

Tópicos Avançados de Programação

Aula 01

Prof. Marcelo Rodrigues do Nascimento

CONVERSA INICIAL

Olá, seja bem-vindo à primeira aula da disciplina Tópicos Avançados em Programação. Nossos objetivos, neste encontro, serão:

- Compreender os conceitos introdutórios da plataforma de desenvolvimento Android
- Preparar o ambiente para o desenvolvimento Android Studio
- Criar a primeira aplicação para Android no Android Studio

Para tanto, será preciso estudarmos o seguinte:

- A arquitetura Android
- Introdução ao desenvolvimento Android
- Componentes de Aplicativo
- A IDE Android Studio

Como você deve saber, o mercado de aplicativos para dispositivos móveis, como celulares e tablets, está em franco crescimento, com uma base de dispositivos ativados que cresce a cada ano. Nosso objetivo é iniciá-lo no mundo dos aplicativos Android, criando uma boa base de conhecimento a respeito de sua arquitetura e proporcionando familiaridade com os diversos conceitos embutidos neste padrão de desenvolvimento.

Ao finalizarmos esta etapa, você terá compreendido os conceitos relacionados à criação de aplicativos, respeitando suas regras e padrões de desenvolvimento. Também criará seu primeiro aplicativo, utilizando a plataforma de Android Studio.

CONTEXTUALIZANDO

A plataforma Android foi criada para suportar dispositivos móveis, como celulares e tablets. Por tratar-se de uma plataforma que busca trabalhar com dispositivos computacionais com restrições de recursos (como energia, por exemplo, já que estes dispositivos normalmente funcionam utilizando uma bateria), o desenvolvimento deste tipo de aplicativo diferencia-se do desenvolvimento para sistemas desktop ou servidores, e possui regras que devem ser entendidas e respeitadas.

Novos mercados surgem a cada dia neste mercado, desde utilitários a aplicativos de entretenimento, como jogos, por exemplo. Também é interessante ressaltarmos que o volume de pessoas acessando a internet através de dispositivos móveis – principalmente celulares – está quase superando o acesso por um computador normal. Segundo dados do International Data Corporation (IDC), o Android está presente em quase 85% dos smartphones, em mais de 190 países ao redor do mundo. De acordo com a Google, a cada dia, mais de um milhão de usuários ligam seus dispositivos Android pela primeira vez.

Muitas empresas vêm se especializando no desenvolvimento de aplicativos próprios ou para terceiros, o que aumentou a busca por profissionais da área. Além disso, a Google disponibiliza uma plataforma de distribuição chamada Google Play Store, que torna seu aplicativo disponível para download dentro de poucas horas. A mesma plataforma mantém o desenvolvedor informado a respeito do número de usuários que utilizam seu aplicativo, feedback do mesmo e eventuais problemas encontrados.

Pesquisa

Que tal nos ambientarmos com a interface do Android Studio? Pesquise no site Android Studio a respeito de sua interface e opções de visualização. O Android Studio vem com seu próprio emulador, o AVD. No entanto, para alguns dispositivos, o AVD pode se mostrar lento. Uma excelente opção a ele é o Genymotion, disponível em: <<https://www.genymotion.com/>>.

Experimente os dois emuladores e encontre aquele que melhor se adequa às suas necessidades.

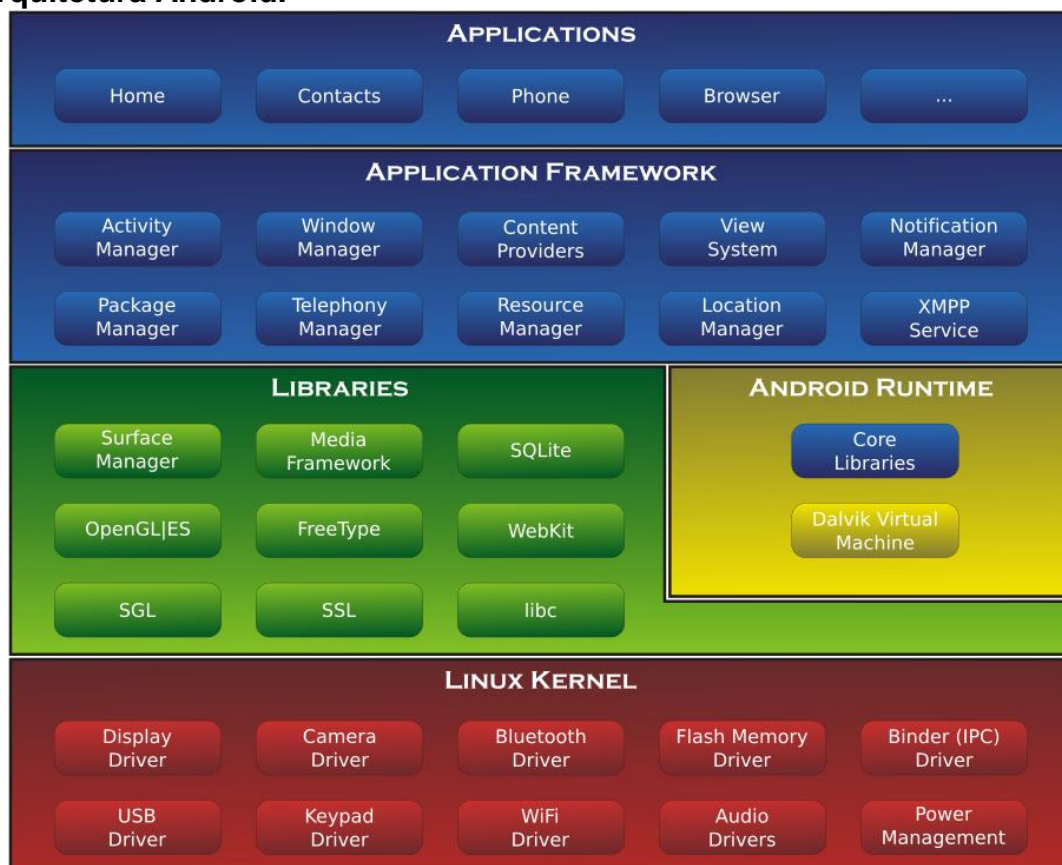
Aproveite para conhecer mais a respeito da plataforma de desenvolvimento Android. Uma excelente fonte de pesquisa é o livro **Android 6 para Programadores – Uma Abordagem Baseada em Aplicativos**, de Deitel, Paul; DEITEL, Harvey e OSWWALD, Alexander. Este livro possui uma extensa explicação referente a Arquitetura Android, seu ambiente de desenvolvimento Android Studio, além de exercícios rápidos, que aumentarão em muito sua experiência.



