicativos para dispositivos móveis (**seção 3.2**).
Implementação e execução do teste	• Teste de Campo (**seção 3.2.1**). • Teste pós-lançamento de download e instalação (**seção 3.1.1**). • Técnicas baseadas em experiência (**[ISTQB_FL_2018]**). • Os testes são baseados em diretrizes de plataforma para a interface do usuário e os armazenamentos de aplicativos. • Testes baseados em diretrizes geralmente serão executados pelos provedores de plataforma para o processo de aprovação de armazenamento de aplicativos. • É recomendável executá-los como provedor de aplicativos antes de entregá-los aos provedores de plataforma para evitar possíveis rejeições.

3.4.2 Abordagens de teste

O teste de aplicativos para dispositivos móveis inclue atividades que serão executadas por desenvolvedores e testadores.

Determinar a profundidade apropriada do teste por nível de teste (ou seja, teste de componente, teste de integração, teste de sistema, teste de campo, aprovação da loja de aplicativos, teste de pós-lançamento e aceite do usuário) é importante para fornecer produtos de boa qualidade. A profundidade dos testes necessários por nível de teste depende de muitos fatores, como a arquitetura do aplicativo, a sua complexidade, e público-alvo de usuários.

As plataformas de desenvolvimento móvel fornecem várias ferramentas para suportar testes em seus vários níveis.

É muito importante entender as ferramentas, e como elas podem ser aplicadas em um determinado nível. Por exemplo, um simulador ou emulador de um dispositivo móvel pode ser usado no nível de teste de componente, se houver a necessidade de aproveitar as APIs de instrumentação e estrutura fornecidas pela plataforma. Além disso, os simuladores ou emuladores de dispositivos móveis podem

©International Software Testing Qualifications Board

ser usados no nível de teste do sistema quando os dispositivos reais não estiverem disponíveis. Isso permite testar a funcionalidade, alguns aspectos limitados de usabilidade, bem como a interface do usuário.

Além disso, a implementação inicial pode servir como um ponto-chave para garantir que os dispositivos sejam configurados corretamente e que todos os pré-requisitos para a execução sejam atendidos no prazo.

Testes de unidade e integração também são importantes, bem como os testes manuais (especialmente na fase de teste de campo). É muito comum que os aplicativos para dispositivos móveis abram a Pirâmide de Teste [Knott15]. Isso significa que pode haver muitos testes manuais.

©International Software Testing Qualifications Board

4 Plataformas de aplicativos, ferramentas e ambiente [80 min]

Palavras-chave

emulador, teste de campo, teste baseado na proximidade, laboratório de teste remoto, simulador

Objetivos de aprendizagem

4.1 Plataformas de desenvolvimento de aplicativos

MAT-4.1.1: (K1) Lembrar dos ambientes de desenvolvimento usados para o desenvolvimento de aplicativos para dispositivos móveis.

4.2 Ferramentas comuns da plataforma de desenvolvimento

MAT-4.2.1: (K1) Lembrar de algumas das ferramentas fornecidas como parte das plataformas de desenvolvimento de aplicativos.

HO-4.2.1: (H1) Usar as ferramentas do kit de desenvolvimento de software para fazer capturas de tela, extrair um registro, e simular eventos de entrada.

4.3 Emuladores e Simuladores

MAT-4.3.1: (K2) Compreender as diferenças entre emuladores e simuladores.

MAT-4.3.2: (K2) Descrever o uso de emuladores e simuladores para testes de aplicativos para dispositivos móveis.

HO-4.3.2: (H1) Criar e executar um dispositivo simulado ou emulado, instalar um aplicativo, e executar alguns testes nele.

4.4 Configurando um laboratório de teste

MAT-4.4.1: (K2) Distinguir entre as várias abordagens para configuração de um laboratório de teste.

©International Software Testing Qualifications Board

4.1 Plataformas de desenvolvimento de aplicativos

Os ambientes de desenvolvimento integrados (IDEs) estão disponíveis no mercado para vários desenvolvimentos de aplicativos para dispositivos móveis. Esses IDEs têm várias ferramentas que ajudam na criação, codificação, compilação, instalação, desinstalação, monitoramento, emulação, registro e teste de aplicativos.

