

Conheça o Android Studio

O Android Studio é o Ambiente de desenvolvimento integrado (IDE, na sigla em inglês) oficial para o desenvolvimento de apps Android e é baseado no IntelliJ IDEA (<https://www.jetbrains.com/idea/>). Além do editor de código e das ferramentas de desenvolvedor avançadas do IntelliJ, o Android Studio oferece ainda mais recursos para aumentar sua produtividade na compilação de apps Android, como:

- Um sistema de compilação flexível baseado em Gradle
- Um emulador rápido com inúmeros recursos
- Um ambiente unificado que possibilita o desenvolvimento para todos os dispositivos Android
- A aplicação de alterações para enviar alterações de código e recursos ao aplicativo em execução sem reiniciar o aplicativo
- Modelos de código e integração com GitHub para ajudar a criar recursos comuns de apps e importar exemplos de código
- Frameworks e ferramentas de teste cheios de possibilidades
- Ferramentas de lint para detectar problemas de desempenho, usabilidade, compatibilidade com versões, entre outros
- Compatibilidade com C++ e NDK
- Compatibilidade integrada com o Google Cloud Platform (<https://cloud.google.com/tools/android-studio/docs/>), facilitando a integração do Google Cloud Messaging e do App Engine.

Esta página traz uma introdução aos recursos básicos do Android Studio. Para ver um resumo das alterações mais recentes, consulte Notas da versão do Android Studio ([/studio/releases/index.html](https://studio/releases/index.html)).

Estrutura do projeto

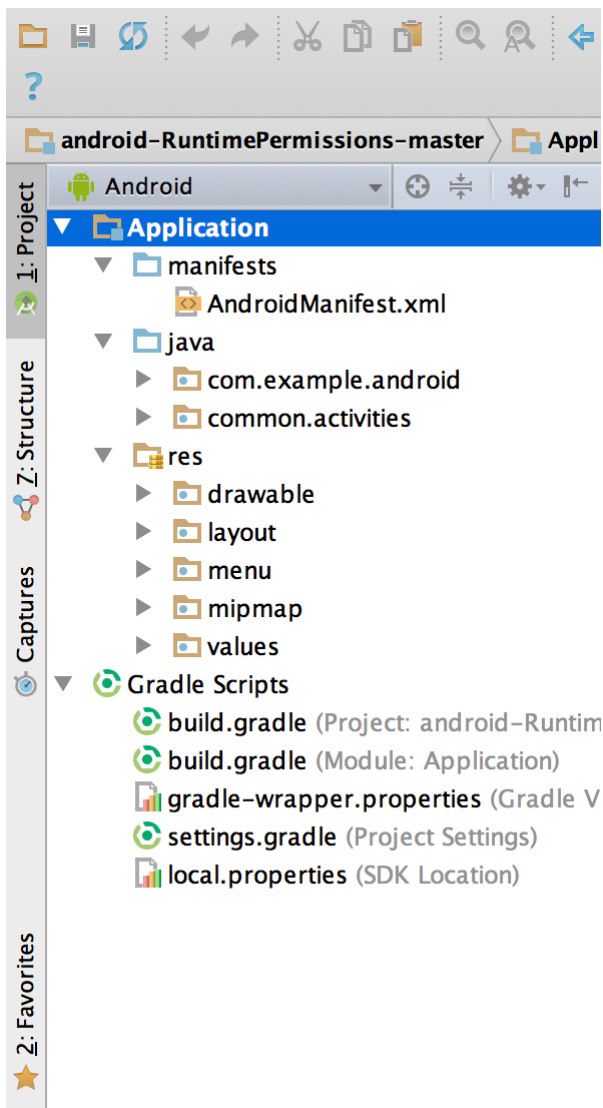


Figura 1. Arquivos do projeto na visualização do Android.

Cada projeto no Android Studio contém um ou mais módulos com arquivos de código-fonte e de recursos. Os tipos de módulos incluem:

- Módulos de apps Android
- Módulos de biblioteca
- Módulos do Google App Engine

Por padrão, o Android Studio exibe os arquivos do projeto na visualização de projetos Android, como mostrado na figura 1. Essa visualização é organizada por módulos para permitir o acesso rápido aos principais arquivos de origem do projeto.

