Desenvolvimento para dispositivos móveis

Introdução

Professor Msc. Fabio Pereira da Silva
E-mail: fabio.pereira@faculdadeimpacta.com.br

Agenda

- Introdução
- Versões do Android
- Arquitetura
- Componentes
- Visão geral

Evolução da telefonia móvel

http://d24w6bsrhbeh9d.cloudfront.net/photo/a3YYeAr_700b_v1.jpg

Evolução da telefonia móvel

- Primeiros telefones móveis tinham uma funcionalidade: fazer ligações.
- Primeiro telefone: Motorola DynaTAC
  - $4000,00
  - 790 g
  - 25 cm (sem antena)

http://img.timeinc.net/time/photoessays/2010/100_gadgets/communication/motorola_dynatac.jpg

Evolução da telefonia móvel

- Com o tempo, novas funcionalidades foram agregadas:
  - Agenda
  - SMS
  - Despertador
  - Jogos
  - Calculadora
  - Tela colorida
  - ...

Evolução da telefonia móvel

- Hoje vivemos a era dos Smartphones
  - Acesso a internet
  - Tela sensível ao toque
  - Câmera
  - GPS
  - Jogos mais complexos
  - Música
  - Armazenamento de dados
  - E inclusive faz ligação!

Evolução da telefonia móvel

- Smartphones são computadores de mão
  - Os mais modernos tem capacidade de processamento igual ou superior a desktops
- Estas características possibilitam a instalação de programas no telefone móvel.
  - O usuário pode customizar o telefone com os programas ou aplicativos.
- Com isso, abriu-se um novo mercado na área de tecnologia: desenvolvimento de aplicativos para dispositivos móveis

Porque desenvolver para mobile?

- Auxilia e agiliza seu negócio
  - Integrar aplicativo móvel com sistemas de back-end
- Agrega valor ao seu produto
- Pessoas acessam seu produto de qualquer lugar
  - Mais de 1,3 bilhão de pessoas têm smartphone[1]
  - 168 milhões no brasil [2]
  - Acesso de qualquer lugar
- Mercado em alta[3]
  - Estimativa de 529.000 empregos diretos na Europa, sendo 60% em desenvolvimento

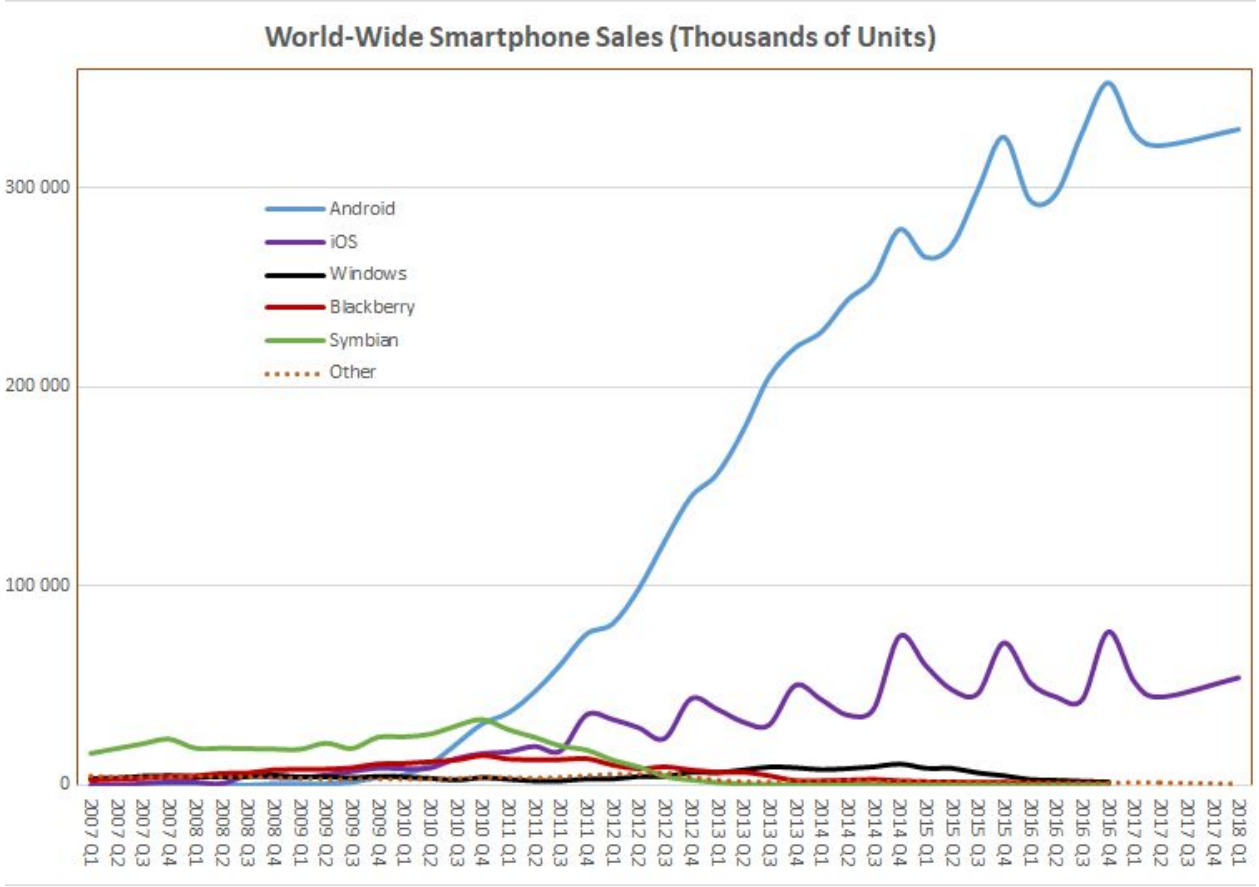
[1] http://www.digitimes.com/news/a20151217PD209.html
[2] http://www.imcgrupo.com/impress/gt/upload/PesTI2016GVcia.pdf
[3] http://www.visionmobile.com/product/the-european-app-economy

Porque desenvolver para mobile?