Por exemplo, podem ser usados o *Android Studio* pode ser usado para desenvolvimento de aplicativos Android, e o *Xcode* para o desenvolvimento de aplicativos para iOS. Estes diferem dos IDEs normais pelo suporte adicional que eles oferecem para as plataformas móveis.

Algumas estruturas de desenvolvimento de plataforma cruzada também estão disponíveis, o que ajuda no desenvolvimento de aplicativos para dispositivos móveis. Eles são executados em várias plataformas e não exigem especificamente codificação.

4.2 Ferramentas comuns da plataforma de desenvolvimento

Os SDKs (kits de desenvolvimento de software) geralmente fornecem vários utilitários que são úteis no desenvolvimento e teste de aplicativos. Esses utilitários abrangem uma ampla variedade de finalidades, como captura de tela, extração de logs, envio de eventos aleatórios e notificações para o dispositivo, monitoramento de vários parâmetros, como memória e utilização da CPU, além da criação de dispositivos virtuais.

Alguns exemplos de tais ferramentas são o *Android Virtual Device* (AVD) *Manager*, o *Android Debug Bridge* (ADB), e o *Android Device Monitor* para plataforma Android, e *Instruments* para iOS.

4.3 Emuladores e simuladores

4.3.1 Visão geral de emuladores e simuladores

No contexto deste syllabus, os termos “emulador” e “simulador” referem-se ao emulador de dispositivos móveis ou ao simulador de dispositivos móveis. Os termos simulador e emulador às vezes são usados de forma intercambiável, mas incorretamente. Para definições, consulte o glossário no capítulo 8.

Um simulador modela o ambiente de tempo de execução, enquanto um emulador modela o hardware e utiliza o mesmo ambiente de tempo de execução que o hardware físico. As aplicações testadas em um simulador são compiladas em uma versão dedicada, que funciona no simulador, mas não em um dispositivo real. Assim, é independente do sistema operacional real.

Por outro lado, os aplicativos compilados para serem implantados e testados em um emulador são compilados como ***bytecode***, que também pode ser usado pelo dispositivo real.

Os simuladores e emuladores são muito úteis no estágio inicial do desenvolvimento, pois eles normalmente se integram aos ambientes de desenvolvimento e permitem rápida implantação, teste e monitoramento de aplicativos.

©International Software Testing Qualifications Board

Os simuladores às vezes também são usados como substitutos dos dispositivos reais nos testes. No entanto, isso é ainda mais limitado do que o uso de emuladores, já que o aplicativo testado em um simulador difere em nível de ***bytecodes*** do aplicativo que será distribuído.

Os emuladores também são usados para reduzir o custo dos ambientes de teste, substituindo dispositivos reais em alguns dos testes. Um emulador não pode substituir totalmente um dispositivo porque o emulador pode se comportar de uma maneira diferente do dispositivo móvel que ele tenta imitar. Além disso, alguns recursos podem não ser suportados, como (multi) touch, acelerômetro e outros. Isso é parcialmente causado por limitações da plataforma usada para executar o emulador.

4.3.2 Usando emuladores e simuladores

O uso de emuladores e simuladores para testes móveis pode ser útil por diversos motivos.

Cada ambiente de desenvolvimento de sistemas operacionais móveis geralmente vem com seu próprio emulador e simulador integrados. Também estão disponíveis emuladores e simuladores de terceiros.

Um testador pode usar qualquer emulador ou simulador adequado ao seu propósito. Usar o emulador ou o simulador requer a sua execução, instalação do aplicativo necessário, e teste desse aplicativo como se estivesse em um dispositivo real.

Geralmente, os emuladores e simuladores permitem a configuração de vários parâmetros de uso. Essas configurações podem incluir emulação de rede em diferentes velocidades, intensidade de sinal e perdas de pacote, alteração de orientação, geração de interrupções e dados de localização por GPS. Algumas dessas configurações podem ser muito úteis, pois podem ser difíceis ou dispendiosas para replicar com dispositivos reais, como posições de GPS globais ou intensidade de sinal.

Conectar-se aos emuladores para fins de instalação pode exigir o uso de ferramentas de linha de comando, como o *Android Debug Bridge* (ADB) para Android ou a conexão de dentro do ambiente de desenvolvimento integrado, como no *Xcode* ou no *Android Studio*.