Todos os arquivos de compilação podem ser vistos no nível superior em **Gradle Scripts**, e cada módulo de app contém as pastas a seguir:

- **Manifestos:** contém o arquivo `AndroidManifest.xml`

- **Java:** contém os arquivos de código-fonte do Java, incluindo o código de teste do JUnit
- **Recursos:** contém todos os recursos que não são código, como layouts XML, strings de IU e imagens em bitmap

A estrutura do projeto Android em disco difere dessa representação simplificada. Para ver a estrutura de arquivos real do projeto, selecione **Project** na lista suspensa **Project** (exibida na figura 1 como **Android**).

Também é possível personalizar a visualização dos arquivos do projeto para se concentrar em aspectos específicos do desenvolvimento do app. Por exemplo, a escolha da visualização **Problems** do projeto exibe links para os arquivos de origem que contêm erros reconhecidos de programação e sintaxe, como a falta de uma tag de fechamento de elemento XML em um arquivo de layout.

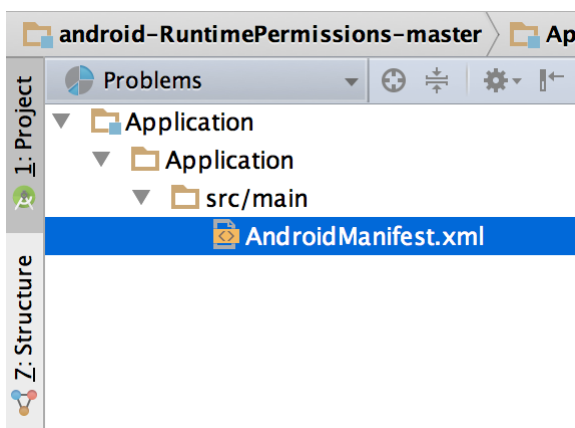


Figura 2. Arquivos do projeto na visualização "Problems", mostrando um arquivo de layout com problema.

Para ver mais informações, consulte [Visão geral de projetos \(/studio/projects/index.html\)](/studio/projects/index.html).

A interface do usuário

A janela principal do Android Studio é composta de diversas áreas lógicas, identificadas na figura 3.

