



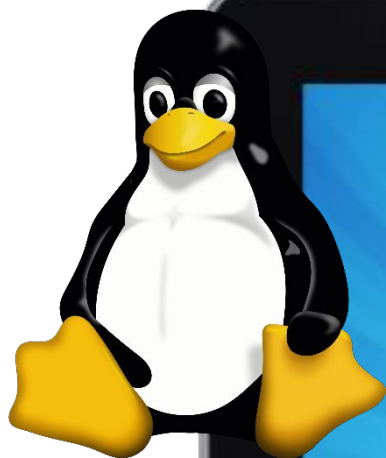
Sejam bem-vindos!

DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Prof. Jarbas Araújo



Revisão para Avaliação Presencial



INTRODUÇÃO AO DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Todavia, assim como ocorre com os computadores, um sistema operacional mobile possui características próprias, como Application Programming Interface (API), bibliotecas e um padrão para arquivos binários.

Quando uma empresa lança o operating system (OS), precisa também de um conjunto de ferramentas para que os desenvolvedores possam criar aplicações, gerando o ecossistema ao redor da plataforma.

Assim, um bom sistema operacional deve ter bons aplicativos para que os usuários tenham suas necessidades supridas



App Store



Desenvolvimento híbrido

O desenvolvimento híbrido é uma espécie de fusão entre uma aplicação Web e um aplicativo mobile nativo, que são desenvolvidos com tecnologia Web (basicamente Hypertext Markup Language [HTML], Cascading Style Sheets [CSS] e JavaScript ou outras linguagens que, posteriormente, resultam nessa tríade), suportada pelos navegadores. Portanto, é muito mais simples e barato desenvolver aplicações mobile dessa forma.

Progressive Web App

Paralelamente ao desenvolvimento de aplicativos nativos utilizando tecnologia híbrida, há a introdução dos Progressive Web Apps (PWA), que são uma aplicação Web que implementa certas especificações técnicas e, em versões mais atuais, sobretudo da plataforma Android, executa em um browser oculto.

Todavia, essa aplicação possui um ícone para acesso e armazena localmente (no dispositivo) imagens, fontes e conteúdo estático para poder ser acessado off-line, inclusive mantendo em cache os dados dinâmicos (MAJCHRZAK; BIØRN-HANSEN; GRØNLI, 2018).

Um framework Web muito conhecido e capaz de gerar PWA é o Quasar, baseado no Vue.js, com uma curva de aprendizado bastante suave e que pode servir bem aos aplicativos que não demandam muitos recursos.

Conceitos fundamentais

Ao lidar com o desenvolvimento para dispositivos móveis, deve-se considerar que o cenário não é o mesmo de quando se desenvolve aplicações desktop ou Web, com foco em acessos por meio de computadores.

Por isso, apesar de os smartphones modernos possuírem uma quantidade razoável de memória e processadores relativamente potentes, é necessário pensar na limitação de recursos dessas plataformas em comparação aos computadores.

Portanto, é necessário entender a fragmentação do mercado de dispositivos, começando pelas telas. Os computadores podem usar uma gama extensa de tamanhos de monitores, mas as resoluções estarão, em sua maioria, entre 1280 x 800, Full High Definition (Full HD) (1920 x 1080) e, mais recentemente, 4K; já os aparelhos móveis têm uma variação muito maior na resolução disponível, bem como no tamanho das telas (Figura 3). Apenas isso já demanda cuidado, que deve ser considerado em tempo de modelagem e projeto da aplicação.

INTRODUÇÃO AO DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

Existem no mercado diversos sistemas operacionais para dispositivos móveis, se destacando o sistema operacional iOS, presente nos produtos da fabricante Apple, e o democrático Android, desenvolvido pela Google, sendo de código aberto e presente em dispositivos móveis de incontáveis fabricantes.

Dispositivos e sistemas operacionais móveis

Segundo Lee, Schneider e Schell (2005), a mobilidade pode ser definida como a capacidade de poder se deslocar ou ser deslocado de forma fácil.

Em computação móvel, mobilidade se refere ao uso de dispositivos portáteis que oferecem a capacidade de realizar, com facilidade, um conjunto de funções de aplicação, como conexão, obtenção e fornecimento de dados a outros usuários e uso de aplicações e sistemas. Dessa forma, podemos classificar como dispositivos móveis:

- ☐ telefones celulares;
- ☐ laptop;
- ☐ smartphones;
- ☐ tablets;
- ☐ robôs

Tipos de sistemas operacionais móveis

- ☐ Android: desenvolvido pela Google, utilizado por diversos fabricantes.
- ☐ iOS: criado pela Apple, exclusivo para os seus dispositivos.
- ☐ Symbian OS: desenvolvido pela Nokia e utilizado por diversos fabricantes, mas foi descontinuado.
- ☐ Windows Phone: criado pela Microsoft, chegou a ser muito utilizado, mas não resistiu à concorrência e recentemente foi descontinuado.

O MERCADO *MOBILE* E SEUS SEGMENTOS

O mercado de *smartphones* apresenta, de um lado uma variedade de dispositivos disponíveis mas, por outro, uma concentração em torno de dois sistemas operacionais prevalentes. A Apple iOS e a Google Android dominam praticamente 100% do mercado em *smartphones* e *tablets*.