Tema 1 - A arquitetura Android

A arquitetura Android é baseada em uma pilha organizada em 4 camadas. Na camada inferior, temos o Linux Kernel. Acima desta, a camada de Bibliotecas e Runtime do Android. Acima desta, a camada de Framework de Aplicações, que suporta o desenvolvimento de novas aplicações, chegando finalmente à última camada, que contém as Aplicações disponíveis no dispositivo, como contatos, telefone, navegador de internet entre outras.

Arquitetura Android.



Linux Kernel

A primeira camada, Linux Kernel, é a camada mais baixa da plataforma Android. Esta camada contém os serviços essenciais disponíveis a todos os seus dispositivos. Gerenciamento de processos e de memória são feitos nesta camada.

O Linux Kernel também contém os drivers de dispositivos (como teclado, áudio, câmera), que tornam nossa tarefa mais fácil quando temos que criar interface de acessos aos mesmos.

Linux Kernel.



Nesta camada encontramos também a Arquitetura de Permissões, que restringe o acesso a dados ou recursos somente a processos que foram autorizados. Aqui, também, encontramos componentes específicos para Android, como, por exemplo, gerenciamento de bateria, compartilhamento e gerenciamento.

Por que um Linux Kernel?

- Excelente gerenciamento de memória e processos
- Modelo de segurança baseado em permissões
- Modelo de controle aprovado
- Suporte a bibliotecas compartilhadas
- Código aberto

Libraries

Na camada de libraries (bibliotecas), encontramos temos as bibliotecas de sistema, normalmente escritas em C e C++, e que são utilizadas para o bom funcionamento do sistema operacional. Também encontramos bibliotecas Java criadas especificamente para Android.

Libraries.



Entre estas bibliotecas podemos destacar:

- **Surface Manager** – atualização da tela
- **SQLite** – utilizada para acesso a dados publicados por provedores de conteúdo e que inclui também as classes de gerenciamento do banco de dados SQLite
- **SSL** – (Secure Sockets Layer), que é utilizada para prover segurança na internet
- **OpenGL** – que dá acesso para interface Java aos gráficos 3D OpenGL/ES e a API de renderização
- **Media Framework** – utilizada para fornecer acesso à diferentes codecs (programas utilizados para codificar e decodificar arquivos de mídia), que possibilitam a gravação e execução de formatos diferentes
- **WebKit** – utilizada para mostrar conteúdo da internet ou conteúdo HTML

Android Runtime

É o terceiro componente da arquitetura Android, e é colocado na segunda camada (a partir de baixo). Este componente é responsável por fornecer a Core Java Library e a Dalvik Virtual Machine. As aplicações Android são normalmente criadas na linguagem de programação Java e, para tornar mais simples o desenvolvimento destas aplicações, o Android fornece uma grande gama de blocos reutilizáveis Java, como por exemplo **java.*** e **javax.***, os pacotes **android.***, responsáveis pelo ciclo de vida dos aplicativos Android e os pacotes **org.***, que suportam operações de rede.

A Dalvik Virtual Machine é o software que executa as aplicações Android, e é similar ao Java Virtual Machine (JVM), mas desenvolvida e otimizada para esta plataforma. A Dalvik utiliza funções principais do Java, como gerenciamento de memória e multithreading, que possibilita que cada aplicação Java rode em seu próprio processo.

Android Runtime.



Application Framework

A terceira camada interage diretamente com o framework de aplicação, que gerencia as funções básicas de um dispositivo Android, como gerenciamento de recursos, gerenciamento de chamadas, etc. Blocos importantes do framework de aplicação são:

- **Activity Manager** – bloco utilizado para gerenciar o ciclo de vida completo das aplicações
- **Content Providers** – utilizado para gerenciar o compartilhamento de dados entre as aplicações
- **Telephony Manager** – utilizado para gerenciar chamadas de voz
- **Location Manager** – utilizado para gerenciar as localizações obtidas utilizando-se o GPS ou uma torre de celular
- **Resource Manager** – este bloco é responsável por gerenciar os tipos diferentes de recursos utilizados em uma aplicação Android

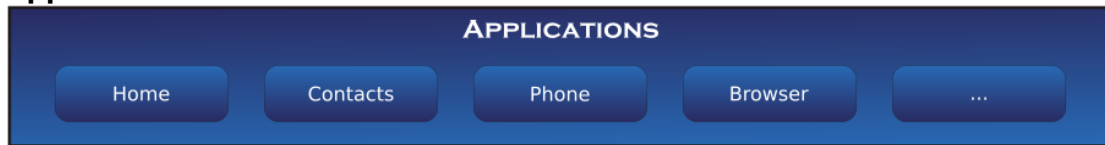
Application Framework.



Applications

Nesta camada estão as aplicações criadas por diferentes fornecedores ou desenvolvedores. O Android vem com aplicações próprias, que incluem o Home, Contatos, Telefone, Browser e leitor de e-mails, entre outras. Um ponto importante é que nenhum destes aplicativos é obrigatório no seu sistema. Se você escrever um aplicativo melhor, pode substituir aqueles que são fornecidos por padrão.

Applications.



Tema 2 - Introdução ao desenvolvimento Android

O Android possui um Software Development Kit (SDK), que é utilizado para o desenvolvimento de suas aplicações e é composto de bibliotecas e ferramentas de desenvolvimento distribuídas juntamente com a plataforma Android Studio, que pode ser baixada gratuitamente a seguir:

<https://developer.android.com/develop/index.html>

Para entender mais a respeito da história por trás da plataforma de desenvolvimento, uma sugestão é o primeiro capítulo do livro **Android 6 para Programadores – Uma Abordagem Baseada em Aplicativos**.