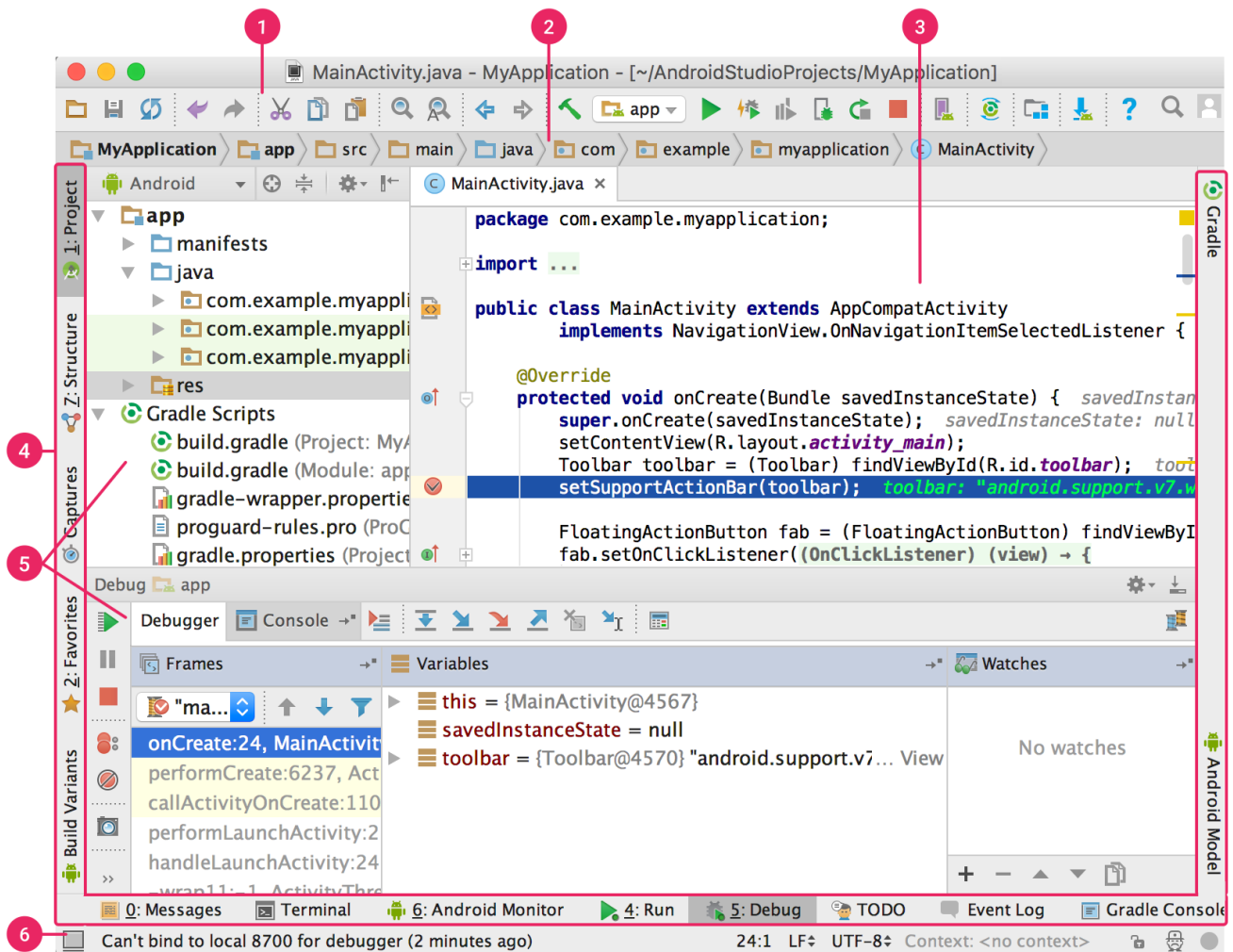


Figura 3. Janela principal do Android Studio.


- 1 A **barra de ferramentas** permite realizar diversas ações, inclusive executar apps e inicializar ferramentas do Android.
- 2 A **barra de navegação** ajuda a navegar pelo projeto e a abrir arquivos para edição. Ela oferece uma visualização mais compacta da estrutura visível na janela **Project**.
- 3 A **janela do editor** é onde você cria e modifica o código. Dependendo do tipo de arquivo atual, o editor pode mudar. Por exemplo, ao visualizar um arquivo de layout, o editor abre o Editor de layout.
- 4 A **barra de janela de ferramentas** fica fora da janela do ambiente de desenvolvimento integrado e contém os botões que permitem expandir ou recolher as janelas de cada ferramenta.
- 5 As **janelas de ferramentas** permitem acessar tarefas específicas, como gerenciamento de projetos, pesquisa e controle de versões, entre outras. As janelas podem ser expandidas e recolhidas.
- 6 A **barra de status** exibe o status do projeto e do próprio ambiente de desenvolvimento integrado, bem como todos os avisos ou mensagens.

Você pode organizar a janela principal para ver mais espaço na tela ocultando ou movendo barras e janelas de ferramentas. Também é possível usar atalhos de teclado para acessar a maioria dos recursos do ambiente de desenvolvimento integrado.

A qualquer momento, você pode pesquisar o código-fonte, bancos de dados, ações, elementos da interface do usuário, entre outros, pressionando duas vezes a tecla "Shift" ou clicando na lupa no canto superior direito da janela do Android Studio. Isso pode ser muito útil quando, por exemplo, você quer localizar uma determinada ação do ambiente de desenvolvimento integrado e esqueceu a forma de acionamento.

Janelas de ferramentas

Em vez de usar perspectivas predefinidas, o Android Studio segue o contexto e exibe automaticamente as janelas de ferramentas relevantes de acordo com o trabalho. Por padrão, as janelas de ferramentas mais comuns são fixadas na barra de janelas de ferramentas, nas bordas da janela de aplicativos.