4.4 Configurando um laboratório de teste

As abordagens a seguir são usadas para configurar um laboratório de teste móvel:

Laboratório local

Com um laboratório local, todos os dispositivos, emuladores e simuladores estão localizados no próprio ambiente. A seleção de dispositivos pode ser feita com base em vários fatores, como classificação do dispositivo (conforme encontrado no Google ou outras análises), sistemas operacionais e versões, dimensões e densidades da tela, disponibilidade e custo, recursos especiais e importância no público-alvo.

As vantagens do laboratório local incluem a disponibilidade do dispositivo para testes específicos baseados em proximidade e aspectos específicos de sensores, bateria, toque e segurança aprimorada.

A configuração desse tipo de laboratório pode exigir grandes orçamentos, dependendo dos dispositivos a serem adquiridos e mantidos. Desafios adicionais incluem disponibilidade e dificuldades com testes em diferentes locais e ambientes.

©International Software Testing Qualifications Board

Laboratório remoto

Esses laboratórios são importantes e úteis para testar quando dispositivos ou redes não estão fisicamente disponíveis no site. O acesso remoto a dispositivos (RDA) permite o acesso por meio de uma conexão de rede a vários dispositivos hospedados no *data center* do provedor. Cada provedor de RDA em potencial precisa ser avaliado quanto à conformidade com os requisitos, especialmente para segurança.

Alguns laboratórios remotos fornecem os seguintes recursos adicionais:

- Versões de dispositivos físicos dedicados (p. ex.: o laboratório de dispositivos móveis).
- Dispositivos genéricos para um sistema operacional e versão específicos.
- Braços robóticos para realizar operações relacionadas a toque e gestos.
- Conexões de rede privada virtual (VPN) para dar acesso ao dispositivo.
- Conexões celulares com vários provedores de rede celular.
- Ferramentas e serviços de automação.

Alguns dos fatores que devem ser lembrados ao usar laboratórios de teste remotos incluem a capacidade de resposta lenta do dispositivo e opções limitadas de interação com esses dispositivos, como *multi-touch* e gestos. Isso pode ser econômico para uso esporádico, mas geralmente é mais caro se usado por longos períodos para uma ampla gama de dispositivos.

Outros fatores incluem disponibilidade da plataforma sob demanda em comparação à necessidade de obter acesso a dispositivos ausentes no laboratório local e a escalabilidade do laboratório, pois ele pode crescer e diminuir à medida que o projeto evolui.

Os cenários de teste que incluem sensores como NFC / Bluetooth ou consumo de bateria são difíceis de testar na nuvem. No entanto, as diferentes localizações geográficas dos laboratórios remotos podem ajudar nos testes que precisam de conexões de rede e GPS.

Um laboratório de testes pode utilizar um ou uma combinação das duas abordagens, dependendo do tipo de teste que precisa ser executado.

©International Software Testing Qualifications Board

5 Automação da execução dos testes [55 min]

Palavras-chave

teste baseado em dispositivo, relatório de teste, teste baseado em agente de usuário

Objetivos de aprendizagem

5.1 Abordagens de automação

MAT-5.1.1: (K2) Distinguir entre abordagens de automação comuns e estruturas para testes de aplicativos móveis.

5.2 Métodos de automação

MAT-5.2.1: (K2) Descrever vários métodos de automação para testar aplicativos móveis.

5.3 Avaliação das ferramentas de automação

MAT-5.3.1: (K1) Relembrar os vários parâmetros a serem considerados durante a avaliação de ferramentas de automação de testes móveis.

5.4 Abordagens para configuração de um laboratório de automação

MAT-5.4.1: (K2) Distinguir entre abordagens comuns de criação de laboratórios de teste com vantagens e desvantagens em relação à automação de testes.

©International Software Testing Qualifications Board

5.1 Abordagens de automação

Existem várias abordagens e estruturas de automação que podem ser usadas no teste de aplicativos para dispositivos móveis. A escolha da abordagem será parcialmente determinada pelo tipo de aplicação.

Duas abordagens comuns de automação de testes são:

- Teste baseado em agente de usuário.
- Teste baseado em dispositivos.

O teste baseado em agente do usuário utiliza a cadeia do identificador do agente do usuário enviada pelo navegador para imitar um determinado navegador em um determinado dispositivo. Essa abordagem pode ser usada para executar aplicativos da web em dispositivos móveis. O teste baseado em dispositivo, por outro lado, envolve a execução do aplicativo em teste diretamente no dispositivo. Essa abordagem pode ser usada para todos os tipos de aplicativos móveis.