Wearables

Smartphone

Tablet



Já os wearables são dispositivos móveis vestíveis, usados junto ao corpo do usuário. Seu conceito não é novo, pois existem projetos de relógios inteligentes há algumas décadas, desde o Hewlett-Packard (HP-01) (no final dos anos de 1970, que continha funções de calculadora, calendário, cronômetro, timer e um teclado com 28 teclas minúsculas); passando pelo relógio da International Business Machines (IBM), baseado em Linux (nos anos 2000, com duração da bateria de apenas seis horas), até o lançamento das linhas Galaxy Watch e Apple Watch.

Nestas últimas, o conceito começou a ganhar popularidade, apesar do valor elevado. A maioria dos problemas dos wearables envolve a duração da bateria, tentar manter um padrão estético e concentrar toda a tecnologia em um diminuto espaço, para que seu uso não se torne incômodo. Contudo, grande parte desses problemas tem sido contornada com tecnologias mais atuais e, hoje, principalmente os smartwatches (Figura 3) e smartbands (as versões simplificadas, em forma de pulseira, geralmente voltadas ao monitoramento de atividades físicas) elevam o crescimento desse tipo de tecnologia.

PRINCIPAIS SISTEMAS OPERACIONAIS

Apple iOS

Dos sistemas em uso atualmente, este foi o primeiro a ser lançado, em 2007, junto com o iPhone. Apresenta uma fatia de mercado que varia de 15% (segundo a IDC) a 28% (segundo a Net Marketshare). É desenvolvido com exclusividade pela Apple para seus *smartphones* e *tablets*.



Android

Sistema desenvolvido pela Google para competir com o iOS e outros sistemas da Apple. Atualmente é o mais utilizado. Sua fatia em *smartphones* chega a 85% de acordo com a IDC. É licenciado para diversos fabricantes de todo o mundo.

O Android e a fragmentação

Em levantamento de 2019, apenas 10% dos usuários de Android estavam utilizando a última versão. 12% ainda utilizavam a versão Lollipop, lançada em 2014. Entre essas duas versões, a 7.1 é a única que não alcança 10% da base de usuários, ou seja, existe uma fragmentação da base dentro do próprio sistema.



Ambientes integrados de desenvolvimento

Ambientes integrados de desenvolvimentos, ou IDEs são softwares que facilitam o desenvolvimento de sistemas. Eles são compostos por editor de código fonte, compilador, depurador e outras funcionalidades. Quando se trata do desenvolvimento para dispositivos móveis, os mais utilizados são Android Studio, Visual Studio Code, IntelliJ, Atom e Xcode. Um desenvolvedor iniciante, deve começar escolhendo apenas um, portanto, não é necessário instalar todos.

Cada aplicação possui requisitos específicos em relação a hardware e software, para que possa ser executada. Consulte o site do desenvolvedor para saber se seu computador atende aos requisitos, pois o não atendimento pode impactar em perda de desempenho ou em não conseguir executar a aplicação.

Ambientes de desenvolvimento de aplicativos móveis

Android Studio — plataforma desenvolvida pela Google com ferramentas para facilitar o desenvolvimento de aplicativos móveis que rodam o sistema Android. Disponibiliza um editor de código nativo e ferramentas para emulação e análise das aplicações. É um dos ambientes mais utilizados para desenvolvimentos voltados para Android (DEITEL; DEITEL; WALD, 2016). Roda em ambientes Windows, Mac, Linux e Chrome OS. Suporta o desenvolvimento de aplicativos em Java, Kotlin e C/C++.

Outras IDEs também bastante utilizadas incluem o NetBeans e Eclipse.

Além dessas, se você tem bastante conhecimento de programação, sabe que é possível desenvolver aplicações utilizando editores de texto simples, embora seja mais complexo para identificar problemas no código.

Ambientes de desenvolvimento de aplicativos móveis

Quando você instala uma IDE, já pode escolher em quais linguagens vai programar e, se necessário, adicionar os respectivos plug-ins. Porém, em algumas linguagens, como Java e Python, você pode instalar o SDK separadamente.

No Quadro 1, você pode observar uma correlação entre as principais linguagens de programação e as IDEs que suportam seu desenvolvimento. Note que o Atom e o Code estão entre as IDEs que suportam a maior quantidade de linguagens

Quadro 1. Correlação entre IDEs e linguagens de programação suportadas para o desenvolvimento de aplicativos móveis

IDE linguagem	Android Studio	Visual Studio Code	IntelliJ	Atom IDE	Xcode
Java	X	X	X	X	
JavaScript		X	X	X	
Kotlin	X		X	X	
TypeScript		X	X	X	
Objective-C		X		X	X
Swift		X		X	X

Plataformas de desenvolvimento: IDE e emulador Android

Para uma boa experiência com o Android Studio, você deve estar atento aos requerimentos mínimos de sistema. No caso da plataforma Windows, os requisitos são os listados abaixo.

- ☐ Microsoft Windows 7/8/10 32 ou 64 bits.
- ☐ 3 GB de memória RAM mínima ou 8 GB de memória RAM recomendado.
- ☐ 2 GB de espaço mínimo disponível em disco ou 4GB de espaço disponível em disco recomendados.
- ☐ Resolução mínima da tela de 1280 × 800 pixels.

A maioria dos IDEs no mercado atual utiliza o SDK da Android como base para suas aplicações, fornecendo uma ferramenta de criação, gerenciamento e depuração do projeto com base nas funcionalidades fornecidas pelo SDK. O Eclipse e o Netbeans são exemplos de IDEs que utilizam o SDK da Android como base, integrando-o ao plugin ADT (Android development tools). Logo, não existe uma diferença clara sobre qual dos dois IDEs seria o mais indicado, pois esta é uma questão relacionada ao desenvolvedor.