- Para expandir ou recolher uma janela de ferramentas, clique no nome da ferramenta na barra de janelas de ferramentas. Também é possível arrastar, fixar, desafixar, anexar e desanexar janelas de ferramentas.
- Para voltar ao layout padrão atual da janela de ferramentas, clique em **Window > Restore Default Layout** ou personalize o layout padrão clicando em **Window > Store Current Layout as Default**.
- Para mostrar ou ocultar toda a barra de janelas de ferramentas, clique no ícone de janela , no canto inferior esquerdo da janela do Android Studio.
- Para localizar uma janela de ferramentas específica, passe o cursor sobre o ícone de janela e selecione-a no menu.

Também é possível usar atalhos de teclado para abrir janelas de ferramentas. A tabela 1 lista os atalhos para as janelas mais comuns.

Tabela 1. Atalhos de teclado para algumas janelas de ferramentas úteis.

Janela de ferramentas	Windows e Linux	Mac
Projeto	Alt+1	Command+1
Controle de versões	Alt+9	Command+9
Executar	Shift+F10	Control+R

Depurar	Shift+F9	Control+D
Logcat	Alt+6	Command+6
Voltar ao editor	Esc	Esc
Ocultar todas as janelas de ferramentas	Control+Shift+F12	Command+Shift+F12

Caso queira ocultar todas as barras de ferramentas, janelas de ferramentas e guias do editor, clique em **View > Enter Distraction Free Mode**. O *Distraction Free Mode* será ativado. Para sair do "Distraction Free Mode", clique em **View > Exit Distraction Free Mode**.

Você pode usar *Speed Search* para pesquisar e filtrar dentro da maioria das janelas de ferramentas no Android Studio. Para usar "Speed Search", selecione a janela de ferramentas e digite a consulta de pesquisa.

Para ver mais dicas, consulte [Atalhos de teclado](/studio/intro/keyboard-shortcuts.html) (/studio/intro/keyboard-shortcuts.html).

Preenchimento automático de código

O Android Studio tem três tipos de preenchimento automático de código, que podem ser acessados usando atalhos de teclado.

Tabela 2. Atalhos de teclado para preenchimento automático de código.

Tipo	Descrição	Windows e Linux	Mac
Preenchimento básico	Exibe sugestões básicas para variáveis, tipos, métodos, expressões, entre outros. Se você chamar o preenchimento básico duas vezes seguidas, verá mais resultados, incluindo membros privados e membros estáticos não importados.	Control+Space	Control+Space
Preenchimento inteligente	Exibe opções relevantes de acordo com o contexto. O preenchimento inteligente detecta os tipos e os fluxos de dados esperados. Se você chamar o preenchimento inteligente duas vezes seguidas, verá mais resultados, incluindo cadeias.	Control+Shift+Space	Control+Shift+Space
Preenchimento de declaração	Preenche a declaração atual, adicionando parênteses, colchetes e chaves ausentes, formatação etc.	Control+Shift+Enter	Shift+Command+Enter

Também é possível realizar correções rápidas e mostrar as ações de intenção pressionando **Alt+Enter**.

Encontrar exemplos de código

O Buscador de exemplos de código do Android Studio ajuda a encontrar amostras de código Android de alta qualidade oferecidos pelo Google de acordo com o símbolo em destaque no momento no seu projeto. Para saber mais, acesse [Encontrar exemplos de código](/studio/write/sample-code.html) (/studio/write/sample-code.html).

Navegação

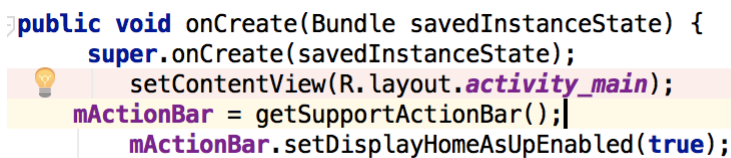
Veja a seguir algumas dicas de navegação no Android Studio.