- Poucas opções de arquiteturas
  - Quase 99% dos smartphones vendidos no mundo em 2016 utilizam Android e iOS [1]
  - Maior facilidade para manter a portabilidade e disponibilidade de apps

[1] http://en.wikipedia.org/wiki/Mobile_operating_system

Por que desenvolver para mobile?



Fonte: https://upload.wikimedia.org/wikipedia/commons/8/83/World_Wide_Smartphone_Sales.png

Principais plataformas

iOS

- Sistema Operacional da Apple Inc.
- iPhone, iPad, iPod, Apple TV
- Sistema proprietário
  - Não é licenciado para outros hardwares
  - Não é de código aberto
- Versão atual: 13.x

iOS

- Desenvolvimento (oficial)
  - Objective C ou Swift
  - Necessita de um computador da Apple com:
    - Mac OS X 10.12.6 ou superior
    - Xcode – ambiente de desenvolvimento (gratuito)
    - iOS SDK – biblioteca
  - Registro de desenvolvedor ($99 / ano)
    - https://developer.apple.com/support/compare-memberships/
    - Existe um programa para universidades, mas é limitado

iOS

- Desenvolvimento
  - Os aplicativos desenvolvidos só podem ser testados no simulador, em aparelhos registrados para seu login de desenvolvedor ou enviando para telefones selecionados (utilizando TestFlight Beta Testing)
  - Não é possível instalar o aplicativo em outros dispositivos antes de publicar na App Store
  - Após a submeter o aplicativo, este será avaliado pela Apple, que libera ou não a publicação do aplicativo

iOS

- Passos básicos para desenvolver para iOS
  - Ter um computador Apple (pago)
  - Baixar Xcode e iOS SDK (gratuito)
  - Cadastrar um Apple ID (gratuito)
  - Cadastrar-se como desenvolvedor (pago)
  - Cadastrar e publicar aplicativo (gratuito)

iOS

- Vantagens
  - SO é otimizado para o hardware
  - Rápido para testar no simulador
  - Construtor de interfaces
  - Opções de resolução de tela bem definidas
- Desvantagens
  - Poucas opções de hardware (menos usuários)
  - Sistema proprietário
  - Maior custo para desenvolvimento
  - Apenas uma opção de plataforma de desenvolvimento

Android

- Sistema operacional da Open Handset Alliance (http://www.openhandsetalliance.com)
  - Liderado pelo Google
  - Motorola, LG, Samsung, Sony, HTC, Toshiba, Telefonica,...
  - Mais de 47 membros, entre provedores, fabricantes de hardware e fabricantes de software
- Baseado em Linux
  - Mesmo kernel de SO como Ubuntu, Debian e outros
  - Android é responsável por... gerenciar memória

Android

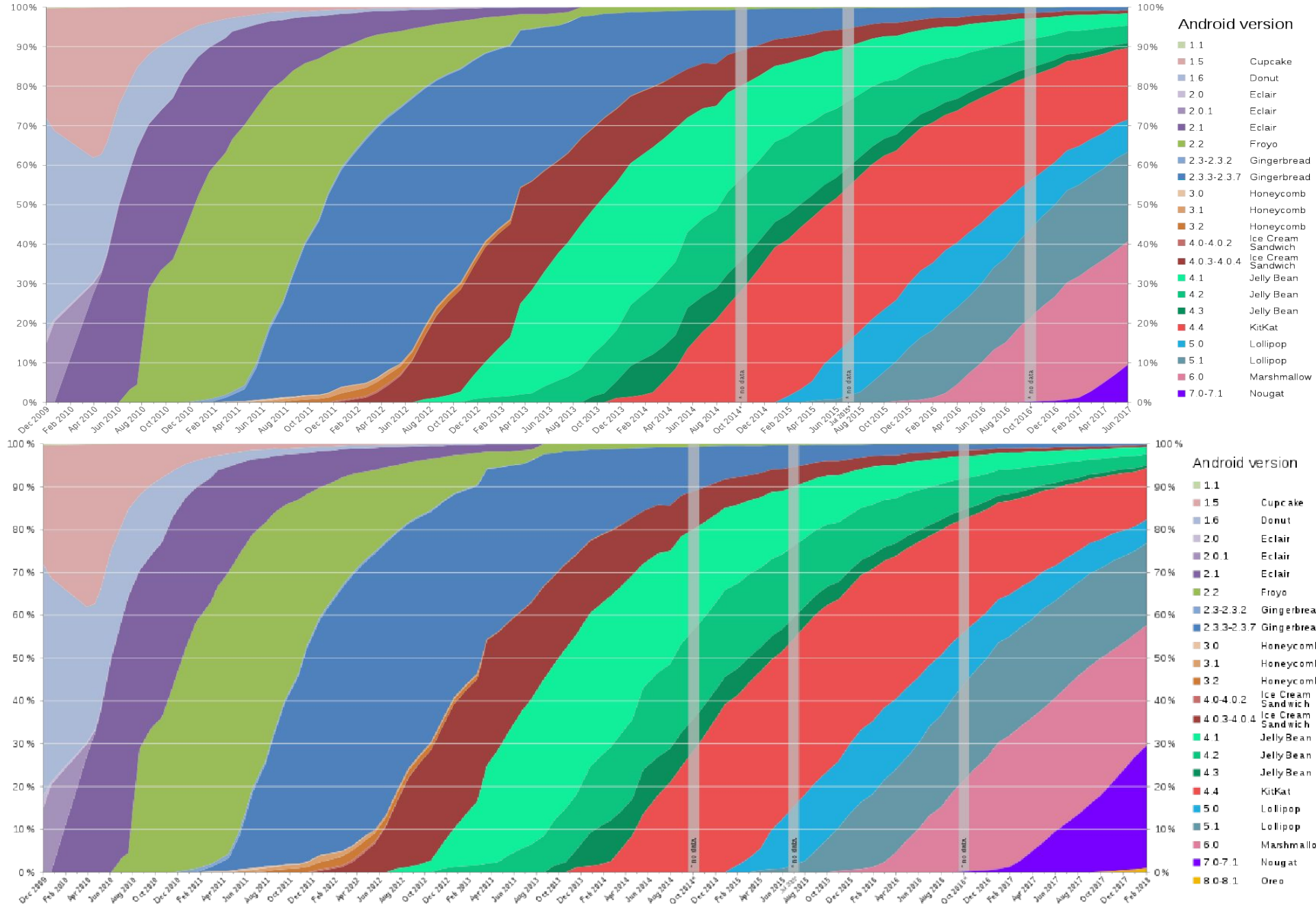
- Apesar de ser escrito em Java e Kotlin, os apps para Android não utilizam a JVM e os byte codes não são executados.
- As classes Java são compiladas para executáveis da máquina virtual Dalvik
- Dalvik é uma máquina virtual especializada desenvolvida e otimizada especificamente dispositivos móveis.