PUBLICANDO APLICAÇÕES PARA ANDROID E IOS

A maior parte das aplicações desenvolvidas têm como finalidade servir a algum propósito em uma base de usuários. Em se tratando de aplicações mobile, as app stores como a Google Play ou a Apple App Store quase sempre são as melhores opções para a distribuição dessas aplicações, sejam elas gratuitas ou pagas.



App Store



Google play

Ao publicar sua aplicação, é necessário seguir um conjunto de normativas, leis e boas práticas. Cada loja de aplicativos apresenta suas próprias normas, assim como cada país tem suas próprias leis - que devem ser seguidas para que a aplicação seja aceita e publicada.

A publicação de um aplicativo nas lojas oficiais é a melhor forma para sua distribuição, pois elas contam com maior confiança da base de usuários e facilidade de instalação.



Conformidade

Crie seu projeto em conformidade com as normas e políticas de ambas as lojas oficiais desde o início – o que evitará retrabalho na fase de lançamento e consequente economia de tempo e esforço.

Fique de olho nas mudanças

As normas de conformidade das *app stores* não são estáticas. Elas sofrem atualizações e, por isso, o responsável pelo projeto deve revisá-las com frequência.



Legislação

Assim como a aplicação deve estar em conformidade com as políticas das *app stores*, também deve respeitar a legislação dos países e estados onde será disponibilizada. Algumas áreas dispõem de regulamentação específica, que deve ser de conhecimento do responsável pelo projeto.

Documentação

As *app stores* solicitam que a aplicação seja documentada de acordo com seus requisitos. A falta de documentação é um dos motivos de recusa da publicação.

PONTOS DE ATENÇÃO

Testes da Apple

A Apple realiza testes individualmente em cada aplicativo lançado em sua plataforma, inclusive em relação a desempenho e responsividade.

Aplicações para a saúde

As normas para publicações de aplicativos com foco na saúde são, em geral, mais rígidas, pois podem por usuários em risco.

Crianças

As aplicações que tenham como público-alvo crianças em geral tem regras rígidas sobre o conteúdo e o formato da aplicação, até mesmo sobre os anúncios veiculados.



Jogos

De acordo com a Apple, os jogos publicados não devem ter como inimigo determinada classe de pessoas, como raça, cultura, governos legítimos, empresas ou outras entidades reais. O Google, por sua vez, enfatiza suas regras mas em relação aos jogos de azar, que não são proibidos, porém, devem seguir algumas regras específicas, por exemplo, impedir o acesso de menores de idade, restringir o acesso aos países com lei que não permite tal conteúdo, bem como ter download gratuito e classificação para adultos. Para o Google, os jogos do tipo fantasy sports por rodada estão sujeitos aos requisitos específicos.

Drogas, medicamentos e aplicações médicas

As lojas são categóricas quanto ao estímulo, venda e promoção de produtos ilícitos, bem como ao incentivo ao consumo de tabaco e álcool. Em relação aos dois últimos itens, o Google mantém seu foco em aplicações voltadas aos menores de idade ou acessíveis por eles. A Apple é mais enfática no tocante ao fumo em geral e ao álcool em excesso.

Nas duas plataformas, é proibido facilitar a venda de medicamentos sem receitas médicas. Para a Apple, aplicações médicas são submetidas à análise minuciosa, inclusive da metodologia utilizada na mensuração de parâmetros físicos de saúde, bem como devem sugerir a consulta a um profissional da área. Por exemplo, aplicações que visem a dosagem de medicamentos podem ser lançadas apenas pelo laboratório responsável por eles, pelo hospital, pela universidade, seguradora de saúde, farmácia ou outra entidade aprovada pelas entidades regulamentadoras de cada país — no Brasil, a Agência Nacional de Vigilância Sanitária (Anvisa), e nos Estados Unidos, a Food and Drug Administration (FDA). Já o Google relata algumas substâncias que, independentemente da declaração de legalidade, serão recusadas se sua venda, anúncio ou promoção forem encontrados no aplicativo.



Privacidade e requisitos técnicos

A manipulação de dados dos usuários deve ser clara e transparente, explicitando o uso que se fará deles. A coleta, o uso e o compartilhamento de dados devem estar descritos na política de privacidade e facilmente acessíveis ao indivíduo, que pode não concordar e não utilizar a aplicação.

Aplicativos com foco no público infantil

As app stores da Apple e do Google são categóricas na proteção infantil. Portanto, qualquer aplicação que almeje alcançar esse público deve seguir normas rígidas de conduta sobre conteúdo, por exemplo, ausência de menções de natureza sexual, incluindo anúncios, drogas, álcool e tabaco (estes são somente permitidos em situações educacionais focadas em sua prevenção).

Google Play Store

Para a publicação de aplicações na Play Store, deve-se vinculá-las a uma conta do Google e fazer o registro de desenvolvedor no Google Play Console, um console de gerenciamento das aplicações publicadas na loja. Esse registro é cobrado somente uma vez e, em 2019, custa 25 dólares, pagos por meio de cartão de crédito internacional.