O tipo de aplicativo também pode determinar a estrutura da automação de teste que seria adequada para esse aplicativo. A web para dispositivos móveis pode ser testada usando ferramentas comuns de automação de aplicativos da Web na área de trabalho, enquanto os aplicativos nativos podem precisar de ferramentas mais específicas. Os provedores de plataforma também podem fornecer ferramentas de automação dedicadas à sua plataforma.

Aa abordagens de automação usadas para aplicativos convencionais são também, frequentemente aplicáveis aos aplicativos móveis. Estes incluem testes de captura e reprodução, orientado por dados, orientado por palavras-chave, e comportamentais, conforme descrito nos syllabus [ISTQB_FL_2018] e [ISTQB_ALTAE_2016].

Normalmente, os principais recursos que uma estrutura de teste de aplicativo móvel deve incluir são:

- Identificação de objetos
- Operações de objetos
- Relatórios de teste
- Interfaces de programação de aplicativos e recursos extensíveis
- Documentação adequada
- Integrações com outras ferramentas
- Ser Independente das práticas de desenvolvimento de testes

5.2 Métodos de automação

Para desenvolver testes automatizados, o testador precisa entender o mecanismo de gravação ou criação dos scripts de automação e como acessar e interagir com os objetos gráficos do aplicativo, como botões, caixas de listagem e campos de entrada.

Existem vários métodos para identificar um objeto gráfico usado para a automação de teste. Estes incluem reconhecimento de imagem, reconhecimento de texto (OCR), e reconhecimento de objetos (web ou nativo, dependendo do tipo de aplicativo).

Um testador precisa não apenas praticar a detecção e identificação de objetos gráficos, mas também entender qual método de identificação de objetos será mais capaz de permitir que testes sejam bem-

©International Software Testing Qualifications Board

sucedidos e executados em uma grande variedade de dispositivos móveis, em paralelo e continuamente.

As principais diferenças entre os métodos de criação de scripts são:

Item de comparação	Identificação do objeto	Comparação imagem/OCR
Confiabilidade	Contanto que o identificador seja constante, o layout da tela pode ser alterado. O risco é que os objetos possam ser identificados e interagidos no código enquanto estão ocultos do usuário. Isso pode levar a resultados falsos-negativo.	As imagens podem ser dimensionadas de acordo com o tamanho da tela, mas os testes falharão assim que o layout for alterado.
Experiência do usuário	Normalmente, o script manual é necessário, pelo menos para melhorar os scripts gravados para facilitar a leitura e a capacidade de manutenção.	Teste completo baseado na interface do usuário (GUI) sem a necessidade de scripts.
Velocidade da execução	Tende a ser mais rápida que a comparação de imagem e, especialmente ao usar ferramentas nativas fornecidas pelo fabricante do sistema.	Tende a ser mais lento devido à necessidade de comparar pixel por pixel da tela com uma imagem de referência (*baseline*).
Manutenção	Depende da qualidade dos scripts de teste.	Principalmente no fornecimento de imagens de referência alteradas.
Desafio de criação	Ter o necessário conhecimento da linguagem de script e dos métodos de modelagem de software para criar uma solução de automação sustentável.	Geração de imagens de referência (baseline), especialmente quando o aplicativo é alterado frequentemente.

5.3 Avaliação das ferramentas de automação

Para ter sucesso na criação de soluções de automação de teste, as equipes de automação precisam escolher um conjunto de ferramentas apropriadas. Precisa ser considerado o entendimento das principais diferenças entre as ferramentas disponíveis e suas adequações aos requisitos do projeto (ver também [ISTQB_ALTAE_2016]).

Os parâmetros para avaliação de ferramentas de automação de teste podem ser divididos em duas categorias:

- de ajuste organizacional
- de ajuste técnico

Os parâmetros de ajuste organizacional são descritos no capítulo 6.2 do syllabus [ISTQB_FL_2018].

Os parâmetros de ajuste técnico incluem:

- Testes de automação de requisitos e complexidades, como o uso de novos recursos como identificação facial, impressão digital e *chatbots* pelo aplicativo.