Android

- Sistema de uso livre e de código aberto (Apache License)
  - Fabricantes customizam o SO
  - Contribui para que o aperfeiçoamento do Android para a plataforma
  - Não é necessário compartilhar as alterações
- Versão atual
  - 10 (Android 10)

Versões do Android

Nome	Número	Data	API
-	1.0	23 setembro de 2008	1
-	1.1	9 de fevereiro de 2009	2
Cupcake	1.5	27 de abril de 2009	3
Donut	1.6	15 de setembro de 2009	4
Eclair	2.0 – 2.1	26 de outubro de 2009	5 – 7
Froyo	2.2 – 2.2.3	20 de maio de 2010	8
Gingerbread	2.3 – 2.3.7	6 de dezembro de 2010	9 – 10
Honeycomb	3.0 – 3.2.6	22 de fevereiro de 2011	11 – 13
Ice Cream Sandwich	4.0 – 4.0.4	18 de outubro de 2011	14 – 15
Jelly Bean	4.1 – 4.3.1	9 de julho de 2012	16 – 18
KitKat	4.4 – 4.4.4	31 de outubro de 2013	19 – 20
Lollipop	5.0 – 5.1.1	12 de novembro de 2014	21 – 22
Marshmallow	6.0 – 6.0.1	5 de outubro de 2015	23
Nougat	7.0 – 7.1.2	22 de agosto de 2016	24 – 25
Oreo	8.0 – 8.1	21 de agosto de 2017	26 – 27
Pie	9	6 de agosto de 2018	28
Android 10	10	3 de setembro de 2019	29