- Alterne entre os arquivos acessados recentemente com a ação *Recent Files*. Pressione **Control+E** (No Mac, **Command+E**) para chamar a ação "Recent Files". Por padrão, o último arquivo acessado é selecionado. Também é possível acessar qualquer janela de ferramentas por meio da coluna esquerda dessa ação.
- Veja a estrutura do arquivo atual com a ação *File Structure*. Chame a ação "File Structure" pressionando **Control+F12** (No Mac, **Command+F12**). Essa ação permite navegar rapidamente para qualquer parte do arquivo atual.
- Pesquise e navegue para uma classe específica no projeto com a ação *Navigate to Class*. Chame a ação pressionando **Control+N** (no Mac, **Command+O**). A ação "Navigate to Class" é compatível com expressões sofisticadas, incluindo maiúsculas intermediárias (camel humps), caminhos, navegar para linha e correspondência do nome do meio, entre muitas outras. Se você chamar a ação duas vezes seguidas, ela mostrará os resultados das classes do projeto.
- Navegue para um arquivo ou pasta com a ação *Navigate to File*. Chame a ação "Navigate to File" pressionando **Control+Shift+N** (no Mac, **Command+Shift+O**). Para pesquisar pastas em vez de arquivos, adicione uma / ao final da expressão.
- Navegue para um método ou campo por nome com a ação *Navigate to Symbol*. Chame a ação "Navigate to Symbol" pressionando **Control+Shift+Alt+N** (no Mac, **Command+Shift+Alt+O**).
- Encontre todos os fragmentos do código que referenciam a classe, método, campo, parâmetro ou declaração na posição atual do cursor pressionando **Alt+F7** (no Mac, **Option+F7**).

Estilo e formatação

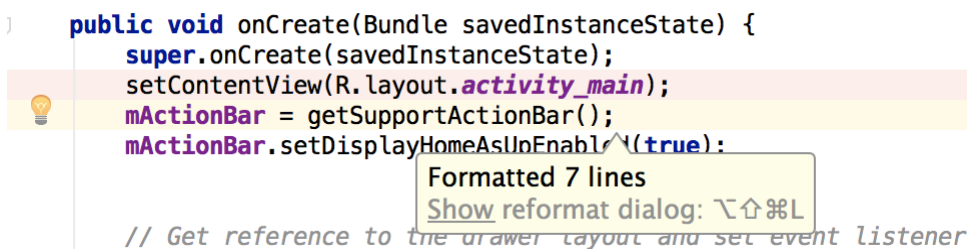
Durante a edição, o Android Studio aplica automaticamente formatação e estilos, conforme especificado nas configurações de estilo do código. Você pode personalizar as configurações de estilo do código de acordo com a linguagem de programação, incluindo a especificação de convenções para tabulação e indentação, espaços, quebras de linha, chaves e linhas em branco. Para personalizar suas configurações de estilo de código, clique em **File > Settings > Editor > Code Style** (no Mac, **Android Studio > Preferences > Editor > Code Style**).

Embora o ambiente de desenvolvimento integrado aplique automaticamente a formatação durante a edição, também é possível chamar explicitamente a ação *Reformat Code* pressionando **Control+Alt+L** (no Mac, **Opt+Command+L**) ou recuar automaticamente todas as linhas pressionando **Control+Alt+I** (no Mac, **Alt+Option+I**).



```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
}
```

Figura 4. Código antes da formatação.



```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
    // Get reference to the drawer layout and set event listener
}
```

Figura 5. Código depois da formatação.

Conceitos básicos do controle de versões

O Android Studio é compatível com diversos sistemas de controle de versões (VCS, na sigla em inglês), entre eles Git, GitHub, CVS, Mercurial, Subversion e Google Cloud Source Repositories.

Depois de importar o app para o Android Studio, use as opções do menu "VCS" do Android Studio para ativar a compatibilidade do VCS com o sistema de controle de versões em questão, criar um repositório, importar os novos arquivos para o controle de versões e realizar outras operações de controle de versões:

1. No menu **VCS** do Android Studio, clique em **Enable Version Control Integration**.

2. No menu suspenso, selecione o sistema de controle de versões a ser associado à raiz do projeto e clique em **OK**.

Agora, o menu "VCS" exibe diversas opções de controle de versões, de acordo com o sistema selecionado.