Os aplicativos para Android são geralmente desenvolvidos em Java, linguagem de programação robusta de código aberto, orientada a objetos e com acesso a amplas bibliotecas de classe que auxiliam no desenvolvimento. As ferramentas Android SDK compilam o código e geram, em conjunto com todos os arquivos e recursos, um arquivo Android Application Package (APK), que é o pacote do Android.

Este pacote normalmente possui a seguinte estrutura:

- **META-INF**
 - MANIFEST.MF – arquivo de manifesto da aplicação
 - CERT.RSA – O certificado da aplicação
 - CERT.SF – Contém os hashes SHA1 para os arquivos incluídos no pacote

- **lib** – diretório que contém o código compilado específico para o tipo de processador
 - armeabi – código compilado para processadores ARM
 - armeabi-v7a – código compilado para todos os processadores ARMv7 e acima
 - arm64-v8a – código compilado para processadores ARMv8 e superior
 - x86 – código compilado para processadores x86
 - x86_64 – código compilado para processadores x86 64
 - mips – código compilado para processadores MIPS
- **res** – diretório que contém todos os recursos não compilados no diretório resources.arsc
- **assets** – diretório contendo todos os arquivos de componente bruto necessários para o funcionamento da aplicação, os quais podem recuperados através do AssetManager
- **AndroidManifest.xml** – um arquivo de manifesto adicional, que descreve o nome, versão, direitos de acesso e bibliotecas utilizadas pela aplicação.
- **Classes.dex** – as classes compiladas para o formato dex, que é entendido pela máquina virtual Dalvik
- **Resources.arsc** – arquivo pré-compilado de recursos, como por exemplo XML binário

O sistema operacional Android é um sistema Linux multiusuário, em que cada aplicativo é um usuário diferente, ou seja, ele é instalado em sua própria área de segurança. Por questões de segurança, o sistema Android implementa o princípio do privilégio mínimo, ou seja, cada aplicativo tem acesso somente aos componentes necessários para sua execução.

Tema 3 - Componentes de aplicativo

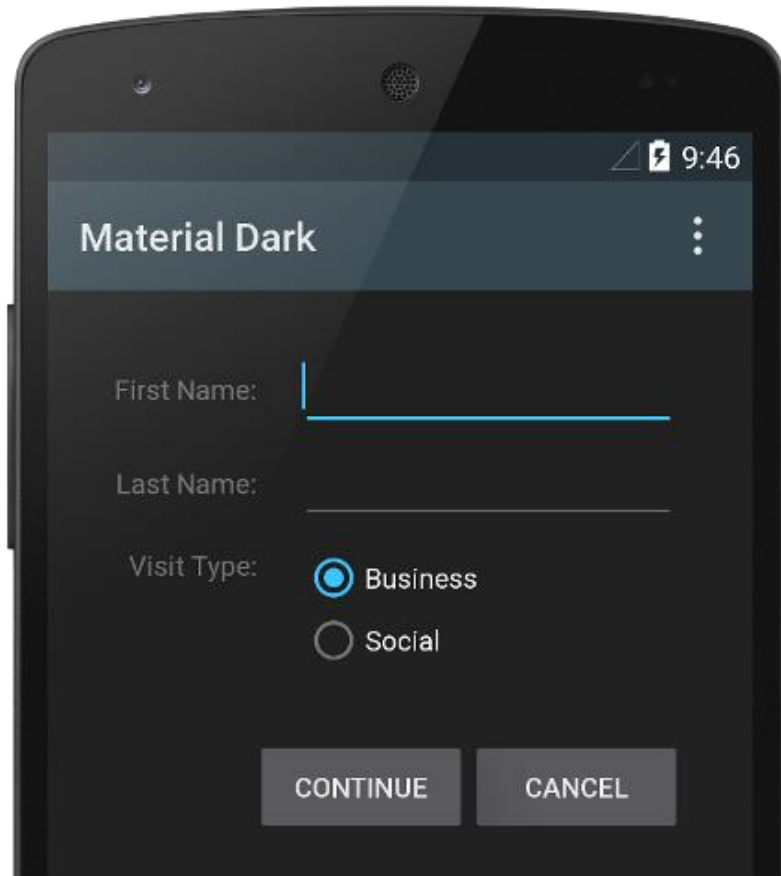
No desenvolvimento Android, contamos com “blocos de construção”, chamados componentes de aplicativos. Cada componente é uma forma diferente pela qual o sistema pode “entrar” no aplicativo. Os aplicativos para Android são criados como uma combinação de componentes distintos que podem ser invocados individualmente. Por exemplo, uma atividade individual fornece uma única tela para a interface de usuário e um serviço realiza trabalhos em segundo plano de forma independente.

De um componente é possível executar outro componente, utilizando uma **intenção**. É possível até mesmo iniciar um componente em um aplicativo diferente, como uma atividade em um aplicativo de mapas para mostrar um endereço. Este modelo fornece vários pontos de entrada para um único aplicativo e permite que qualquer aplicativo se comporte como o “padrão” de um usuário em uma ação que outros aplicativos podem invocar, como enviar um e-mail, compartilhar um link.

Atividades (Activity)

Representam uma tela única de interface de um usuário. É implementada como uma subclasse de Activity. Estas fornecem a tela com a qual os usuários podem interagir. Cada atividade recebe uma janela, que representa a interface do usuário, com botões e caixas de texto, por exemplo. Aplicativos geralmente possuem várias atividades vinculadas entre si. Um aplicativo normalmente possui uma atividade principal, que é mostrada ao usuário quando este o executa pela primeira vez. Após isso, cada atividade pode iniciar uma nova atividade para executar diferentes ações, como, por exemplo, listar os e-mails recebidos, mostrar detalhes de um determinado e-mail e finalmente responde-lo.

As atividades são ordenadas em forma de pilha no formato UEPS (último a entrar, primeiro a sair). A cada nova atividade criada, a atividade anterior é interrompida, mas é armazenada na pilha, para que possa ser reestabelecida quando o usuário pressionar o botão de retorno.