Ao criar a conta, você estará apto a publicar suas aplicações, mas não a vendê-las, ou sequer os produtos in-app, e, para que isso seja possível, precisa-se de uma conta de comerciante, o Google Wallet (DEITEL; DEITEL; DEITEL, 2015). Alguns países não permitem que o usuário possua esse tipo de conta, contudo, não é o caso do Brasil, inclusive, as aplicações vendidas no país têm o Real (R\$) como base e moeda, além de concordância com o sistema tributário nacional. Após a criação da conta de desenvolvedor, você pode vinculá-la à conta de comerciante por meio do console de desenvolvedor do Google Play



Google play

Apple App Store

A publicação na Apple App Store é um pouco mais trabalhosa; e seu processo, mais complexo. Mesmo que sua aplicação tenha sido desenvolvida utilizando um ambiente multiplataforma, como o Ionic, deve-se partir de um Macintosh com o Xcode.

Outro fator a ser considerado, desde a concepção do projeto, é de que a Apple realiza testes de qualidade, segurança e performance nas aplicações submetidas à sua loja. Assim, deve-se desenvolver a aplicação de uma forma que não prejudique sua performance e executar testes para verificar bugs antes de enviá-la.

Para efetuar a publicação, é necessário aderir ao Developer Program, que custa 99 dólares por ano para os desenvolvedores individuais, diferentemente do Google Play, em que a inscrição é paga apenas uma vez. Deve-se, ainda, fazer o requerimento e aguardar o código de ativação enviado pela companhia.

Uma vez liberada a conta, você terá acesso às ferramentas de testes, ao Apple Pay e iCloud, sendo que o desenvolvedor usará uma ferramenta chamada iTunes Connect para cadastrar e publicar a aplicação. Todo lançamento, atualizações e revisões devem passar por um processo de aprovação, que pode demorar alguns dias — período em que serão utilizados para os testes feitos pela Apple.

Outra diferença no processo é que a Apple, em contas de pessoas jurídicas, exige que a empresa tenha o registro Data Universal Numbering System (DUNS) number, um número de identificação internacional das empresas.



App Store

Projeto de aplicativo via mockups

Discutir a ideia inicial para o desenvolvimento de um aplicativo é o primeiro passo para o entendimento do escopo de um projeto de software.

A partir dessa conversa inicial, é interessante fazer um rascunho das telas do aplicativo, de modo a validar se o que se entende é de fato o que se espera do software.

A elaboração de mockups ajuda nessa fase inicial de alinhamento entre o desenvolvedor (ou equipe de desenvolvimento) e quem está demandando o software.

Mockups fazem bem esse papel de rascunho. Eles podem ser elaborados como desenhos, usando papel e caneta, ou ter interfaces com alguma funcionalidade, sendo elaborados em softwares específicos.



O desenvolvimento de mockups também é conhecido como sinônimo de prototipação.

Sendo assim, para desenvolver um mockup, você pode se basear em algum processo cíclico para prototipação, pois isso facilita o entendimento e a melhoria contínua do seu modelo.

Tipos de leiaute RelativeLayout e LinearLayout

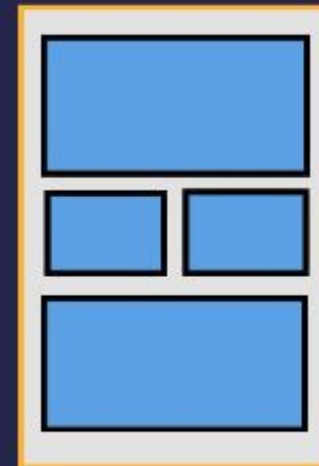
RELATIVELAYOUT APLICADO NO ANDROID

Layouts são muito importantes no Android, pois permitem que a interação dos usuários com os Apps seja a melhor possível.

RelativeLayout

Como o próprio nome sugere, o RelativeLayout mostra **a posição dos componentes em relação uns aos outros**. A posição pode ser especificada em relação a elementos consecutivos ou ao componente pai.

O RelativeLayout **é o layout mais flexível fornecido pelo Android**, pois permite posicionar elementos na tela. Por padrão, definem-se todos os componentes no canto superior esquerdo do *layout*.



Tipos de leiaute RelativeLayout e LinearLayout

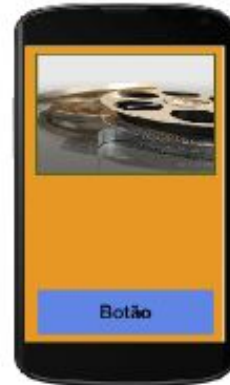
Exemplo

Abaixo veja os atributos do RelativeLayout:

Id: Definição identificação Layout

Gravity: Define a posição x-y de um componente na tela.

IgnoreGravity: Este pode ser adicionado para ignorar a gravidade de um componente.



Em relação ao layout pai



Em relação a outras views

Disposição

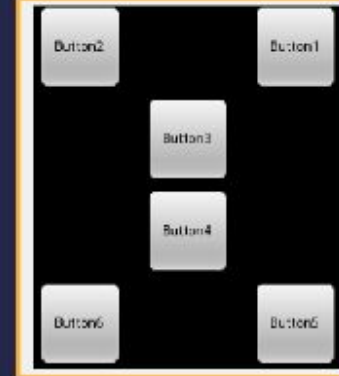
Apresenta quatro diferentes métodos construtores:

RelativeLayout
(Context context)

RelativeLayout
(Context context, AttributeSet attrs)

RelativeLayout
(Context context, AttributeSet attrs, int defStyleAttr)

RelativeLayout
(Context context, AttributeSet attrs, int defStyleAttr, int defStyleRes)



Métodos

setGravity(): Define a gravidade dos filhos como centralizado, esquerda ou direita.

setHorizontalGravity(): Usado para posicionar componentes na horizontal.

setVerticalGravity(): Usado para posicionar componentes na vertical.

requestLayout(): Usado para requisitar *layout*.

setIgnoreGravity(): Usado para ignorar a gravidade de um determinado componente.

getGravity(): Usado para obter a posição de componentes.

getAccessibilityClassName(): Retorna a *ClassName* do objeto.