©International Software Testing Qualifications Board

- Testes dos requisitos do ambiente, como diversas condições de rede, importações ou criações de dados de teste e virtualização do lado do servidor.
- Recursos de relatório de teste e realimentação.
- A capacidade do framework de gerenciar e conduzir a execução em grande escala localmente ou em um laboratório remoto.
- Integração do *framework* de teste com outras ferramentas utilizadas na organização.
- Suporte e disponibilidade de documentação para *upgrades* atuais e futuros.

5.4 Abordagens para configuração de um laboratório de automação

Ao realizar testes de aplicativos móveis, os desenvolvedores e testadores têm opções em torno do laboratório de teste de dispositivos que eles usariam para direcionar sua automação de teste.

- Laboratório local
- Laboratório remoto

Várias combinações dessas abordagens podem ser aplicadas. Suas principais características são descritas e comparadas no capítulo 4.4.

Os laboratórios de testes de dispositivos locais geralmente são difíceis e consomem tempo para serem mantidos. Ter dispositivos localmente em paralelo com emuladores e simuladores serviria melhor às fases iniciais de desenvolvimento e teste do aplicativo móvel.

Ao atingir um estágio mais avançado do desenvolvimento do aplicativo, as equipes precisam executar testes de regressão completos, testes funcionais e testes não funcionais. Esses testes são melhor executados em um laboratório de dispositivos completo. É onde um laboratório de testes de dispositivos remotos é gerenciado, atualizado continuamente e mantido na nuvem. Esses laboratórios de teste de dispositivos remotos complementam um laboratório de testes de dispositivos no local e garantem que combinações suficientes de dispositivos e sistemas operacionais estejam disponíveis e atualizados. Fazendo uso de laboratórios de teste de dispositivos remotos disponíveis, as equipes obtêm acesso a um conjunto maior de recursos suportados, incluindo relatórios de teste mais avançados e recursos avançados de automação de testes.

Por fim, ao executar em escala por meio de uma estrutura de automação de teste ou por meio de um trabalho de integração contínua (CI), a estabilidade geral do laboratório de teste é essencial para a eficiência e a confiabilidade do teste. Tais laboratórios são normalmente projetados para garantir que os dispositivos e sistemas operacionais estejam sempre disponíveis e estáveis.

Os laboratórios de teste de dispositivos remotos nem sempre são necessários nos estágios posteriores de desenvolvimento do aplicativo. Laboratórios locais bem projetados e mantidos podem ser tão bons ou até melhores, do que qualquer laboratório remoto.

©International Software Testing Qualifications Board

Referências

Documentos ISTQB®

[ISTQB_FL_2018] ISTQB® Certified Tester, Foundation Level Syllabus, 2018

[ISTQB_FLAT_2014] ISTQB® Certified Tester, Foundation Level Extension Syllabus, Agile Tester, 2014

[ISTQB_FLUT_2018] ISTQB® Certified Tester, (Foundation Level) Specialist Syllabus, Usability Testing, 2018

[ISTQB_FLPT_2018] ISTQB® Certified Tester, (Foundation Level) Specialist Syllabus, Performance Testing, 2018

[ISTQB_ALSEC_2016] ISTQB® Certified Tester, (Advanced Level) Specialist Syllabus, Security Testing, 2016

[ISTQB_ALTAE_2016] ISTQB® Certified Tester, (Advanced Level) Specialist Syllabus, Test Automation Engineer, 2016

[ISTQB_GLOSSARY] ISTQB®'s Glossary of Terms used in Software Testing, v.3.2

Literatura referenciada

[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

Livros e artigos

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

Links (websites)

Os links listados estavam acessíveis em 5 de janeiro de 2019, quando esse syllabus foi elaborado.

[URL1] http://gs.statcounter.com/

©International Software Testing Qualifications Board

[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-peopleabandon-mobile-apps-after-only-one-use/amp/

[URL5] https://www.google.com/accessibility/

[URL6] https://www.apple.com/uk/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

©International Software Testing Qualifications Board

Apêndice A - Objetivos de aprendizagem e níveis cognitivos

Os seguintes objetivos de aprendizagem são definidos como aplicáveis nesse syllabus. Cada tópico será examinado de acordo com o objetivo de aprendizagem associados a ele.

