

Desenvolvendo com qualidade para Android



Rafael Alves Feliciano

fael.anjelus@gmail.com

@rafaelmeteoro

<https://github.com/rafaelmeteoro>

Graduado em Ciência da Computação pela UFJF

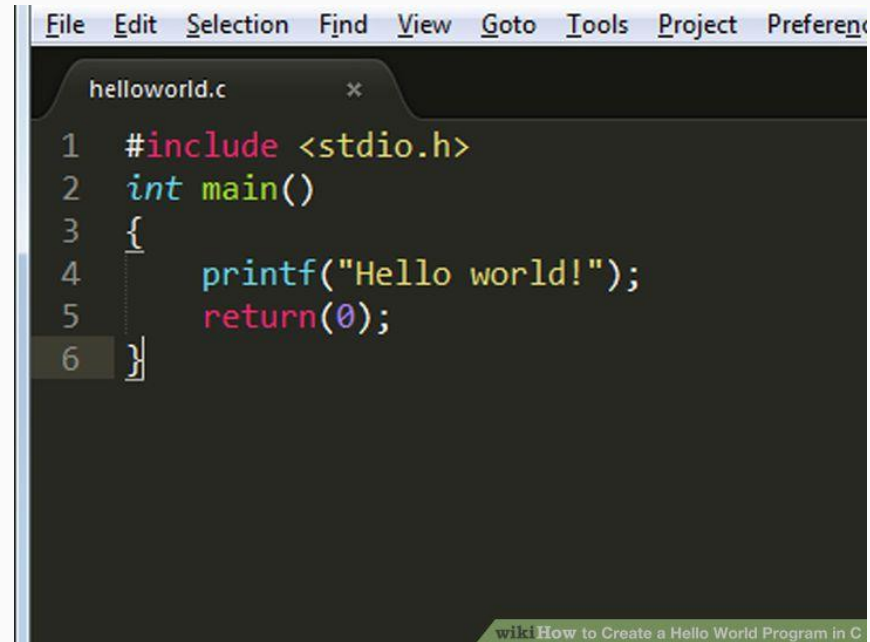
Desenvolvedor Android na Smarti9



O que é preciso para desenvolver um app de qualidade?

Uma breve história...

hello world



A screenshot of a code editor window with a dark theme. The window title is 'helloworld.c'. The menu bar includes 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', and 'Preferences'. The code is as follows:

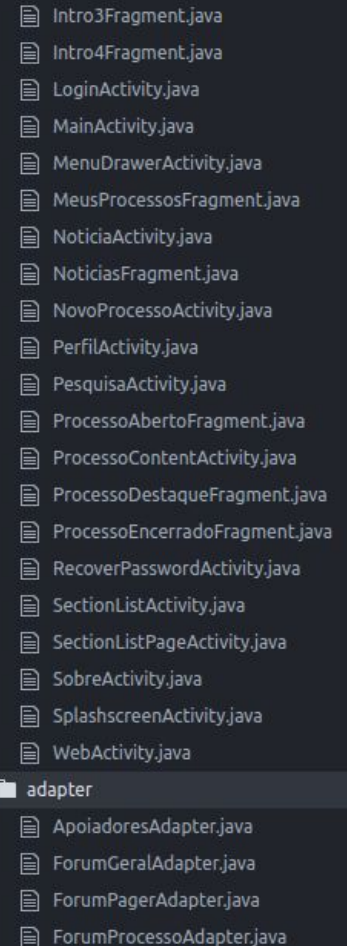
```
1  #include <stdio.h>
2  int main()
3  {
4      printf("Hello world!");
5      return(0);
6  }
```

At the bottom right, there is a green banner with the text 'wikiHow to Create a Hello World Program in C'.

super activitys

```
791
792     @Override
793     protected void onPostExecute(Void aVoid) {
794         super.onPostExecute(aVoid);
795         mSwipeRefreshLayout.setRefreshing(false);
796         updating = false;
797         adapter.setFeatureProcessList(featureProcessList);
798     }
799 }
800 }
801
```

Organização dos arquivos?