Vantagem do RelativeLayout

Tipo de *layout* poderoso para desenvolver **interfaces de usuário iterativas**.

- ☐ **Elimina o aninhamento** de ViewGroups.
- ☐ Mantém a **hierarquia de layouts plana**.

Nos casos em que houver a necessidade de utilizar vários grupos de LinearLayout aninhados, deve ser avaliada a troca para o RelativeLayout.

Tipos de leiaute RelativeLayout e LinearLayout

Na plataforma Android, o tipo de leiaute que possibilita posicionar os componentes entre si é chamado **RelativeLayout**. Por exemplo, temos em uma tela um componente TextView e desejamos posicionar um componente Button à sua esquerda; no arquivo XML, temos a possibilidade de fazer referência ao componente TextView para que o componente Button seja posicionado em determinado lugar tendo como atribuição a posição do componente TextView. Na Figura 2, podemos perceber os componentes devidamente alinhados graças ao recurso do RelativeLayout.



Figura 2. RelativeLayout.

Fonte: Romanato (2016, documento on-line).

Tipos de leiaute RelativeLayout e LinearLayout

LinearLayout

Um dos tipos de leiaute mais utilizados quando desenvolvemos aplicativos para o Android, possibilita que os componentes sejam organizados tanto em posição vertical quanto horizontal, alinhando todos os componentes em uma única direção, conforme especificado.

Todos os componentes de um LinearLayout (Figura 3) são colocados um após o outro; logo, uma lista vertical terá somente um componente por linha, independentemente da dimensão de sua largura, e uma lista horizontal terá a altura de apenas uma linha (a altura do componente mais alto). O LinearLayout respeita as margens entre os componentes e o seu alinhamento (direita, centro ou esquerda).



Figura 3. LinearLayout.

Fonte: Romanato (2016, documento on-line).

Modelagem e projeto de aplicativos móveis

Para desenvolver um software, é essencial que o projeto passe pelas técnicas da engenharia de software, especialmente para sua modelagem e construção. No caso do processo de desenvolvimento de aplicativos móveis, essa modelagem é essencial para o bom êxito do produto.

A modelagem de um projeto para aplicativo móvel passa por uma análise que abarca desde a análise de requisitos até implementação e teste, determinando padrões que viabilizarão um projeto claro e que atenda às necessidades do usuário.

Modelagem e projeto de aplicativos móveis

Modelagem de requisitos para aplicativos móveis

A, a análise de requisitos possibilita a especificação da função e de como se apresenta o desempenho do software, bem como da interface do software com outros elementos do sistema, estabelecendo quais são as restrições de projeto que o software deve enfrentar (ESTEVEZ, 2006).

De acordo com Pressman e Maxim (2016),

“Hoje muitos aplicativos podem ter sido desenvolvidos por amadores. Porém, para os engenheiros de software, os aplicativos envolvem vários desafios relativos ao seu sistema e à sua construção, sobretudo pelo fato de que os aplicativos, para serem executados em um dispositivo móvel, precisam seguir especificidades próprias.”

Modelagem e projeto de aplicativos móveis

Para Pressman e Maxin (2016), a modelagem de requisitos para aplicativos móveis depende de alguns fatores como:

- ★ o tamanho e a complexidade do incremento da aplicação;
- ★ o número de envolvidos;
- ★ o tamanho da equipe de desenvolvimento do aplicativo;
- ★ o tempo de trabalho desenvolvido pela equipe junta;
- ★ até que ponto o sucesso da organização depende de forma direta do sucesso da aplicação.

Modelagem e projeto de aplicativos móveis

Na modelagem de requisitos para aplicativos móveis, pode-se aplicar o modelo de **desenvolvimento ágil**, caracterizado por uma atividade de comunicação pela qual você poderá identificar os envolvidos e as categorias de usuário, o contexto de negócio, as informações e as metas de aplicação, os cenários e os requisitos de aplicação de uso, sendo as informações entradas para definição da modelagem de requisitos (PRESSMAN; MAXIM, 2016).

A respeito da análise de requisitos dessa modelagem, Pressman e Maxim (2016, p. 214) afirmam que:

“..A análise captura essas informações, estrutura-as usando um esquema de representação formalmente definido (onde apropriado) e depois produz como saída modelos mais rigorosos. O modelo de requisitos fornece uma indicação detalhada da verdadeira estrutura do problema e dá uma visão da forma de solução ...”

Modelagem e projeto de aplicativos móveis

Pressman e Maxim (2016) definem cinco classes principais de modelos para os aplicativos móveis. Veja-as a seguir:

Modelo de conteúdo. Identifica o indício completo do conteúdo (texto, gráficos, imagens, áudio e vídeo) que será fornecido pela aplicação. De forma geral, um modelo de conteúdo se refere a qualquer informação apresentada ao usuário de forma coesa.

Exemplo: A árvore de dados foi desenvolvida para um componente desse sistema, que vai apresentar as informações de forma hierárquica. Esse modelo de conteúdo é responsável por descrever o objeto de conteúdo.