Activity.**Serviços (Services)**

Componentes executados em segundo plano para realizar operações de longa duração ou executar trabalhos remotos. Não representam uma interface com o usuário. Um serviço pode ter duas formas:

- **Iniciado** – um componente do aplicativo, como uma activity, por exemplo, inicia-o chamando `startService()`. Após iniciado, o serviço pode ficar em segundo plano indefinidamente, mesmo que o componente que o chamou seja destruído. Neste modelo, normalmente não temos o retorno da operação para o componente que originou a chamada.
- **Vinculado** – neste caso, o componente vincula-se ao serviço, iniciando-o pela chamada `bindService()`. Ao ser vinculado, o serviço criará uma interface cliente-servidor, que permitirá que os componentes interajam

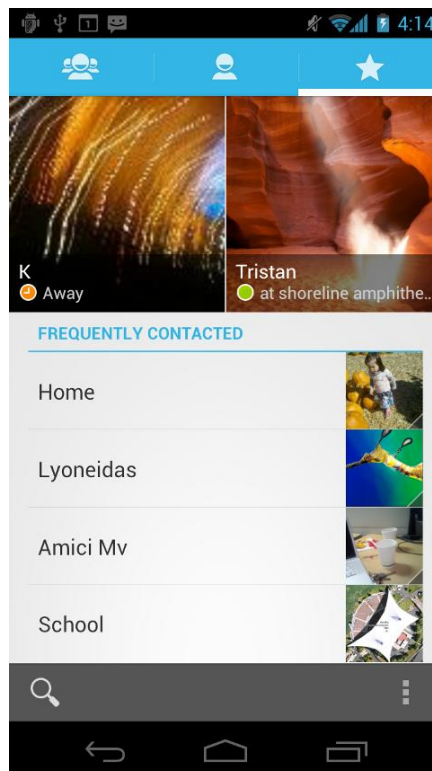
com este, enviando solicitações e recolhendo resultados. Este tipo de serviço somente permanece em execução enquanto o componente que criou o vínculo mantiver o vínculo com o mesmo. Ao se destruir o componente, ou ao se desvincular o serviço, o mesmo é destruído.

Provedores de Conteúdo (Content Providers)

São responsáveis por prover às aplicações o conteúdo das quais elas necessitam para funcionar, ou seja, os dados. Ao utilizarmos provedores de conteúdo, tornamos a forma como os dados são gravados transparentes à aplicação, o que permite que esta mantenha o foco nas interações com os usuários. Através do provedor de conteúdo, os aplicativos acessam e até modificam os dados gerados por outros aplicativos instalados, como, por exemplo, a lista de contatos, caso o provedor de conteúdo permita seu acesso.

O Android inclui provedores para gerenciamento de imagens, áudio e vídeo, além de informações de contatos pessoais. Ainda que com algumas restrições, esses provedores podem ser acessados por qualquer aplicativo Android.

Contatos.



Receptores de Transmissão (Broadcast Receivers)

Este é o componente que responde (e inicia) anúncios a todo o sistema Android. Transmissões de que o sistema está com a bateria baixa ou que a tela está ligada são originados no sistema. Aplicativos podem iniciar transmissões, como, por exemplo, de que uma imagem foi capturada, informando a outros aplicativos sobre este evento, possibilitando assim que os mesmos disparem alguma ação.

Ao contrário de uma Activity, um BroadcastReceiver não possui qualquer tipo de interface com o usuário, mas possibilita a criação de uma notificação na barra de status. A intenção é de que os receptores de transmissão efetuem a menor quantidade de serviços possível, delegando seu trabalho para os Services.

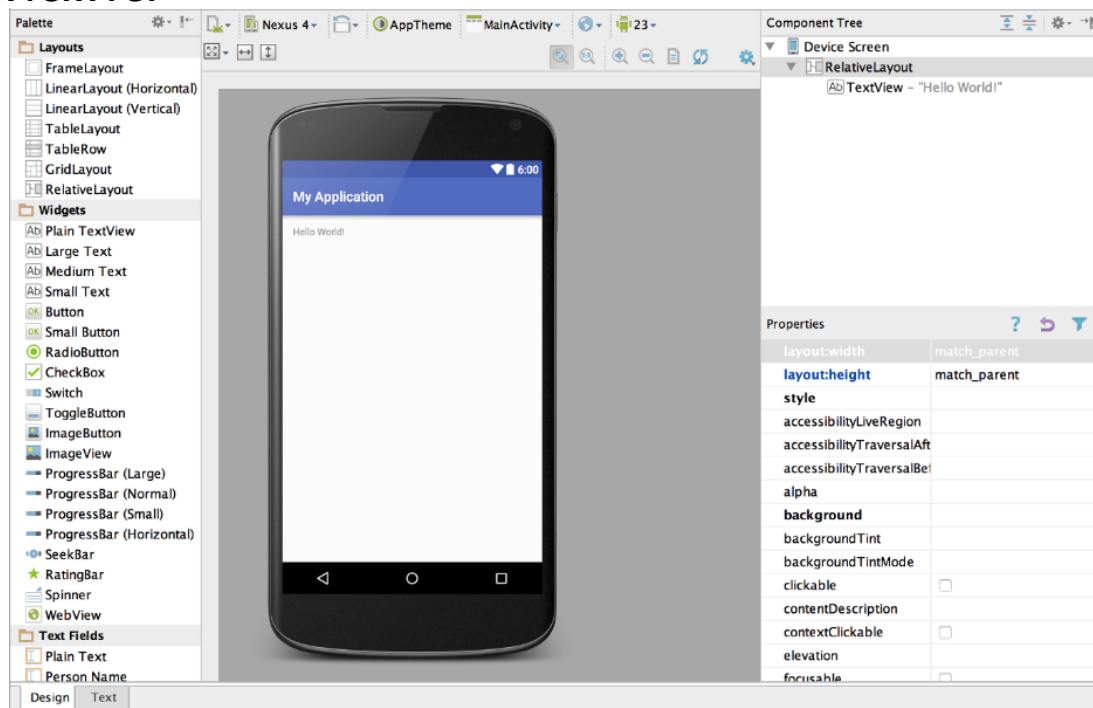
Notificação.