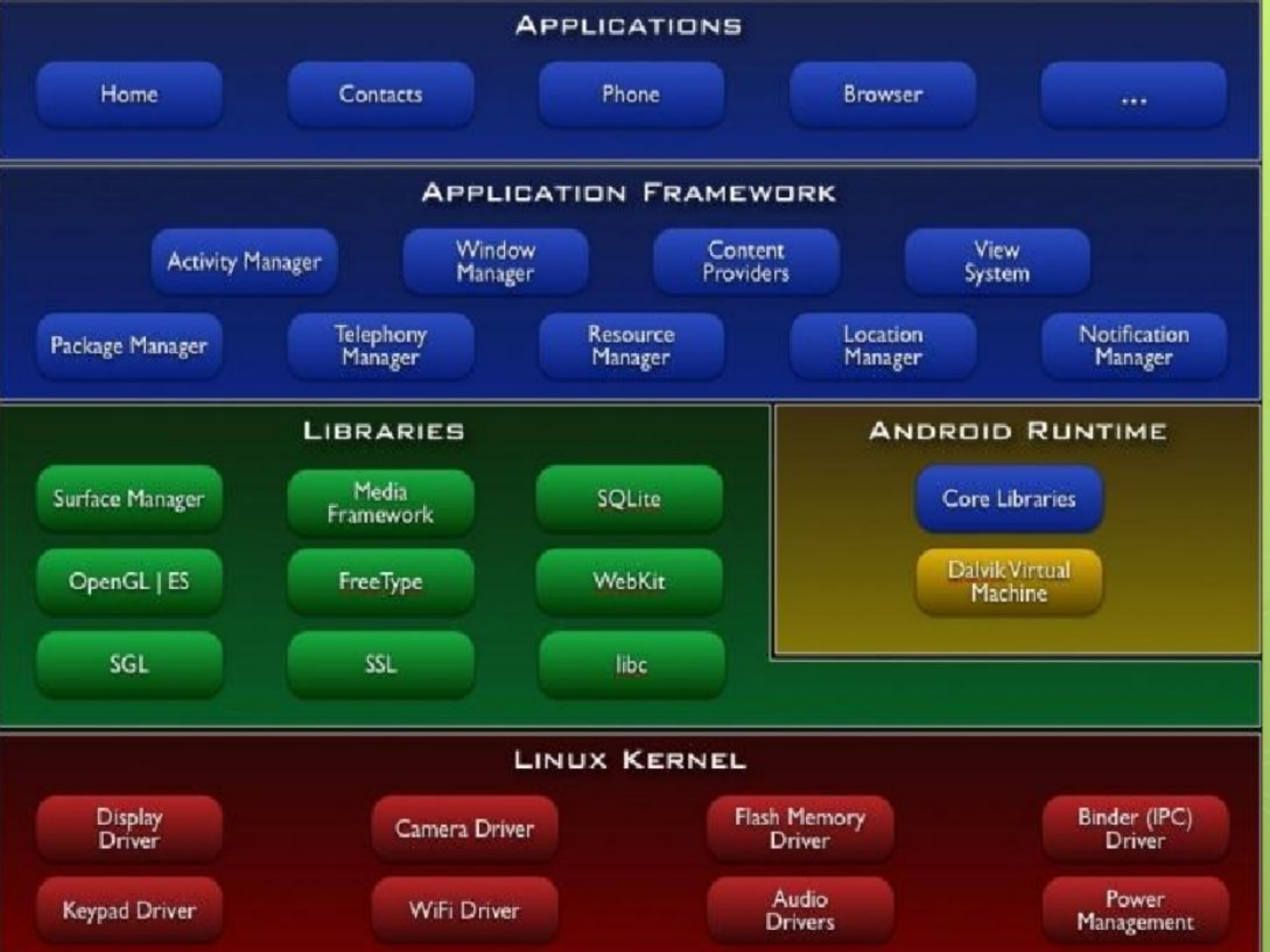


https://en.wikipedia.org/wiki/Android_version_history#/media/File:Android_historical_version_distribution_-_vector.svg

Arquitetura do Android

- A arquitetura do sistema operacional Android é dividida em camadas.
- Cada parte é responsável por gerenciar seus respectivos processos.

Camadas	Descrição
Camada de aplicações	Onde encontram-se todos os aplicativos que são executados sobre o sistema operacional.
Camada de bibliotecas	Possui as bibliotecas C e C++ que são utilizadas pelo sistema.
Camada de Runtime	Camada onde é instanciada a máquina virtual Dalvik para gerenciamento do desempenho.
Camada de kernel do Linux	O núcleo do sistema operacional Android é derivado do Kernel 2.6 do Linux herdando várias características da plataforma.


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

Android – Desenvolvimento (oficial)

- Linguagem Java ou Kotlin
- Qualquer computador com
  - JDK
  - Android SDK
  - Android Studio
- Totalmente customizável
- É possível substituir aplicativos nativos
  - Tela inicial, agenda, mensagens...

Android – Desenvolvimento

- Pode-se integrar aplicativos desenvolvidos com qualquer outro aplicativo, de forma muito simples
- Emulador
  - Simula SMS, chamada telefônica, conexão com internet
- Pode-se testar o aplicativo no emulador ou em qualquer dispositivo com Android conectado ao computador
  - Outra opção é copiar o arquivo .apk para o dispositivo
  - O arquivo .apk é gerado após construir o projeto (build)
- Custos

Android

- Passos básicos para desenvolver para Android
  - Ter qualquer computador (pago, mais barato)
  - Baixar (todos gratuitos):
    - Java JDK (versão 7 ou superior) (http://www.oracle.com/technetwork/java/javase/dow_nloads/index.html)
    - Android Studio (já vem com SDK) (https://developer.android.com/studio)