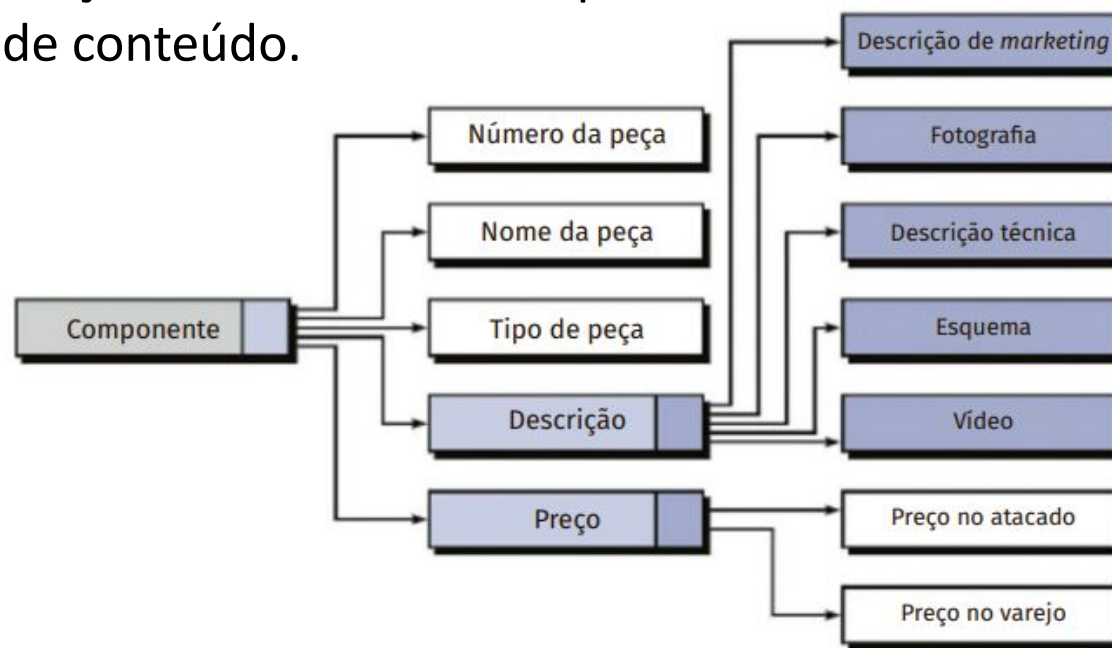


Figura 1. Árvore de dados para um componente de um sistema de venda.

Fonte: Adaptada de Pressman e Maxim (2016).

Modelagem e projeto de aplicativos móveis

Modelo de interações. É possível que você encontre interação em praticamente todos os aplicativos que usar. A interação é um fator importante para que haja o engajamento do usuário com o que está sendo exposto pelo conteúdo do aplicativo e com a forma como ele funciona, ou seja, como o aplicativo se comporta.

Para Pressman e Maxim (2016), um modelo de interação precisa ser composto de:

- casos de uso;
- diagrama de sequência (modelado conforme UML — Unified Modeling Language);
- diagrama de estado (modelado conforme UML);
- protótipos de interface do usuário.

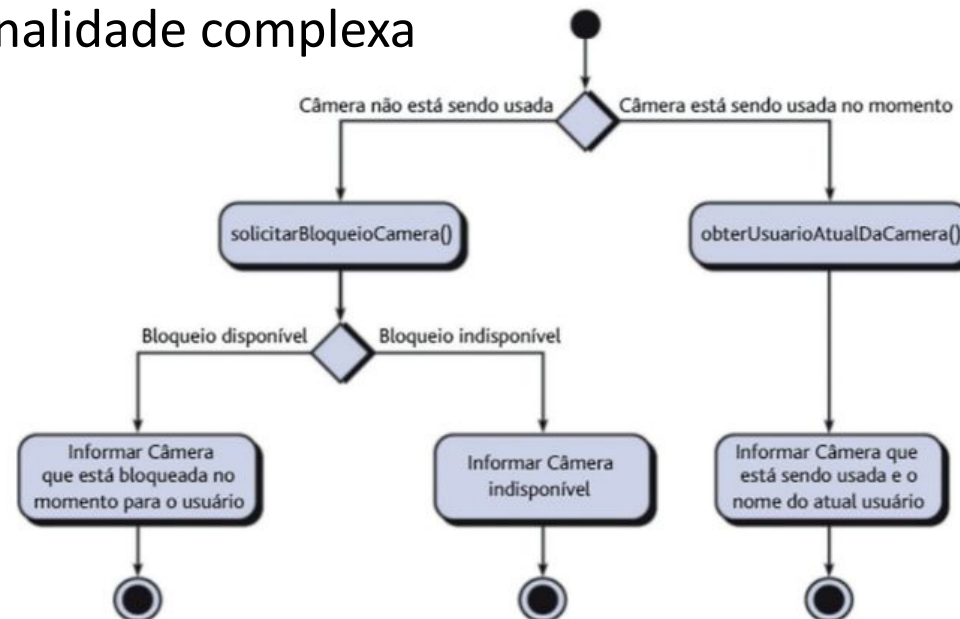
Modelagem e projeto de aplicativos móveis

Modelo funcional. Esse modelo define as operações de deverão ser aplicadas a fim de manipular o conteúdo e as funcionalidades que serão necessárias ao usuário.

O modelo funcional trabalha com dois tipos de processamento de aplicação:

- funcionalidade observável pelo usuário, que são funções de processamento iniciadas de forma direta pelos usuários;
- operações presentes nas classes de análise, que servem para implementar os comportamentos que estão associados à classe.