Tema 4 - A IDE Android Studio

O Android Studio é a Integrated Development Environment, ou Ambiente integrado de Desenvolvimento (IDE), oficial para desenvolvimentos de aplicativos Android e é baseado na versão comunitária do IntelliJ IDEA. Ele vem com todos os pacotes para desenvolvimento na plataforma Android, incluindo o Software Development Kit (SDK), ferramentas e emuladores. Possui um editor WYSIWYG (What you see is what you get – O que você vê é o que você é o que você obtém), termo utilizado para classificar ferramentas de edição e desenvolvimento, que permitem visualizar em tempo real o que será publicado.

WYSIWYG.



É baseado no sistema de build Gradle, um sistema de automatização de alta performance, responsável pelo gerenciamento de dependências, empacotando tudo em um único arquivo, no caso do Android um arquivo APK. Possui um editor de códigos inteligente, com recursos avançados de sugestão de código, refatoramento e análise.

Esta IDE possui recursos que permitem o desenvolvimento para todas as plataformas que utilizem Android em um ambiente único, incluindo telefones, tablets, Android TV, Android Wear e Android Auto.

Seleção de plataforma para desenvolvimento.

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately 97.4% of the devices that are active on the Google Play Store.
[Help me choose](#)

☒ Wear

Minimum SDK API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass

Minimum SDK Glass Development Kit Preview (API 19)

Cancel Previous **Next** Finish

Antes de iniciarmos a instalação do Android Studio, precisamos do Java Development KIT (JDK) instalado em nossa máquina. Para obter a versão mais recente da JDK, acesse a seguir:

<http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html>

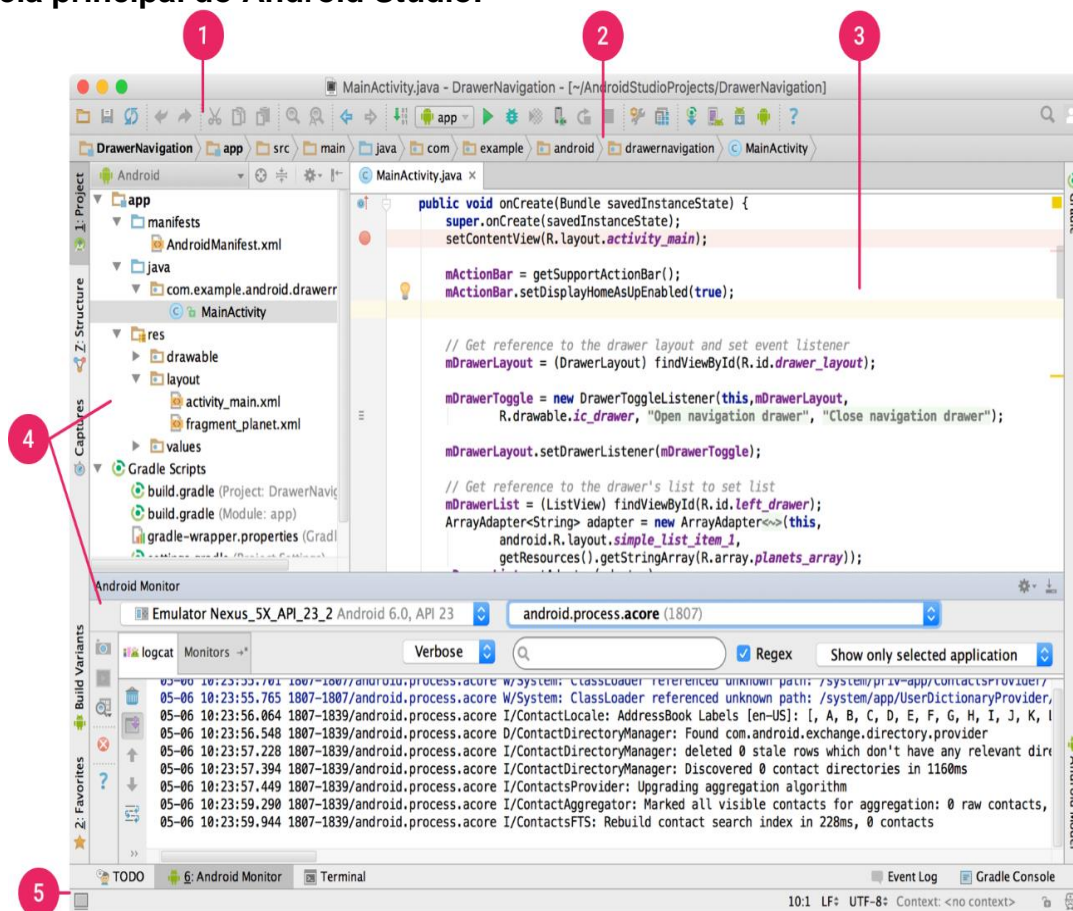
Após o download, efetue a instalação para que seu sistema esteja pronto para receber o Android Studio. A última versão do Android Studio pode ser encontrada no seguinte link:

<https://developer.android.com/studio/index.html>

Para facilitar sua instalação, sugiro que acompanhe o seguinte vídeo de instalação e configuração do Android Studio no seguinte link:

https://www.youtube.com/watch?v=h4_2fCJJdg0

Tela principal do Android Studio.