Nível 1: Lembrar (K1)

O candidato reconhecerá, lembrará um termo ou conceito.

Palavras-chave: identificar, lembrar, recuperar, lembrar, reconhecer, saber

Exemplos:

Pode reconhecer a definição de "falha" como:

- "Não entrega de serviço para um usuário final ou qualquer outro stakeholder" ou
- "Desvio do componente ou sistema de sua entrega, serviço ou resultado esperado"

Nível 2: Entender (K2)

O candidato pode selecionar as razões ou explicações para as declarações relacionadas ao tópico e pode resumir, comparar, classificar, categorizar e dar exemplos para o conceito de teste.

Palavras-chave: resumir, generalizar, abstrair, classificar, comparar, mapear, contrastar, exemplificar, interpretar, traduzir, representar, inferir, concluir, categorizar, construir modelos.

Exemplos:

Pode explicar o motivo pelo qual a análise e a modelagem do teste ocorrem o mais cedo possível:

- Para encontrar defeitos quando eles são mais baratos para remover.
- Para encontrar primeiro os defeitos mais importantes.

Pode explicar as semelhanças e diferenças entre integração e teste de sistema:

- *Semelhanças*: os objetos de teste para testes de integração e de sistema incluem mais de um componente, e os testes de integração e de sistema podem incluir tipos de teste não funcionais.
- *Diferenças*: o teste de integração concentra-se em interfaces e interações, e o teste do sistema concentra-se em aspectos do sistema completo, como o processamento de ponta-a-ponta

Nível 3: Aplicar (K3)

O candidato pode selecionar a aplicação correta de um conceito ou técnica e aplicá-lo a um determinado contexto.

Palavras-chave: implementar, executar, usar, seguir um procedimento, aplicar um procedimento

Exemplos:

- É possível identificar valores de limite para partições válidas e inválidas
- Pode selecionar casos de teste de um determinado diagrama de transição de estado para cobrir todas as transições.

©International Software Testing Qualifications Board

Referências

- 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

©International Software Testing Qualifications Board

Apêndice B - Glossário de termos específicos

2G: 2ª geração de tecnologia de telecomunicação móvel sem fio.

3d-touch: Veja "*Force Touch*"

3G: 3ª geração de tecnologia de telecomunicação móvel sem fio.

4G: 4ª geração de tecnologia de telecomunicação móvel sem fio.

5G: 5ª geração de tecnologia de telecomunicação móvel sem fio.

acesso a dispositivos remotos: interagir com um dispositivo fisicamente localizado em um local diferente do usuário, geralmente pela Internet.

Android Debug Bridge (ADB): ferramenta de linha de comando que permite a comunicação com um dispositivo.

Android Device Monitor (ADM): uma ferramenta autônoma que fornece uma interface de usuário para ferramentas de depuração e análise de aplicativos para Android.

Android Studio: O ambiente de desenvolvedor integrado oficial (IDE) para o Android. O Android Studio fornece ferramentas para criar aplicativos em todos os tipos de dispositivos Android.

aplicativo de plano de fundo: um aplicativo que está sendo executado em segundo plano.

aplicativo em primeiro plano: um aplicativo que é executado no primeiro plano do dispositivo para interação direta do usuário.

aplicativo freemium: um modelo de negócios no qual os usuários não pagam nada para fazer o download do aplicativo e recebem ofertas opcionais no aplicativo.

aplicativo nativo: um aplicativo especialmente desenvolvido para uma determinada plataforma, geralmente usando APIs de plataforma e ferramentas de desenvolvimento fornecidas pela plataforma.

aplicativo pago: um aplicativo monetizado com a venda em lojas de aplicativos.

aplicativo pré-instalado: um aplicativo móvel instalado pelo fabricante do dispositivo. Normalmente, o usuário não pode desinstalar esses aplicativos.

aplicativos baseados em propagandas: um modelo de monetização de aplicativos em que as organizações de desenvolvimento ganham dinheiro com anúncios exibidos no aplicativo.

aplicativos baseados em transação: um aplicativo no qual o usuário paga por transação.

aplicativos multiplataforma: aplicativos desenvolvidos e desenvolvidos para rodar em múltiplas plataformas usando a mesma base de código para todas as plataformas.

arquivo simples: um arquivo sem hierarquia interna.

atalho do aplicativo: um atalho para um conjunto específico de ações definidas em um aplicativo por desenvolvedores de aplicativos no Android 7.1 ou superior.

barcode: uma representação óptica de dados legível por máquina (código de barras).

biblioteca: uma coleção de recursos não voláteis usados por programas de computador, ou seja, funções do aplicativo.