A screenshot of a file explorer interface with a dark background. It displays a list of Java files, each preceded by a document icon. The files are: Intro3Fragment.java, Intro4Fragment.java, LoginActivity.java, MainActivity.java, MenuDrawerActivity.java, MeusProcessosFragment.java, NoticiaActivity.java, NoticiasFragment.java, NovoProcessoActivity.java, PerfilActivity.java, PesquisaActivity.java, ProcessoAbertoFragment.java, ProcessoContentActivity.java, ProcessoDestaqueFragment.java, ProcessoEncerradoFragment.java, RecoverPasswordActivity.java, SectionListActivity.java, SectionListPageActivity.java, SobreActivity.java, SplashScreenActivity.java, and WebActivity.java. Below this list is a folder named 'adapter', indicated by a downward arrow and a folder icon. The 'adapter' folder is expanded, showing a sub-list of files: ApoiadoresAdapter.java, ForumGeralAdapter.java, ForumPagerAdapter.java, and ForumProcessoAdapter.java.

- Intro3Fragment.java
- Intro4Fragment.java
- LoginActivity.java
- MainActivity.java
- MenuDrawerActivity.java
- MeusProcessosFragment.java
- NoticiaActivity.java
- NoticiasFragment.java
- NovoProcessoActivity.java
- PerfilActivity.java
- PesquisaActivity.java
- ProcessoAbertoFragment.java
- ProcessoContentActivity.java
- ProcessoDestaqueFragment.java
- ProcessoEncerradoFragment.java
- RecoverPasswordActivity.java
- SectionListActivity.java
- SectionListPageActivity.java
- SobreActivity.java
- SplashscreenActivity.java
- WebActivity.java
- adapter
 - ApoiadoresAdapter.java
 - ForumGeralAdapter.java
 - ForumPagerAdapter.java
 - ForumProcessoAdapter.java

Consequências

- Código desorganizado
- Difícil manutenção
- Não é tão fácil testar
- Adicionar nova funcionalidade também é difícil e pode ocasionar erros em outras partes

E agora?



“Palma, palma, não priemos cânico.”



Soluções

No universo de desenvolvimento, contamos com inúmeras ferramentas para solucionar um problema.

Podemos citar:

Clean Architecture, Model-View-Presenter, Design Patterns, Repository Patterns

Boas práticas

A idéia é simples. Um conjunto de práticas tem como propósito entregar um sistema que seja:

- **Independente de Frameworks.** A arquitetura não depende da existência de alguma biblioteca. Isso permite que você use tais estruturas como ferramentas, em vez de ter que enfiar o sistema em suas restrições limitadas.

Boas práticas

- **Testável.** As regras de negócio devem ser testadas sem depender de uma interface do usuário, banco de dados, servidor Web, ou qualquer agente externo.
- **Independente de UI.** A UI pode mudar facilmente, sem alterar o resto do sistema.
- **Independente do banco de dados.** Você precisa ser capaz de trocar seu banco, do SQLite para PaperDB, RealmDB, ou qualquer outro, sem que suas regras de negócios sejam afetadas durante o processo.