1. A barra de ferramentas, que permite um grande número de ações, incluindo a execução de seu App;
2. A barra de navegação o ajuda a navegar pelos arquivos abertos de seu projeto para edição;

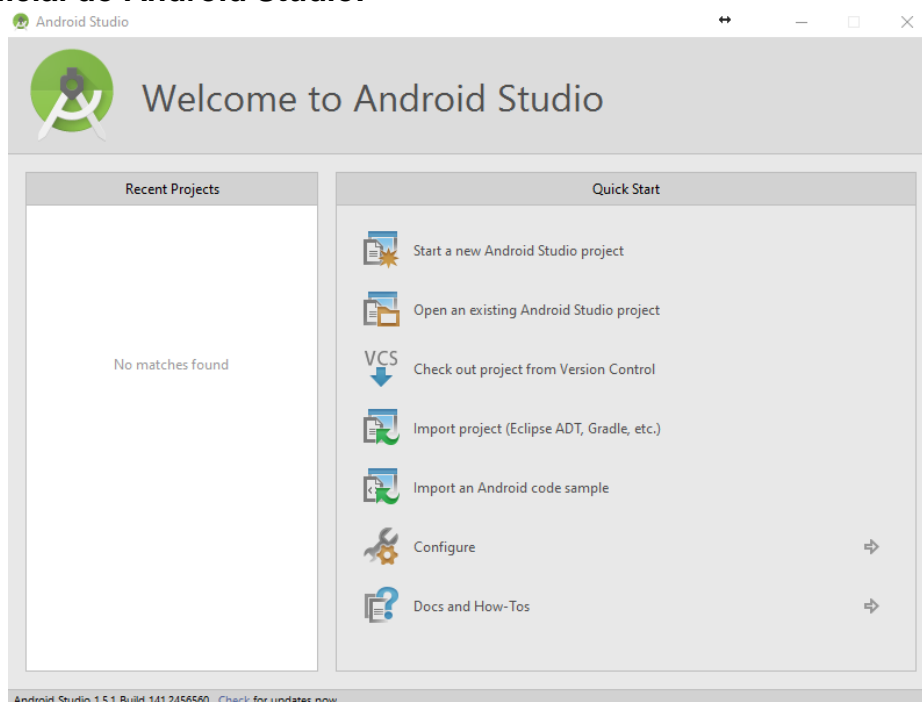
3. A tela de edição é onde você cria e modifica seu código. Dependendo do tipo de arquivo, esta tela pode mudar. Por exemplo, ao visualizar um arquivo de layout, o editor é alterado para o editor de layouts e oferece a opção de visualização do arquivo XML correspondente;
4. A janela de ferramentas proporciona acesso a tarefas específicas de gerenciamento do projeto, busca, controle de versão e mais;
5. A barra de status mostra o status de seu projeto de da própria IDE, além de mensagens e avisos.

NA PRÁTICA

Criando um primeiro aplicativo

Agora que temos o Android Studio devidamente instalado e configurado em nosso ambiente de desenvolvimento, chegou a hora de criarmos nossa primeira aplicação. Clique duas vezes no ícone do Visual Studio para que a tela de boas-vindas seja apresentada:

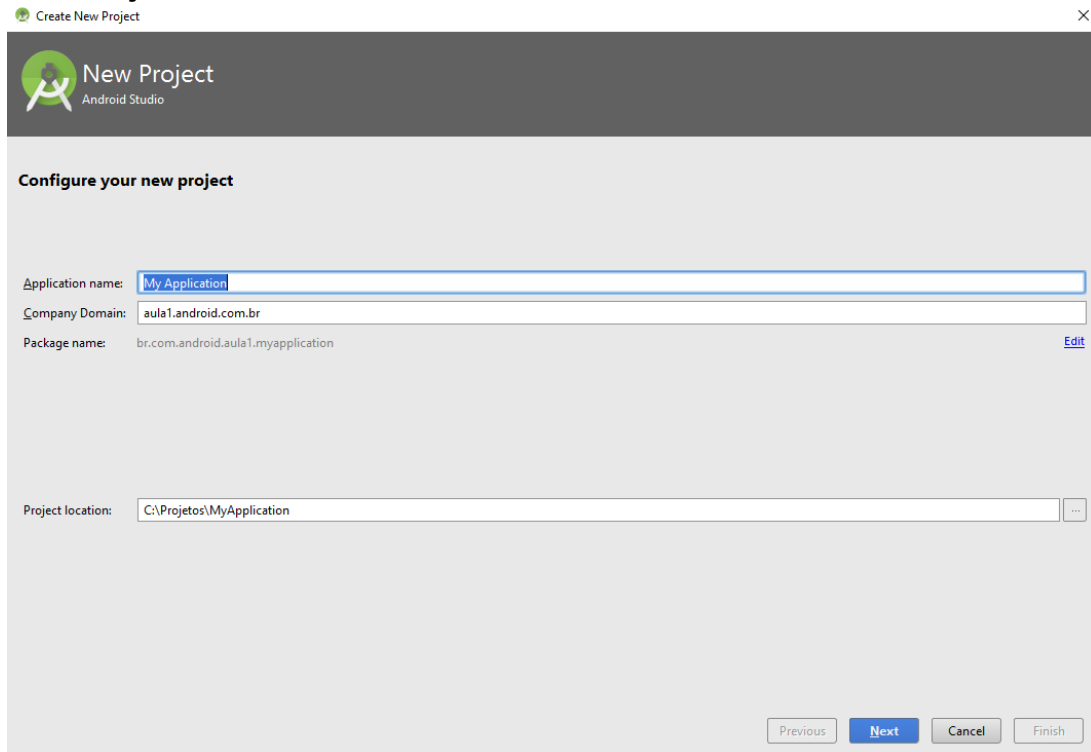
Tela inicial do Android Studio.



Nesta tela, podemos criar um novo projeto Android Studio, abrir um projeto Android Studio existente, buscar atualizações de um projeto da ferramenta de controle de versões, importar um projeto de outro ambiente de desenvolvimento (como o Eclipse ADT), importar um exemplo de projeto do GitHub, além de configurar o ambiente de desenvolvimento e obter acesso à documentos e tutoriais.

Para nosso primeiro projeto, clique na primeira opção: Start a new Android Studio project.

Novo Projeto



Create New Project

New Project
Android Studio

Configure your new project

Application name: My Application

Company Domain: aula1.android.com.br

Package name: br.com.android.aula1.myapplication [Edit](#)

Project location: C:\Projetos\MyApplication

Previous Next Cancel Finish

Nesta tela, somos solicitados a preencher o nome da aplicação, o domínio de internet da companhia e o destino do projeto. Para este exercício, vamos deixar com o valor padrão apresentado. Clique no botão Next para ser direcionado à próxima tela.

Agora, devemos escolher qual será o dispositivo-alvo de nossa aplicação. Lembre-se: o ambiente Android Studio possibilita o desenvolvimento para todas as plataformas suportadas pelo Android, permitindo inclusive que um mesmo projeto seja portado entre as plataformas.