Android

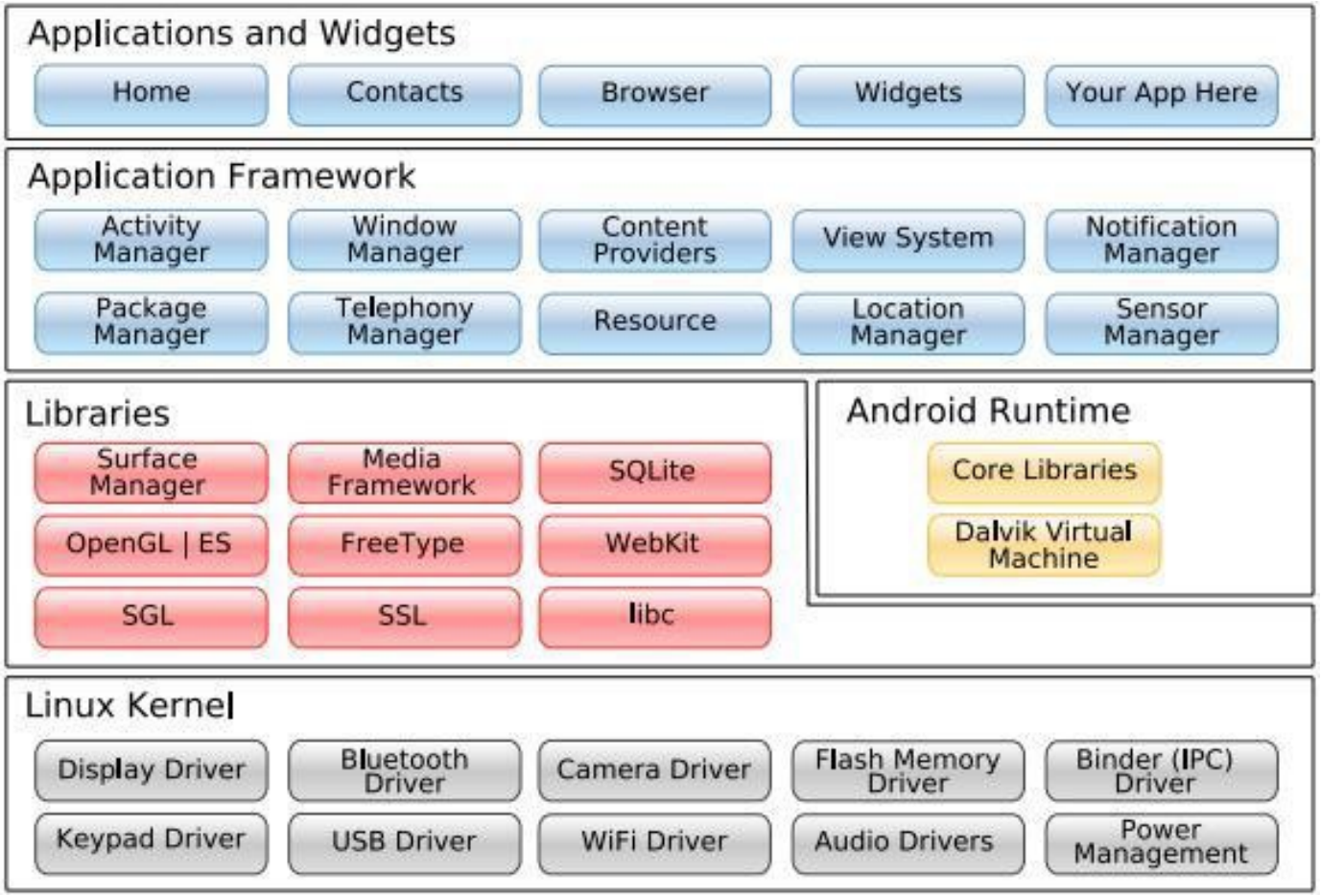
- Passos básicos para desenvolver para Android
  - Registrar uma conta Google (gratuita)
  - Pagar a taxa de desenvolvimento (pago)
  - Publicar aplicativo (gratuito)
    - O aplicativo não passa por processo de revisão

Android

- Vantagens
  - Várias opções de hardware (mais usuários)
  - Sistema livre e de código aberto
  - Menor custo para desenvolvimento
  - Desenvolvimento em qualquer plataforma
- Desvantagens
  - SO não é otimizado para o hardware
  - Emulador mais lento
  - Muitas opções de resolução de tela

Android

- Arquitetura



http://kebomix.wordpress.com/2010/08/17/android-system-architecture/

iOS x Android

iOS	Android
Buy a Mac, download the free Xcode Installer from the Mac App Store, and start writing code	download the SDK, setup Eclipse and install Google's ADT Plugin
development is done in Objective-C	development is done in Java or C/C++
deploy costs $99/yr and app has to pass a screening process	Google simple takes a $25 flat fee to shelf your apps
iOS "Simulator" - runs native code	Android Emulator - runs on a virtual machine
Debug takes 5 seconds on the iOS Simulator	Debug takes about 30 seconds to redeploy and start up in the Emulator on a perfectly-modern machine
has Interface Builder	create UI layouts in XML
Devices have known screen dimensions and hardware	suffers from fragmentation - many versions of the OS and Devices on the market

http://www.slideshare.net/technologythree/introduction-to-mobiledevelopment#

Outras formas de desenvolver aplicativos

- Cada plataforma tem sua forma oficial de desenvolvimento
  - Android: Java/Kotlin + Android Studio
  - iOS: Swift + Xcode + MacOS
- Quando o aplicativo é desenvolvido na plataforma oficial, ele é chamado de aplicativo nativo
  - Geralmente melhor desempenho
  - Biblioteca é otimizada para uso do hardware e recursos do hardware
  - Não há camada intermediária entre o aplicativo e abiblioteca do SO

Outras formas de desenvolver aplicativos

- Entretanto, o desenvolvimento nativo não é a única escolha para desenvolver um aplicativo
- Existem basicamente 4 tipos de aplicativos:
  - Aplicativos web
  - Aplicativos híbridos, que agregam características das duas abordagens
  - Aplicativos multiplataforma nativos
  - Aplicativos nativos

Aplicativos Web - características

- Conhecido com Web App
- Acesso pelo navegador ou WebView
- Funciona como um site ou sistema web
  - Normalmente tem uma interface/layout específico para mobile
    - Responsivo
    - Interações e usabilidade diferentes

Aplicativos Web - características

- Desenvolvimento é mais rápido
  - Utiliza tecnologias como HTML, JS e CSS
  - Muito recurso humano disponível
  - A lógica o back-end ou serviço pode ser desenvolvido na linguagem mais apropriada para a equipe
- Recursos de hardware limitado
  - Não acessa tudo que o dispositivo oferece
- Não é acessível pela loja de aplicativos
- Totalmente dependente da internet