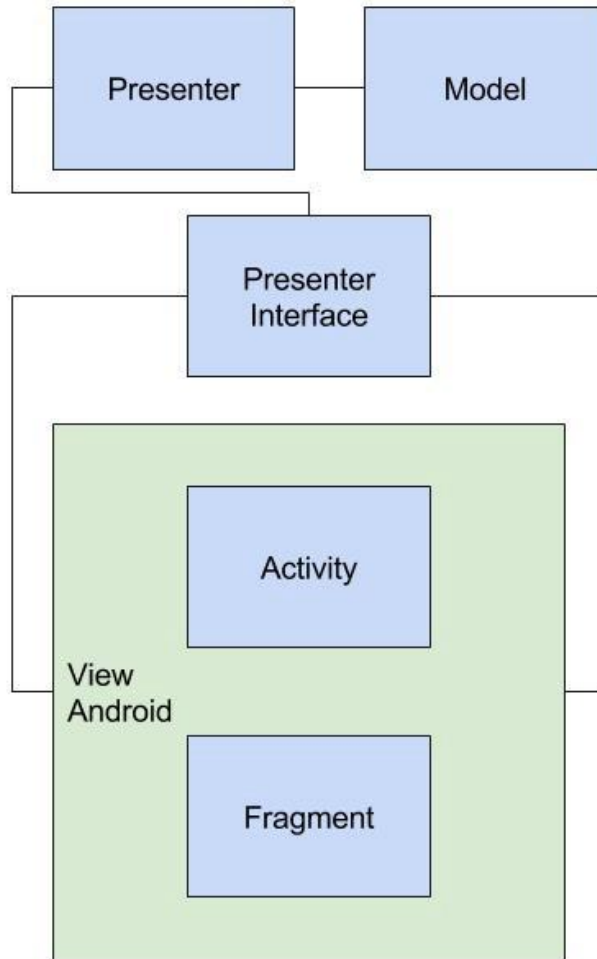
Boas práticas

- **Independente de qualquer agente externo.** Na verdade suas regras de negócios simplesmente não sabem nada sobre o mundo exterior.

Model View Presenter (ou MVP)

MVP é um padrão de apresentação da interface do usuário. Ele separa as responsabilidades em quatro componentes.

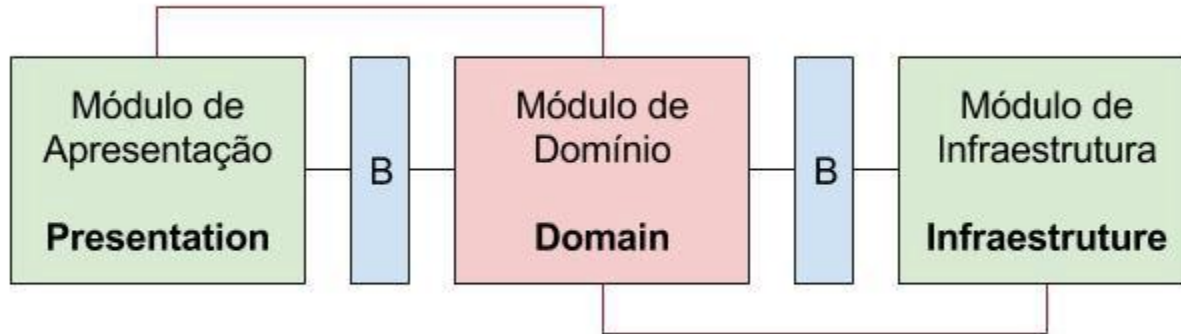
Com o MVP se torna fácil desacoplar a View do Presenter. Isso facilitamos **Instrumentation Test** com a View e nos **Unit Test** com o Presenter.



Clean Architecture

O objetivo é o princípio da responsabilidade única, separando o interesse de cada módulo e mantendo as regras de negócio sem conhecer qualquer detalhe sobre o mundo exterior, assim eles podem ser testados sem dependência de qualquer elemento externo.