Observação: também é possível usar a opção de menu **File > Settings > Version Control** para definir e modificar as configurações do controle de versão.

Sistema de compilação Gradle

O Android Studio usa o Gradle como o sistema de compilação de base, com outros recursos específicos do Android disponibilizados pelo plug-in do Android para o Gradle (</studio/releases/gradle-plugin.html>). Esse sistema de compilação é executado como uma ferramenta integrada no menu do Android Studio e de forma independente na linha de comando. Você pode usar os recursos do sistema de compilação para fazer o seguinte:

- Personalizar, configurar e ampliar o processo de programação.
- Criar diversos APKs para seu app com diferentes recursos usando o mesmo projeto e os mesmos módulos.
- Reutilizar código e recursos nos conjuntos de origem.

A flexibilidade do Gradle permite que você faça tudo isso sem modificar os arquivos de origem principais do seu app. O nome dos arquivos de compilação do Android Studio é `build.gradle`. Esses são arquivos de texto simples que usam a sintaxe do Groovy (<http://groovy-lang.org>) para configurar a compilação com elementos oferecidos pelo plug-in do Android para o Gradle. Cada projeto tem um arquivo de compilação de nível superior para todo o projeto e arquivos de compilação de módulo separados para cada módulo. Quando você importa um projeto existente, o Android Studio gera automaticamente os arquivos de compilação necessários.

Para saber mais sobre o sistema de compilação e como configurá-lo, consulte Configurar sua compilação (</studio/build/index.html>).

Variantes de compilação

O sistema de compilação pode ajudar a criar versões diferentes do mesmo aplicativo a partir de um único projeto. Isso é útil quando existe uma versão gratuita e uma versão paga

do app ou quando você quer distribuir vários APKs para configurações de dispositivo diferentes no Google Play.

Para ver mais informações sobre a configuração de variantes de compilação, consulte [Configurar variantes de compilação](/studio/build/build-variants.html) (/studio/build/build-variants.html).

Compatibilidade com vários APKs

A compatibilidade com vários APKs permite a criação eficiente de vários APKs com base na densidade de tela ou no ABI. Por exemplo, é possível criar APKs separados de um app para as densidades de tela hdpi e mdpi, considerando-os ao mesmo tempo uma variante única, além de permitir que eles compartilhem configurações de APK de teste, javac, dx e ProGuard.

Para ver mais informações sobre a compatibilidade com vários APKs, leia [Compilar vários APKs](/studio/build/configure-apk-splits.html) (/studio/build/configure-apk-splits.html).

Redução de recursos

A redução de recursos no Android Studio remove automaticamente os recursos não utilizados do app empacotado e das dependências de biblioteca. Por exemplo, se o aplicativo estiver usando o [Google Play Services](https://developers.google.com/android/guides/overview) (https://developers.google.com/android/guides/overview) para acessar a funcionalidade do Drive e você não estiver usando o [Login do Google](/training/sign-in/index.html) (/training/sign-in/index.html) no momento, a redução poderá remover os diversos recursos drawable dos botões `SignInButton`.

Observação: a redução de recursos é utilizada em conjunto com ferramentas de redução de código, como o ProGuard.

Para ver mais informações sobre a redução de código e recursos, consulte [Reduzir código e recursos](/studio/build/shrink-code.html) (/studio/build/shrink-code.html).

Gerenciamento de dependências

As dependências do seu projeto são especificadas por nome no arquivo `build.gradle`. O Gradle se encarrega de encontrar as dependências e disponibilizá-las na compilação. Você pode declarar dependências de módulo e também dependências binárias remotas e locais no arquivo `build.gradle`. O Android Studio configura os projetos para usarem o Maven Central Repository por padrão. Essa configuração está incluída no arquivo de compilação

de nível superior do projeto. Para ver mais informações sobre como configurar dependências, leia [Adicionar dependências de compilação](/studio/build/dependencies.html) (/studio/build/dependencies.html).

Ferramentas de depuração e criação de perfil

O Android Studio ajuda a depurar e a melhorar o desempenho do código, o que inclui ferramentas de depuração em linha e análise de desempenho.