Aplicativos Web

- Usuários visualizam mais rapidamente
  - É mais fácil abrir seu app no browser do que pagar para baixar um aplicativo
  - Por estar na web, também é mais fácil de atrair novos usuários fora da plataforma móvel
- Publicação mais rápida
  - Para disponibilizar nova versão basta atualizar os arquivos para o servidor
- Atualização automática
  - Usuário não precisa atualizar o aplicativo

Aplicativos Web

- Globalização
  - Um mesmo projeto pode servir para várias plataformas e dispositivos
    - Android, iOS, Windows, Web...
    - Ou então pode-se aproveitar pelo menos o back-end, onde estará a maior parte das regras de negócio
    - O acesso direto ao servidor também economiza energia do dispositivo
    - Mas acaba gastando mais do plano de dados
- Grande parte da lógica e navegação é executada e um servidor web

Aplicativos Web "Embutido"

- Alguns aplicativos são desenvolvidos como um Web App mas acessados de um aplicativo nativo
- O aplicativo nativo é criado apenas para acessar o Web App em uma web view (assim como faz o navegador) e poder disponibilizar em uma loja de aplicativos
- Continua com limitação de uso dos recursos do hardware
- Continua com dependência da internet

Aplicativos Híbridos

- Desenvolvimento que utiliza tecnologias da web (HTML, CSS e JavaScript), mas que executa dentro de um aplicativo nativo
  - A interface é desenvolvida utilizando HTML, CSS e JavaScript
  - A lógica é feita com JavaScript
    - Não é possível utilizar outra linguagem
    - Apenas tecnologias de cliente web são utilizadas

Aplicativos Híbridos

- O que foi criado utilizando HTML, CSS e Javascript é colocado em um projeto Nativo
  - possibilita acesso a recursos do dispositivo
    - GPS, câmera, contatos, push, notificações
  - O que foi criado com HTML é executado em uma WebView
    - A diferença para um Web App “embutido” é que a parte web está executando no aplicativo, e não em um servidor externo

Aplicativos Híbridos

- Depende de uma biblioteca de terceiros (Cordova, Phonegap)
  - Camada colocada entre o aplicativo e a biblioteca da plataforma
- Desenvolvimento mias rápido, já que utiliza tecnologia web
  - Precisa apenas de um rápido aprendizado para aprender como acessar os recursos nativos pela biblioteca auxiliar
- Possibilita que o aplicativo funcione offline

Aplicativos Híbridos

- Possibilita o desenvolvimento de aplicativos multiplataforma
  - Qualquer plataforma tem um recurso de WebView
- Como é multiplataforma, atinge um maior número e usuários
- Pode publicado na loja e instalado no dispositivo
- Pode ter perda de desempenho para aplicativos muito complexos
- A maior parte das características de um aplicativo nativo estão presentes nos híbridos

Aplicativos Híbridos

- Principais bibliotecas e ferramentas
  - Cordova
  - Phonegap
  - Ionic
  - Appcelerator
  - JqueryMobile
  - Intel XDK

Aplicativos Multiplataforma nativos

- Não utiliza a linguagem da plataforma para desenvolvimento, mas também não executa em uma WebView
- Os componentes são nativos, não HTML
- O funcionamento pode ser diverso:
  - Compilação para a linguagem nativa (compilação cruzada)
  - Biblioteca que acessa direto a biblioteca da plataforma

Aplicativos Multiplataforma nativos

- Possuí a maioria das características de um aplicativo híbrido ou nativo:
  - Multiplataforma (apenas híbrido)
  - Acesso a recursos do dispositivo
  - Funcionamento offline
  - Publicação na loja
  - Instalação no dispositivo

Aplicativos Multiplataforma nativos

- O desempenho depende muito da solução que for utilizar
  - Usualmente uma tecnologia que utiliza compilação cruzada tende a ter menor desempenho, uma vez que não temos controle do que é feito na compilação
- Principais tecnologias
  - Xamarin: desenvolvimento utilizando C#
  - Nativescript: desenvolvimento utilizando Javascript, Typescript e XML
  - React Native: desenvolvimento utilizando Javascript, Typescript e XML

Aplicativos Nativos

- Documentação mais completa
- Maior facilidade para reportar erros
- Bibliotecas próprias
  - Recursos como orientação da tela e multi-toque são suportados pelo navegador (WebKit) no Web App
  - Porém Agenda, GPS, acelerômetro estão apenas em aplicativos nativos