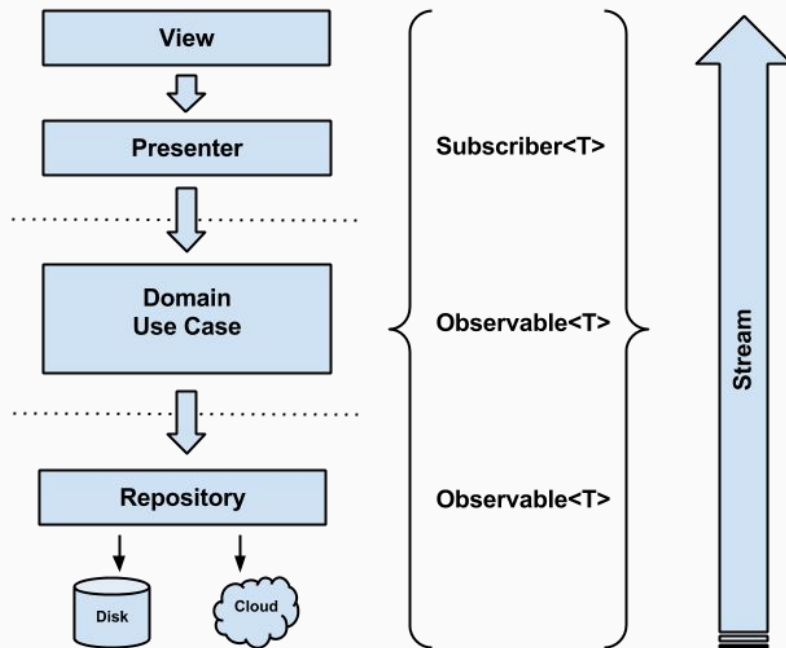
Clean Architecture



Clean Architecture

- **Presentation (módulo Android):** responsável pela interface do aplicativo e a exibição dos dados recebidos do domínio.
- **Domain (módulo Java):** responsável pelas entidades e as regras de domínio específicas do seu projeto. Esse módulo deve ser totalmente independente da plataforma Android.
- **Infraestrutura (módulo Android):** responsável pelo banco de dados, acesso a internet e outros “detalhes” da aplicação.

Clean Architecture



Repository Design Pattern

Em aplicações android, na maioria das vezes você precisa de buscar e manter os dados em algum tipo de armazenamento. Na maioria das vezes será SQLite.

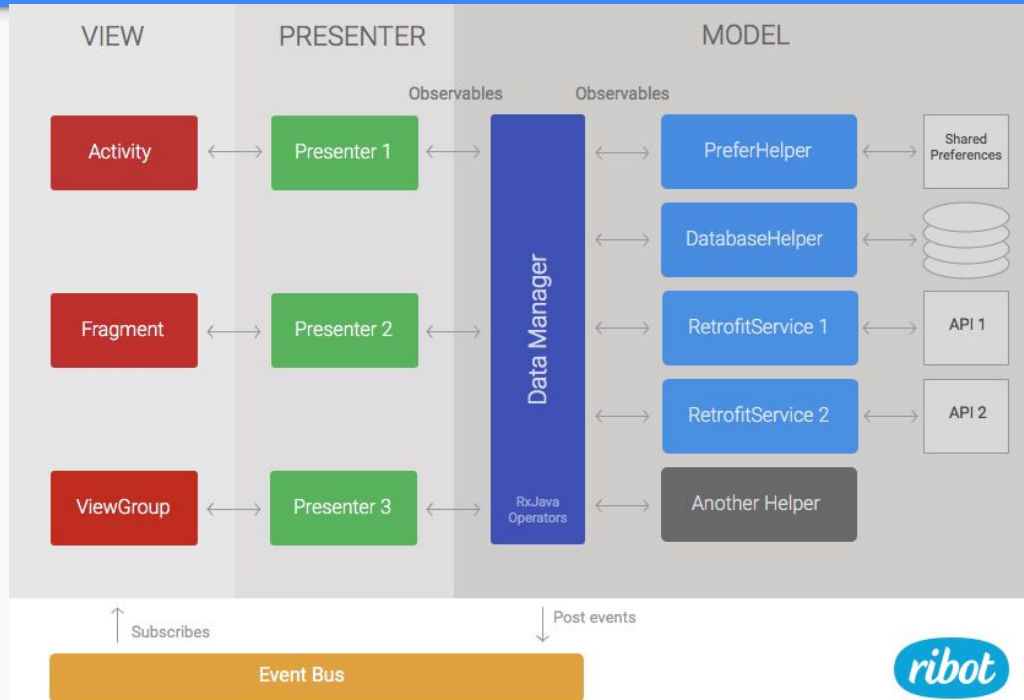
Provavelmente você deseja acessar os dados de forma mais legível e fácil.

Muitas das vezes se utiliza de algum ORM (mapeamento objeto-relacional).

Unindo os Módulos

Para unir todos os módulos e fazer a arquitetura funcionar, geralmente se utiliza o conceito de Inversão de Controle e Injeção de Dependência. No Android tem várias boas alternativas como: AndroidAnnotations, Dagger e RoboGuice.

Arquitetura final



Conceitos?