Para nosso exemplo, utilizaremos a plataforma de telefone e tablet. Selecione a primeira opção.

Plataforma-alvo.

Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK: API 15: Android 4.0.3 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately 97,4% of the devices that are active on the Google Play Store.
[Help me choose](#)

☐ Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

☐ Android Auto

☐ Glass

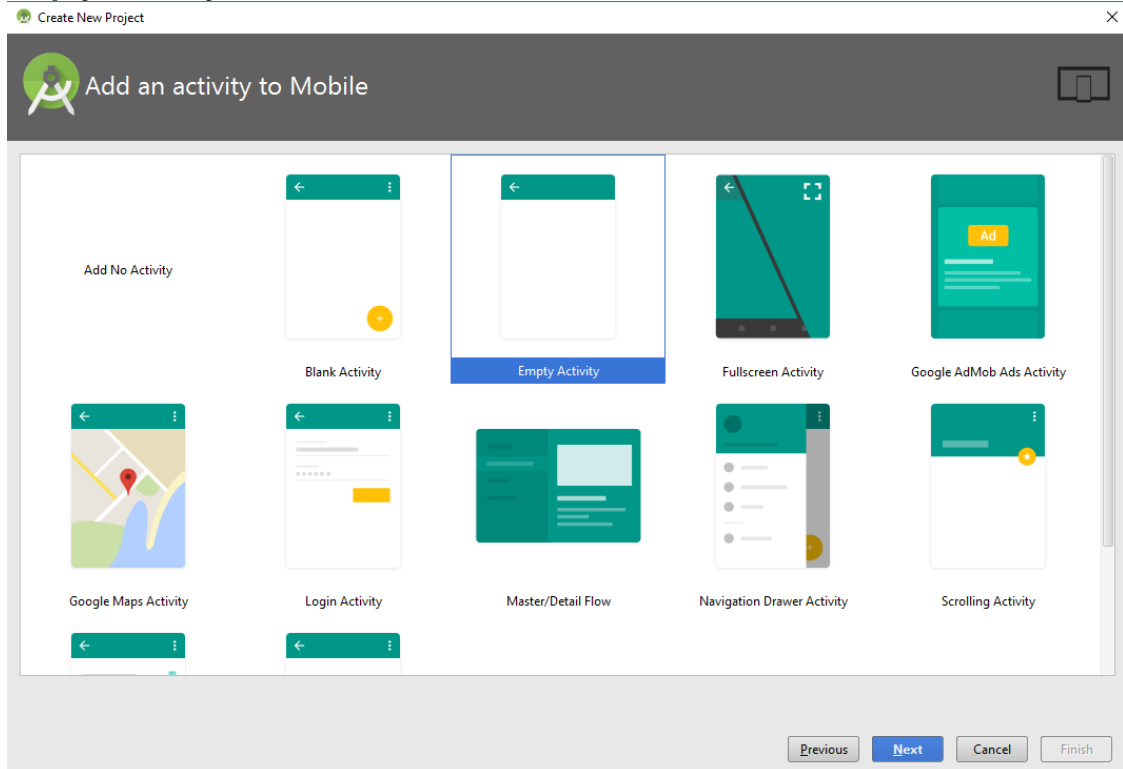
Minimum SDK: Glass Development Kit Preview

Previous Next Cancel Finish

Ao clicarmos no botão Next, somos apresentados à tela de templates de Activities, onde podemos definir qual será nosso primeiro modelo de Activity para este projeto. Podemos criar, neste momento, um projeto sem Activity, com uma Activity completamente em branco, uma Activity vazia, em tela cheia, entre muitas outras.

Para nosso primeiro exemplo, selecione a opção Empty Activity e clique no botão Next para ser redirecionado à próxima tela.

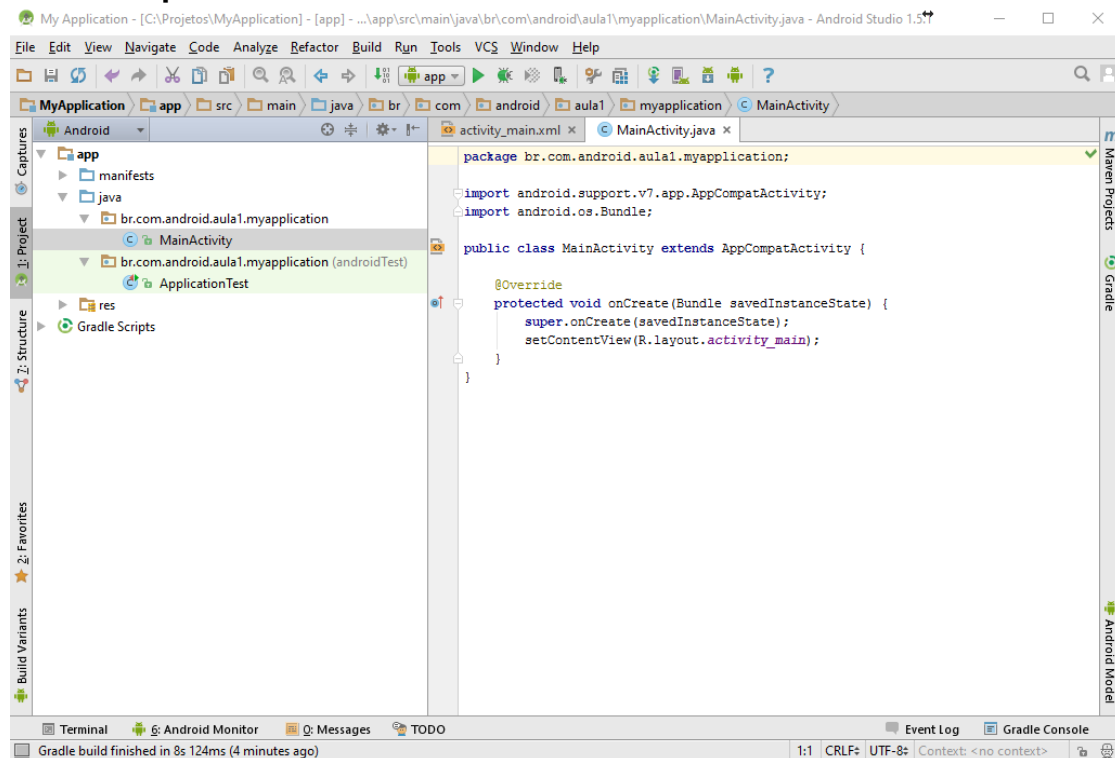
Empty Activity.