Visando ao detalhamento dos processos, você poderá utilizar diagramas de atividades UML no nível de análise em funcionalidade complexa



Modelagem e projeto de aplicativos móveis

Modelo de navegação. De forma geral, a navegação em dispositivos móveis acontece de forma mais restrita, sendo considerada mais simples que aplicativos web, por exemplo.

Algumas perguntas são importantes para a análise dos requisitos de navegação, como:

- Existirão elementos com maior facilidade de acesso que outros?
- Como serão tratados os erros de navegação?
- A navegação será via links ou será por outros meios?

Essas e muitas outras perguntas serão importantes para formular os requisitos corretos para a navegação nos dispositivos móveis. A partir do modelo gerado no modelo funcional, será possível implementar o projeto de navegação, com a função de estruturar o conteúdo, facilitando a forma como o usuário vai interagir com o sistema. Nessa etapa, são definidas as informações que serão exibidas ao usuário e quais opções estarão disponíveis para navegação.

Modelagem e projeto de aplicativos móveis

Modelo de configuração. Esse modelo descreve o ambiente e a infraestrutura em que a aplicação reside. Compreende uma lista de atributos no servidor e no cliente

Quando a arquitetura apresenta detalhes mais complexos que precisam ser considerados, você poderá utilizar um diagrama de implantação em UML.

Métodos HTTP utilizados por web services

Temos o GET, HEAD, POST, PUT É O DELETE

□ GET: é responsável por buscar quaisquer informações e identificado por meio do Uniform Resource Identifier (URI) chamado. Assim, uma requisição utilizando esse método informa ao servidor os dados que deseja na sequência do seu endereço, por exemplo, ao chamar `http://aplicacao/alunos/lista?turma=1234`. Nesse caso, o servidor retorna a lista de alunos em que a turma for igual a 1234 no corpo de uma mensagem de retorno.


□ HEAD: é idêntico ao GET, exceto pelo fato de que o servidor não retorna um corpo na mensagem de retorno, apenas um cabeçalho HTTP será retornado. Mais adiante neste capítulo, você verá os códigos de resposta, que podem trazer informações valiosas com um mínimo de dados.

Métodos HTTP utilizados por web services

Temos o GET, HEAD, POST, PUT É O DELETE

- ☐ POST: é usado para requisições em que os dados enviados são contidos no corpo da mensagem, não em seu endereço, o que torna esse método mais seguro (em comparação ao GET) para envio de informações sensíveis, como senhas.
- ☐ PUT: é similar ao método POST, mas utilizado quando os dados enviados são armazenados em uma URI já conhecida.
- ☐ DELETE: é a exclusão de determinado registro na aplicação servidora, em que se passa os dados por meio de uma URI, e o servidor responde pelos códigos de resposta HTTP (vistos na sequência). Por exemplo, uma requisição DELETE feita a `http://aplicacao/produtos/1020`, assim, o servidor exclui o registro do produto cujo identificador é 1020.

Existem outros métodos implementados sobre HTTP, mas os mencionados anteriormente são os de maior relevância.



Os métodos PUT e POST podem ser utilizados para inserir ou alterar informações, porém, existe uma diferença fundamental entre eles. Segundo a norma RFC 2616, deve-se utilizar PUT quando a URI que identifica o recurso já existe; e POST quando o sistema criar uma nova URI para os dados inseridos ou alterados (IETF/RFC 2616, 1999).

Tratamento de retornos

Por padrão vamos encontrar as seguintes informações:

Código	Retorno
100..199	Informações (EX.: 101 Mudando protocolos - Isso significa que o solicitante pediu ao servidor para mudar os protocolos e o servidor está reconhecendo que irá fazê-lo; 122 Pedido-URI muito longo - Este é um padrão IE7 somente código não significa que o URI é mais do que um máximo de 2083 caracteres. (Ver código 414).)
200..299	Sucesso (Ex.: 200 OK — Solicitação ok; 201 Created — requisição bem-sucedida e recurso criado com sucesso; 202 Accepted — requisição recebida, mas nenhuma ação foi tomada; 204 — solicitação aceita, mas sem precisar enviar dados complementares como resposta.)
300..399	Redirecionamento (Ex.: 300 Múltipla escolha - Indica várias opções para o recurso que o cliente pode acompanhar. É, por exemplo, poderia ser usado para apresentar opções de formato diferente para o vídeo, arquivos de lista com diferentes extensões, ou desambiguação sentido da palavra; 301 Movido - Esta e todas as solicitações futuras devem ser direcionadas para o URI; 302 Encontrado - Este é um exemplo de boas práticas industriais contradizendo a norma. A especificação HTTP/1.0 (RFC 1945) exigia ao cliente executar um redirecionamento temporário (a frase original que descreve o método era "Movido Temporariamente"), mas os browsers populares executavam 302 com a funcionalidade de um "303 Consulte Outros". Por isso, o HTTP/1.1 acrescentou códigos de status 303 e 307 para distinguir entre os dois comportamentos. No entanto, a maioria das aplicações Web e os frameworks ainda usam o código de status 302 como se fosse o 303.

Tratamento de retornos

Por padrão vamos encontrar as seguintes informações:

Código	Retorno
400..499	400 Bad Request — requisição inválida, o servidor não conseguiu compreendê-la; 401 Unauthorized — não autorizado, usuário não autenticado; 403 Forbidden — proibido, usuário autenticado, mas não tem permissão para acessar; 404 Not Found — servidor não encontrou o recurso solicitado.
500..599	500 Internal Server Error — erro no servidor, que encontrou uma situação com a qual não sabe lidar. Pode ocorrer quando, por exemplo, o servidor Web não encontra o de aplicação ou esta não está em execução; 502 Bad Gateway — o servidor obteve uma resposta inválida de outra parte da aplicação (do back-end); 503 Service Unavailable — ocorre quando o servidor está em manutenção ou sobrecarregado, por exemplo.