Código



<https://github.com/rafaelmeteoro/PokedexGDGJF>

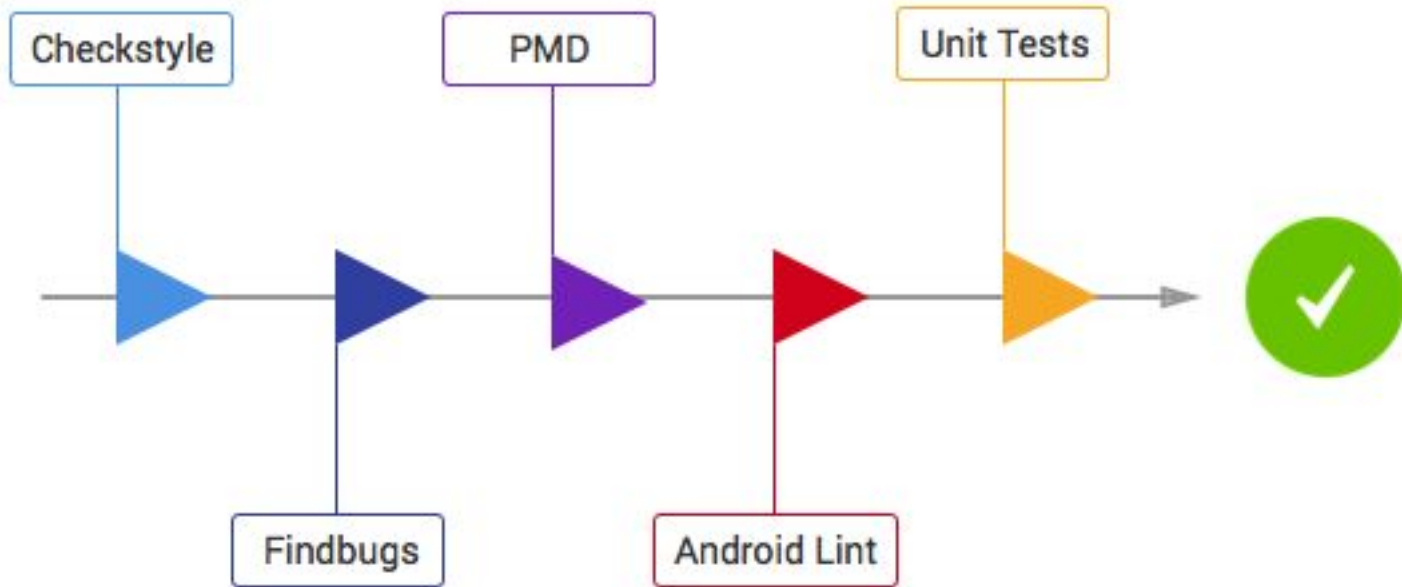


Considerações Finais

Code Quality (ferramentas de análise de código)

- PMD: Ele encontra falhas comuns de programação como variáveis não utilizadas, blocos catch vazio, criação de objetos desnecessários, e assim por diante.
- Findbugs: Esta ferramenta utiliza análise estática para encontrar bugs no código Java.
- Checkstyle: Ele garante que o estilo do código segue as diretrizes do código Android.

Check task



Dicas

devjf.slack.com

androiddevbr.slack.com



Perguntas?



Fontes

- <https://medium.com/android-dev-br/mva-quando-menos-%C3%A9-mais-363f1303bb36#.7ipqvsafj>
- <https://medium.com/android-dev-br/desmistificando-o-mvc-e-mvp-no-android-abe927d01df7#.wi7biykh5>
- <https://medium.com/exploring-android/introducing-bourbon-dribbble-android-mvp-and-a-common-code-module-1d332a4028b5#.ru225ihh2>
- <https://github.com/android10/Android-CleanArchitecture>
- <https://medium.com/android-dev-br/clean-architecture-para-android-eb492513263e#.qsblz6uu7>
-

Fontes

- <https://medium.com/android-dev-br/clean-architecture-para-android-eb492513263e#.qsblz6uu7>
- <https://github.com/ribot/ribot-app-android>
-