Depuração em linha

Use a depuração em linha para melhorar o acompanhamento do código na visualização do depurador com a verificação em linha de referências, expressões e valores de variáveis. As informações de depuração em linha incluem:

- Valores de variáveis em linha
- Objetos de referência que referenciam um objeto selecionado
- Valores de retorno de métodos
- Expressões de lambda e de operadores
- Valores de dicas

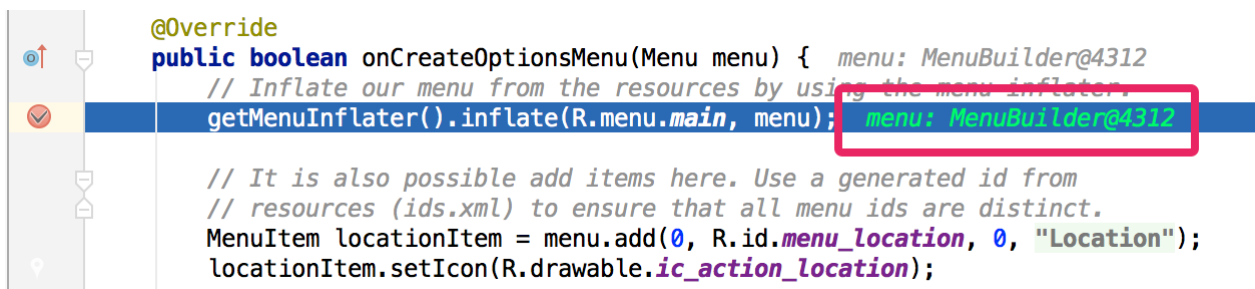


Figura 6. Valor de uma variável em linha.

Para ativar a depuração in-line, na janela **Debug**, clique em **Settings**



e selecione a caixa de seleção de **Show Values Inline**.

Criadores de perfis de desempenho

O Android Studio oferece criadores de perfis de desempenho para que você acompanhe mais facilmente o uso de memória e CPU do app, encontre objetos desalocados, localize vazamentos de memória, otimize o desempenho de gráficos e analise solicitações de rede. Com seu app em execução em um dispositivo ou emulador, abra a guia **Android Profiler**.

Para ver mais informações sobre criadores de perfis de desempenho, consulte [Ferramentas de criação de perfis de desempenho](/studio/profile/index.html) (/studio/profile/index.html).

Despejo de heap

Durante o monitoramento do uso de memória no Android Studio, você pode iniciar a coleta de lixo e, ao mesmo tempo, despejar o heap do Java de um instantâneo de alocação heap em um arquivo de formato binário HPROF específico do Android. O visualizador de HPROF exibe classes, instâncias de cada classe e uma árvore de referência para ajudar a acompanhar o uso de memória e localizar vazamentos.

Para ver mais informações sobre como trabalhar com despejos de heap, consulte [Analisar o heap e as alocações](/studio/profile/memory-profiler.html) (/studio/profile/memory-profiler.html).

Memory Profiler

Você pode usar o Memory Profiler para rastrear a alocação de memória e observar onde os objetos são alocados quando você realiza determinadas ações. O conhecimento dessas alocações permite otimizar o desempenho e o uso da memória do app por meio do ajuste das chamadas de métodos relacionadas a essas ações.

Para ver informações sobre como rastrear e analisar alocações, consulte [Analisar o heap e as alocações](/studio/profile/memory-profiler.html) (/studio/profile/memory-profiler.html).

Acesso a arquivos de dados

As ferramentas do Android SDK, como [Systrace](/studio/profile/systrace-commandline.html) (/studio/profile/systrace-commandline.html) e [logcat](/studio/debug/am-logcat.html) (/studio/debug/am-logcat.html), geram dados de desempenho e depuração para a análise detalhada de aplicativos.

Para visualizar os arquivos de dados gerados que estão disponíveis, abra a janela de ferramentas "Captures". Na lista de arquivos gerados, clique duas vezes em um arquivo para exibir os dados. Clique com o botão direito do mouse em qualquer arquivo `.hprof` para convertê-lo ao formato padrão para [investigar o uso de RAM](/studio/profile/investigate-ram.html) (/studio/profile/investigate-ram.html).