Aplicativos Nativos

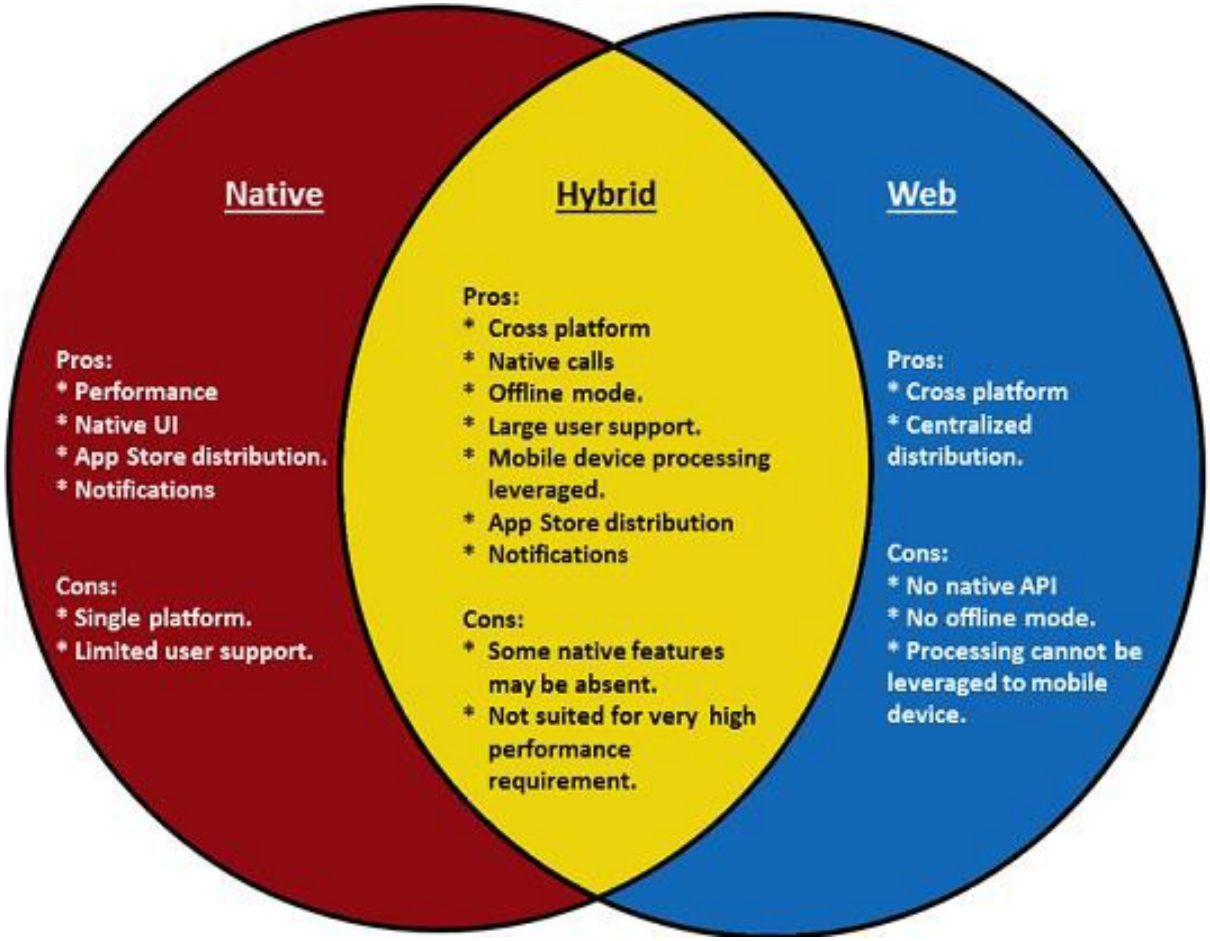
- Disponível na loja de aplicativos
- Instalação no dispositivo
- Criação de um ícone de acesso rápido
- Funcionamento offline
- Velocidade nativa do dispositivo
- Maior segurança

Aplicativos Nativos

- Grande parte das regras de negócio e navegação é executada no dispositivo
- Funciona como um software desktop, com possibilidade de uma arquitetura cliente/servidor (consumo de serviços)
- Armazenamento de dados localmente
- Tarefas em segundo plano