©International Software Testing Qualifications Board

bluetooth: uma tecnologia de comunicação sem fio de curto alcance.

bytecode: um conjunto de instruções projetado para execução eficiente por um interpretador de software. Também chamado de código portátil ou código-p.

camada única: uma abordagem de design de aplicativo de *backend* na qual um único servidor fornece todos os serviços necessários para um aplicativo.

célula GSM: parte de uma rede GSM que pode ser identificada por seu ID de célula exclusivo

compatibilidade com versões anteriores: a capacidade de um aplicativo funcionar em versões anteriores de plataformas.

compra no aplicativo: conteúdo extra e recursos disponíveis diretamente de um aplicativo.

comunicação assíncrona: um tipo de comunicação em que os dados podem ser transmitidos de forma intermitente, e não em um fluxo contínuo.

comunicação síncrona: Um método de transferência de dados no qual os fluxos de dados são enviados (a montante) e recebidos (a jusante) à mesma velocidade e são espaçados por sinais de temporização.

conflito de upload: um erro ao tentar enviar um arquivo que já está presente no destino do upload.

dados confidenciais: dados que precisam de proteção especial, como senhas e dados pessoais.

dados de celulares: dados transferidos através de uma rede celular.

dispositivo complementar: um dispositivo de computador projetado para funcionar em cooperação com um dispositivo inteligente dependente.

dpi / ppi: acrônimo de pontos por polegada / pixels por polegada. Um número que expressa a densidade de uma exibição, seja em pontos ou pixels.

emulador: um aplicativo de software que imita o comportamento do hardware.

emulador móvel: representação virtual de uma plataforma de hardware. Por exemplo, o emulador do Android é um hardware virtual que executa uma imagem real do sistema operacional Android. A mesma imagem do SO se for implantada em um hardware ela funcionará.

espaço móvel: um termo de encapsulamento que inclui qualquer coisa em relação à tecnologia de dispositivos móveis, do mercado e seus players aos dispositivos e aplicativos.

estado de energia: um perfil definido ou predefinido pelo usuário.

estouro: uma situação em que os dados recebidos excedem o que pode ser acomodado.

force-touch: uma tecnologia desenvolvida pela Apple Inc. que permite que *trackpads* e telas sensíveis ao toque distingam entre diferentes quantidades de força aplicadas em suas superfícies.

fragmentação de dispositivo: a diversidade de dispositivos disponíveis, sua configuração de hardware exclusiva e os seus impactos nos aplicativos e na experiência do usuário.

framework de desenvolvimento multiplataforma: Um framework para desenvolver um aplicativo para várias plataformas usando a mesma base de código.

frequência da CPU: a taxa de clock do processador.

©International Software Testing Qualifications Board

globalização: veja internacionalização.

Global Positioning System (GPS): uma rede de satélites envia sinais de tempo. Ao incluir o sinal de pelo menos 3 satélites, um receptor pode calcular sua posição relativa para os satélites via triangulação.

Global System Mobile (GSM): originalmente *Groupe Spécial Mobile.* É um padrão desenvolvido pelo Instituto Europeu de Padrões de Telecomunicações (ETSI) para descrever os protocolos para redes celulares digitais de segunda geração usadas por dispositivos móveis. Atualmente, o padrão mais comum para comunicação móvel no mundo.

I18N: palavra baseada em números para internacionalização.

IDE: um aplicativo de software que fornece recursos abrangentes para programadores de computador para desenvolvimento de software.

integridade dos dados: a precisão e a consistência dos dados ao longo de todo o seu ciclo de vida, incluindo armazenamento, processamento e recuperação.

internacionalização: o processo de preparação de uma aplicação para acomodar várias versões localizadas.

interrupção: um evento que ocorre durante outro evento.

IoT: Internet das Coisas. Um dispositivo ou, por exemplo, um sensor de interesse conectado à internet.