Como optamos, na tela anterior, pela criação de uma Activity vazia (Empty Activity), somos, agora, convidados a customizar a mesma, fornecendo informações como nome da Activity e nome do arquivo de Layout desta. Mais uma vez, deixaremos os valores-padrão. Clique no botão Finish para dar início à construção de nosso projeto, com base nas opções que viemos customizando até o momento. A operação de customização pode levar algum tempo até que seja finalizado, portanto tenha paciência.

Ao término da customização, a tela principal do Android Studio será aberta com o template de nossa Activity, pronta para o início de desenvolvimento.

Tela Principal.



Vamos, então, analisar o código gerado automaticamente:

1. **package** br.com.android.aula1.myapplication
2. **import** android.support.v7.app.AppCompatActivity
3. **import** android.os.Bundle
4. **public class** MainActivity **extends** AppCompatActivity {
5. **@Override**
6. **protected void** onCreate(Bundle savedInstanceState) {
7. **super.onCreate**(savedInstanceState);
8. **setContentView**(R.layout.**activity_main**);
9. }
10. }

Na primeira linha, temos o nome do pacote onde encontra-se a classe que foi criada automaticamente, no caso, nossa classe MainActivity.

```
package br.com.android.aula1.myapplication
```

Na segunda e terceira linhas, temos a importação dos pacotes necessários à compilação deste projeto. Na segunda linha temos o pacote de retrocompatibilidade para a ActionBar em dispositivos que utilizem a API level 7 (Eclair) e superior, enquanto na terceira linha temos o pacote que nos permite enviar dados mapeados através de uma chave (String). O Bundle (grupo de objetos agrupados) é utilizado normalmente para a transmissão de dados entre várias atividades.

```
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

Na linha 4, temos a definição de nossa classe (MainActivity), informando que trata-se de uma classe com encapsulamento público e que herda (extends) os atributos e métodos da classe AppCompatActivity.

```
public class MainActivity extends AppCompatActivity {
```

Na linha 5, informamos que o método a seguir, criado na classe AppCompatActivity, deve ser redefinido por nossa classe MainActivity.

```
@Override3
```

Na linha 6, definimos que o método a ser redefinido será o referente ao evento onCreate (que faz parte do ciclo de vida de uma Activity).

```
protected void onCreate(Bundle savedInstanceState) {
```


Na linha 7, informamos que, antes da redefinição do método, devemos repassar à classe antecessora (AppCompatActivity) os valores que nos foram informados no argumento savedInstanceState.

```
super.onCreate(savedInstanceState);
```

Na linha 8, finalmente, damos início à redefinição deste método, no caso, definindo que o ContentView a ser utilizado por esta Activity utilizará o layout designado pelo arquivo R.layout.activity_main

```
setContentView(R.layout.activity_main);
```

Ficou confuso? Não se preocupe! Estes são termos que se tornarão muito comuns durante o desenvolvimento para Android, e a familiaridade com eles aumentará à medida que nossas aulas forem se desenrolando.

FINALIZANDO

Nesta aula foram apresentados os conceitos de introdução ao desenvolvimento de aplicativos para Android, sua arquitetura e também seu ambiente oficial de desenvolvimento.

Referências

Imagens

Figura 10 – Logo Android Studio.
<http://tools.android.com/_/rsrc/1460493325829/config/customLogo.gif?revision=1>. Acessado em: 30 jul. 2016.

Figura 11 – WYSIWYG. <https://developer.android.com/studio/images/new-project-wizard-final-results_2-1_2x.png>. Acessado em: 30 jul. 2016.

Figura 12 – Android Device.
<https://developer.android.com/studio/images/new-project-wizard-devices_2-1_2x.png>. Acessado em: 30 jul. 2016.

Figura 13 – Tela principal do Android Studio.
<https://developer.android.com/studio/images/intro/main-window_2-1_2x.png>. Acessado em: 30 jul. 2016.

Figura 7 – Activity.
<<https://developer.android.com/design/material/images/MaterialDark.png>>. Acessado em: 30 jul. 2016.

Figura 8 – Contacts. <<https://developer.android.com/sdk/images/4.0/contact-faves-lg.png>>. Acessado em: 30 jul. 2016.

Figura 9 – Notificação.
<https://developer.android.com/wear/images/notif_summary_framed.png>. Acessado em: 30 jul. 2016.

Figuras 1, 2, 3, 4, 5, 6.
<<https://upload.wikimedia.org/wikipedia/commons/a/af/Android-System-Architecture.svg>>. Acessado em: 30 jul. 2016.

Livros

DEITEL, Paul; DEITEL, Harvey; OSWALD, Alexander. **Android 6 para Programadores** – Uma Abordagem Baseada em Aplicativos. Boockman: 2015.

Sites

ANDROID. **Android 5.0 Lollipop.** Disponível em:
<https://www.android.com/intl/pt-BR_br/history/>. Acessado em: 30 jul. 2016.

ANDROID. **Android Studio.** Disponível em:
<<https://developer.android.com/studio/intro/index.html>>. Acessado em: 30 jul. 2016.

ANDROID. **Fundamentos de desenvolvimento android.** Disponível em:
<<https://developer.android.com/guide/components/fundamentals.html>>.
Acessado em: 30 jul. 2016.

ANDROLID. **Provedores de conteúdo.** Disponível em:
<<https://developer.android.com/reference/android/provider/package-summary.html>>. Acessado em: 30 jul. 2016.

WIKIPEDIA. **Pacote de Aplicações Android.** Disponível em:
<https://en.wikipedia.org/wiki/Android_application_package>. Acessado em: 30 jul. 2016.