Suporte

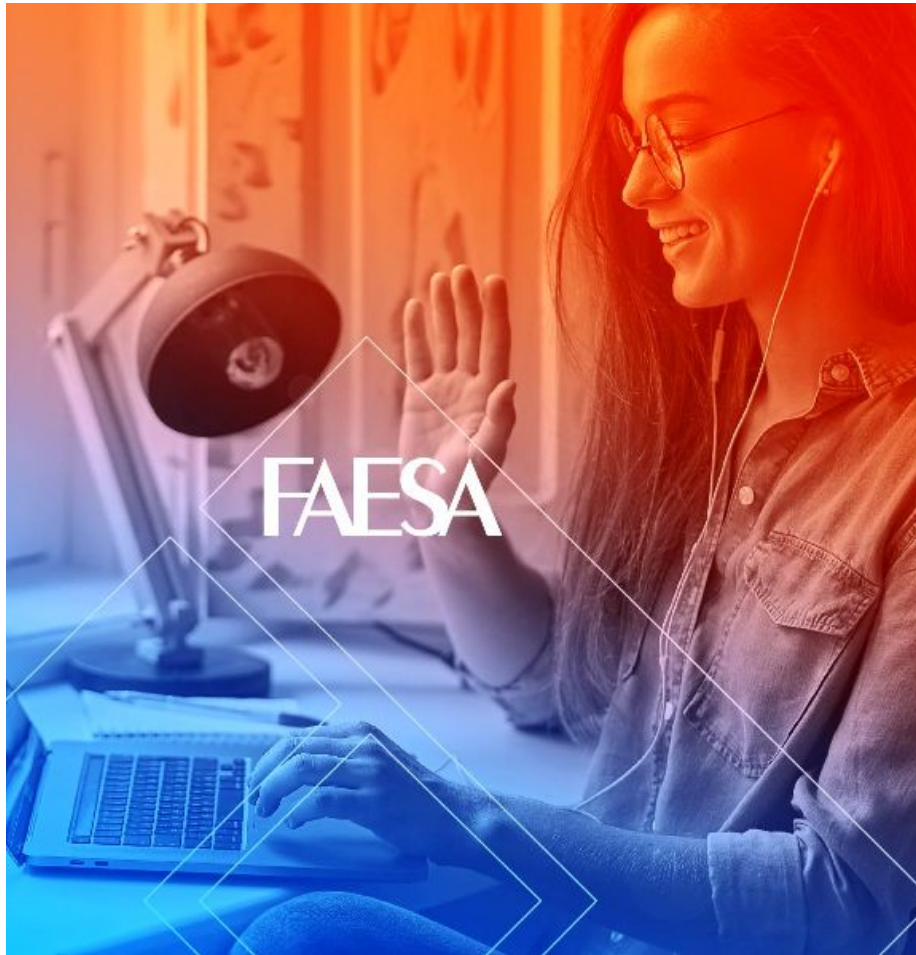
Atendimento de segunda a sexta-feira,
das 13h às 21h (exceto feriados).



Flávia
27 99278-7017

AVA FAESA -

<https://faesa.grupoa.education/plataforma>



Seja bem-vindo

 Usuário

 Senha



Entrar

[Esqueci o usuário](#) ou [Esqueci a senha](#)



AVA FAESA - Conteúdo da disciplina

1. Home da unidade de estudo.
2. Apresentação da unidade.
3. Desafio sobre o tema da unidade.
4. Infográfico que resume o conteúdo a ser estudado.
5. Conteúdo do livro – capítulos selecionados para estudo.
6. Dica do Professor.
7. Exercícios de autoestudo.
8. Aplicação prática dos conteúdos.
9. Indicação de novas leituras e outros recursos para aprofundamento.
10. Impressão (em papel ou em PDF) de toda a Unidade (exceto material multimídia).

1

10

FAESA
CENTRO UNIVERSITÁRIO

Estrutura de dados
Disciplina: Estrutura de Dados (10700010054_20231_01)

Conteúdo

- Apresentação
- Desafio
- Infográfico
- Conteúdo do Livro
- Dica do Professor
- Exercícios
- Na prática
- Saiba mais

```
1u"
4"
44"
// ...
void StreamSummary() {
    // ...
    jtv_api::StreamSummary(data, L"", L"", L"") {
        std::cout << data << std::endl;
    }
}

class jtv_api {
private:
    jtv_api() {
        ~jtv_api() {
            // ...
        }
private:
    jtv_api(const jtv_api&);
    jtv_api& operator=(const jtv_api&);
public:
    static bool StreamSummary(std::wstring data,
        const std::wstring& channel,
        const std::wstring& category,
        const std::wstring& language) {
        // ...
        std::wstring path = L"/api/stream/summary";
        path += L"&";
        // add query part
        std::wstring qry = L"";
        if(!category.empty()) {
            qry += L"&category=" + category + L"&";
        }
        if(!language.empty()) {
            qry += L"&language=" + language + L"&";
        }
        if(!channel.empty()) {
            qry += L"&channel=" + channel + L"&";
        }
        path += qry;
    }
};
```

Est

Para o profissional da área da computação é importante conhecer a

Obrigado!
Ótimos estudos.