Inspeções de código

Sempre que você compila um programa, o Android Studio executa automaticamente inspeções de [lint](/studio/write/lint.html) (/studio/write/lint.html) configuradas e outras [inspeções de ambiente de](#)

desenvolvimento integrado (<https://www.jetbrains.com/help/idea/2019.1/code-inspection.html>) para ajudar a identificar e corrigir problemas na qualidade estrutural do código.

A ferramenta de lint verifica os arquivos de origem do projeto Android para localizar possíveis bugs e melhorias de otimização em relação a critérios de precisão, segurança, desempenho, usabilidade, acessibilidade e internacionalização.

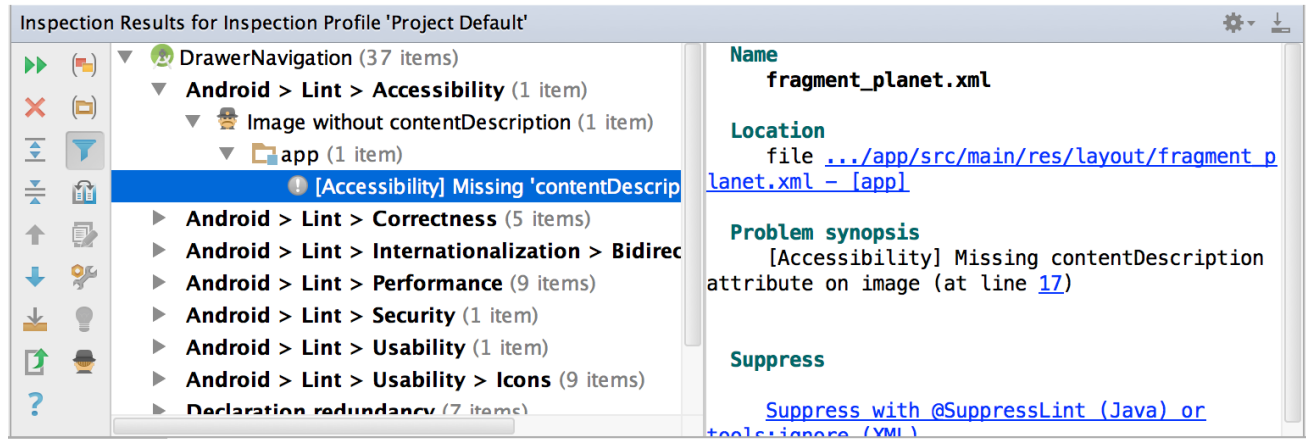


Figura 7. Resultados de uma inspeção de lint no Android Studio.

Além das verificações de lint, o Android Studio também realiza inspeções de código do IntelliJ e valida anotações para otimizar o fluxo de trabalho da programação.

Para ver mais informações, consulte Como aprimorar o código com Lint (</studio/write/lint.html>).

Anotações no Android Studio

O Android Studio é compatível com anotações para variáveis, parâmetros e valores de retorno para ajudar a detectar bugs, como exceções de ponteiros nulos e conflitos de tipos de recursos. O Android SDK Manager empacota a biblioteca Support-Annotations no Android Support Repository para uso com o Android Studio. O Android Studio valida as anotações configuradas durante a inspeção do código.

Para ver mais detalhes sobre as anotações do Android, consulte Aprimoramento da inspeção do código com anotações (</studio/write/annotations.html>).

Mensagens de registro

Durante a compilação e execução do app no Android Studio, você pode ver a saída do adb (</studio/command-line/adb.html>) e mensagens de registro do dispositivo na janela Logcat (</studio/debug/am-logcat.html>).

Criação de perfis de desempenho

Para criar perfis de desempenho para CPU, memória e rede do seu app, abra o [Android Profiler](/studio/profile/android-profiler.html) clicando em **View > Tool Windows > Android Profiler**.

Content and code samples on this page are subject to the licenses described in the [Content License](/license).
Java is a registered trademark of Oracle and/or its affiliates.