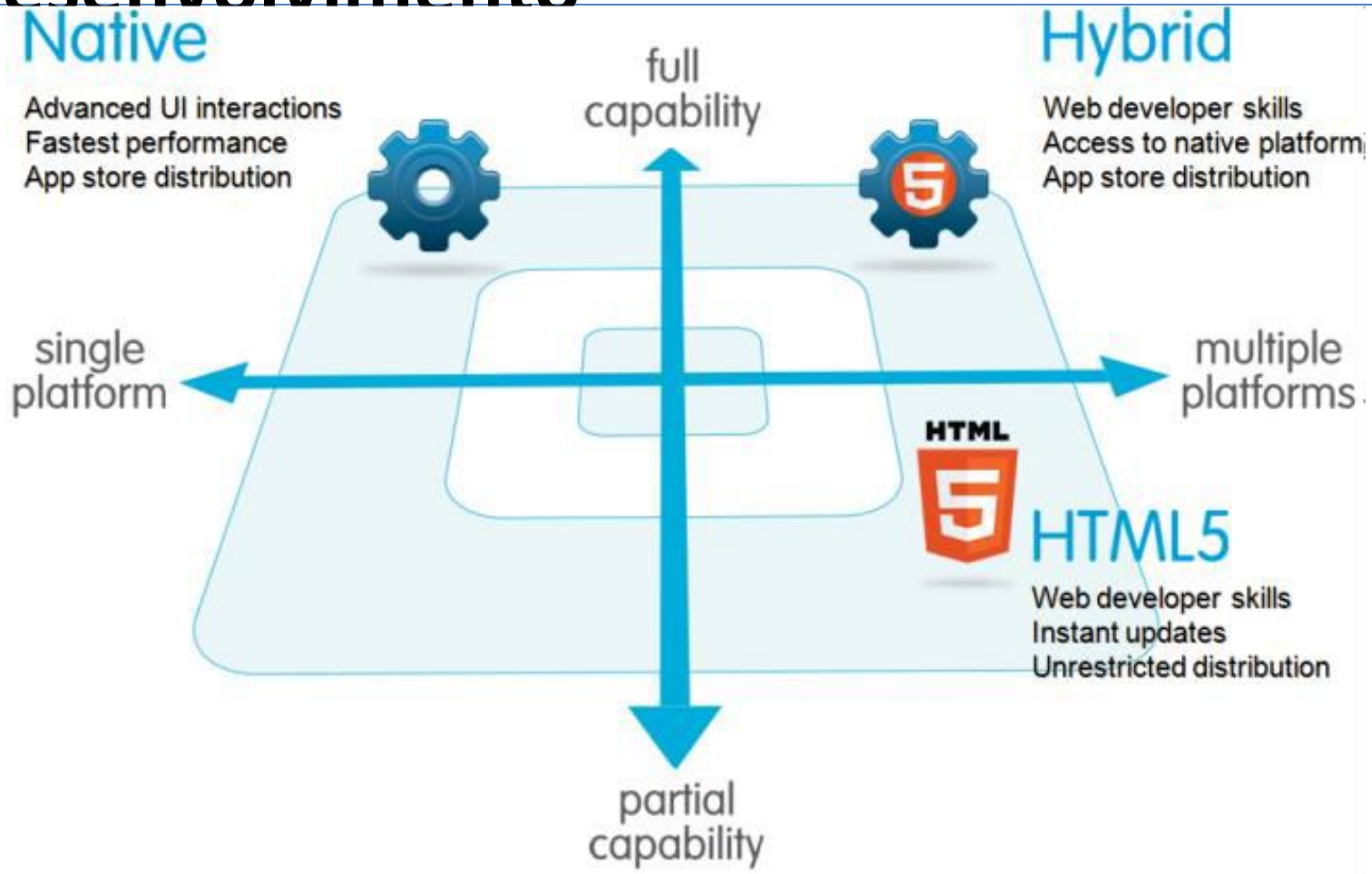
Estratégias de desenvolvimento

- Comparação



http://developer.nokia.com/community/wiki/Arquitetura_de_aplica%C3%A7%C3%B5es_multiplat
aforma_para_dispositivos_m%C3%B3veis

Estratégias de desenvolvimento



https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options

Estratégias de desenvolvimento

- Porque escolher um aplicativo nativo?
  - Pode funcionar sem a internet
  - Melhor performance
  - Mais fácil de encontrar
  - Armazenamento de dados locais
  - Utilizar recursos do dispositivo
  - Executar tarefas em segundo plano (notificações)
  - Melhor usabilidade e navegação
    - Independente da interpretação do browser de HTML/JS/CSS
  - Boa alternativa quando:
    - Aplicativo é bastante sofisticado
    - Precisa utilizar recursos do dispositivo

Estratégias de desenvolvimento

- Quando escolher um aplicativo nativa?
  - Jogos interativos
  - Uso regular
  - Cálculos complexos
  - Armazenamento de dados
  - Uso de recursos do dispositivo
  - Não precisa de conexão com a internet

Estratégias de desenvolvimento

- Porque escolher um aplicativo web?
  - Pode ser o suficiente para algumas aplicações
  - Desenvolvimento em HTML5, JS e CSS
  - Roda na maioria dos browsers modernos
  - Fácil publicação e manutenção
  - Boa alternativa quando:
    - Aplicativos que não usam recursos no dispositivo
    - Muitas atualizações
    - É totalmente dependente da internet

Modelos de desenvolvimento

- Existe a melhor?
  - Não existe a melhor estratégia para desenvolver sua solução
  - Tudo depende de tempo, dinheiro, conhecimento e necessidades do negócio
    - Vale a pena investir em um aplicativo nativo?
    - O aplicativo web supre minhas necessidades?
    - Qual o público alvo? Os usuários ou clientes estão interessados em um aplicativo móvel?
- Em algumas situações alguns modelos de desenvolvimento podem ajudar:

Modelos de desenvolvimento

- Desenvolvimento Espelhado
  - É comum que um produto tenha uma versão gratuita mais simples e uma versão paga completa
    - Conhecido como “freemium”
  - Alguns fazem um aplicativo nativo gratuito limitado, e outro completo
  - Outra abordagem é disponibilizar uma versão na web e outra nativa
    - A versão web é gratuita e limitada
    - A versão nativa é paga e completa, com funcionalidades que utilizam recursos do

Modelos de desenvolvimento

- Desenvolvimento Misto
  - Parte do produto é nativo, parte do produto é web
  - Comum quando há dois tipos de usuários
    - Ex.: Administradores de um sistema não utilizaram recursos do dispositivo, apenas vão fazer configuração e entrada de dados. Estes não precisam de um sistema nativo, um sistema web supre as necessidades. Por outro lado, você pode prover uma melhor experiência para usuários finais com uma aplicação nativa, que tira proveito dos recursos gráficos, velocidade, e recursos.

Modelos de desenvolvimento

- Modelo cliente-servidor
  - Modelo extremamente comum, utilizado em grande parte dos aplicativos
  - Combina um back-end desenvolvido para a web com um front-end nativo
    - O que na realidade acaba se tornando um aplicativo nativo, que utiliza dados que estão na web.
  - Grande reaproveitamento de código das regras de negócio
    - Pode-se pensar em fazer diferentes interfaces para diferentes dispositivos a fim de aumentar a

Referências

- https://developer.android.com/training/basics/firstapp?hl=pt-br
- https://www.sncticet.ufam.edu.br/2017/downloads/christianreis.pdf
- https://homepages.dcc.ufmg.br/~fernando/classes/android/slides/Class1.pdf
- https://pt.slideshare.net/AnaDoloresLimaDias/android-9149956
- https://developer.android.com/guide/components/activities/activity-lifecycle?hl=pt-br
- https://www.tutorialspoint.com/android/android_hello_world_example.htm