L10N: código utilizado para localização.

loja de aplicativos: uma plataforma de distribuição de aplicativos na qual os desenvolvedores podem carregar seus aplicativos e os usuários podem procurar aplicativos para baixar e instalá-los em sua plataforma.

memória externa: uma memória adicional que é adicionada ao dispositivo por meio de uma interface padrão. Atualmente, os cartões SD são mais comuns para telefones celulares.

memória interna: uma memória incluída no hardware do dispositivo.

mercado de terceiros: uma plataforma de distribuição de aplicativos não operada por um provedor de plataforma.

mnemônico: um auxiliar de memória.

modo avião: Um modo de operação especial para dispositivos móveis em que os transmissores de rádio são desativados para evitar interferência nos sistemas de operação / comunicação de vôo.

modo bloqueio ou não perturbe: um modo operacional de dispositivos móveis que podem ser ativados pelo usuário para suprimir certos recursos - notificações comuns e chamadas de voz.

modo de economia de energia: um modo operacional de dispositivos móveis que pode ser ativado pelo usuário ou pelo próprio dispositivo para economizar energia.

modo paisagem: a orientação do dispositivo na qual a largura da tela é maior que a altura.

modo retrato: uma orientação de dispositivo na qual a altura da tela é maior que a largura.

multicamadas: uma abordagem de desenho de aplicativo de *backend* em que 2 ou mais servidores fornecem funcionalidades especializadas.

©International Software Testing Qualifications Board

multi-touch: um tipo de interação com um dispositivo usando vários eventos de toque em paralelo.

Near Field Communication (NFC): uma tecnologia de comunicação de rádio de curto alcance.

OCR: reconhecimento óptico de caracteres. O reconhecimento de imagens de texto contidas em uma imagem eletrônica e sua conversão em texto codificado por máquina.

orientação: a colocação de um objeto no celular comumente usados para expressar a maneira como o dispositivo é usado. Pode ser paisagem ou retrato.

plataforma móvel: um ecossistema em torno de um sistema operacional móvel, geralmente incluindo ferramentas de desenvolvimento, o próprio sistema operacional e um canal de distribuição de aplicativos.

ponto cego (ou morto): um local sem rede de telecomunicação sem fio.

preferências: o dispositivo geral ou os parâmetros de configuração do aplicativo que podem ser alterados pelo usuário.

proporção: a relação de largura para altura de uma exibição ou imagem.

QRcode: o código QR (abreviado de Quick Response Code) é a marca comercial de um tipo de código de barras matricial (ou código de barras bidimensional).

regionalização: o processo de ajustar um aplicativo ou produto a uma determinada região por meio de ações como ajustes de tradução e formatação.

rede celular: uma rede celular é uma rede criada de várias células independentes, mas conectadas.

Rede Privada Virtual (VPN): um canal privado criptografado através de uma rede pública.

Retenção de dados do aplicativo: os dados gerados pelo usuário e / ou o conteúdo do aplicativo é retido no dispositivo quando o aplicativo for desinstalado para ser acessado por outros aplicativos ou quando o mesmo aplicativo é reinstalado.

rooting: O processo de obtenção de acesso root ao sistema operacional do dispositivo. Termo é geralmente utilizado na plataforma Android. Semelhante ao *jailbreaking* no iOS.

SDK: um conjunto de ferramentas e bibliotecas para desenvolver softwares para uma determinada plataforma.

sensor: um dispositivo, módulo ou subsistema cuja finalidade é detectar eventos ou alterações em seu ambiente e enviar as informações para outros componentes eletrônicos, geralmente um processador de computador.

side-loading: o carregamento / instalação de uma aplicação através de um meio que não seja uma loja de aplicativos.

simulador móvel: um ambiente de tempo de execução virtual. Por exemplo, o simulador do iOS finge ser iOS, mas na verdade não é um iOS real.

sincronização de dados: o processo de colocar dados no mesmo estado em duas ou mais origens.

sistema backend: um sistema de servidor que fornece funcionalidade para outros sistemas.

tamanho da porta de visualização: um tamanho de tela virtual usado pelo navegador para ajustar o layout à tela.

©International Software Testing Qualifications Board

validação de dados: a avaliação se os dados estão corretos, precisos, consistentes e úteis.

Xcode: um ambiente de desenvolvimento integrado fornecido pela Apple para desenvolver aplicativos OSX e iOS.

©International Software Testing Qualifications Board