OnMuseum

Projeto Arquitetural

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Objetivo

Este documento tem o objetivo de descrever filosofia, decisões, restrições, justificativas, premissas e quaisquer outros aspectos que dão forma ao projeto e sua implementação.

# Metas Arquiteturais e Filosofia

O principal direcionador da escolha foi a necessidade de mobilidade, uma vez que o visitante estará no museu e se faz necessária sua locomoção atravez da exposição; será preciso ultilizar a plataforma nativa pois se faz necessário de itens como a camera e 3G/Wifi. Por questões de usabilidade e para fins auxiliar o aplicativo mobile será criado uma solução web, para administração de dados por parte do museu. O banco de dados será gerenciado pelo PARSE, para garantir uma futura expansibilidade do projeto para outras plataformas, qual usa o protocolo REST, garantindo um baixo tráfego no 3G e uma boa confiabilidade.

# Premissas e Dependências

# Requisitos Críticos da Arquitetura

<http://developer.android.com/index.html>

# Decisões, Restrições e Justificativas

* **Uma página de cadastro web:** para acesso do museu, cadastro das obras, geração do QR Code; garantindo assim a usabilidade do sistema.
* **Utilizar o PARSE para administração da persistência**: garantindo assim uma futura expansibilidade para outras plataformas como Windows fone ou IOS sem precisar migrar os dados.
* **Não usar arquitetura Web para criar a solução:** se fará necessário uso de algumas funcionalidade do dispositivo o que não é possível usando a solução web.

# Mecanismos Arquiteturais

# http://www.dicas-l.com.br/imagens/hello_android_01.jpg

Na camada **Linux Kernel** está localizado o sistema operacional da plataforma, que é baseado no Linux. Ela é responsável por serviços de mais baixo nível da plataforma, como gerenciamento de memória e processos, segurança, etc.

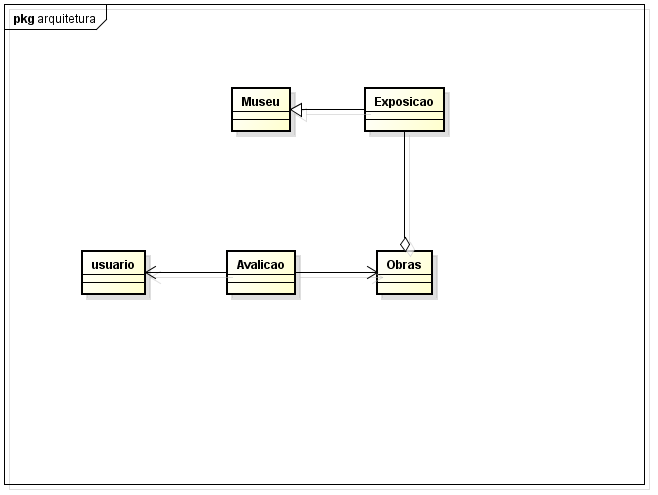
Na camada **Libraries** estão as bibliotecas nativas escritas em C/C++ que fazem parte da plataforma. Estão nesta camada APIs como o OpenGL ES (para renderização 3D), Parse (gerenciador de bancos de dados) e suporte a diversos formatos de áudio e vídeo.

A camada **Android Runtime** dá condições para que as aplicações baseadas na plataforma sejam executadas. Um dos componentes desta camada são as *core libraries*, que disponibilizam uma API Java utilizada para programação (grande parte das funcionalidades encontradas no Java SE estão disponíveis para o Android). Já o outro componente é a *Dalvik Virtual Machine*, que é uma máquina virtual para suporte à execução de aplicações.

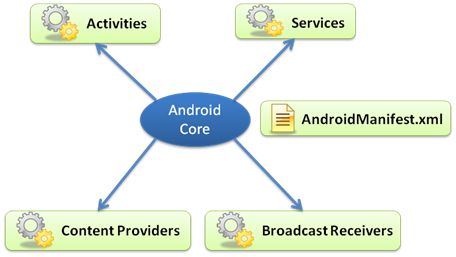
Na camada **Application Framework** ficam as APIs do Android que são normalmente utilizadas pelas aplicações que executam sobre a plataforma. Os gerenciadores de serviços de telefonia, localização e notificação são alguns exemplos do que este framework disponibiliza.

Por fim, a camada **Applications** representa as aplicações que executam sobre a plataforma. Elas podem ser tanto aplicações nativas - como o gerenciador de contatos, navegador, calendário, etc. - como aplicações criadas por terceiros. Aliás, para o Android não existe distinção entre aplicações nativas e aplicações desenvolvidas por outras pessoas, e é esta característica que garante a ela o alto grau de flexibilidade e extensibilidade da plataforma.

# Principais Abstrações



# Camadas do Framework da Arquitetura



**Activities** são as representantes das telas da aplicação. Associada a uma *activity* normalmente existe uma *view*, que define como será feita a exibição visual para o usuário. As *activities* são responsáveis por gerenciar os eventos de tela e também coordenam o fluxo da aplicação.

Os **Services** são códigos que executam em segundo plano. Normalmente são utilizados para tarefas que demandam um grande tempo de execução.

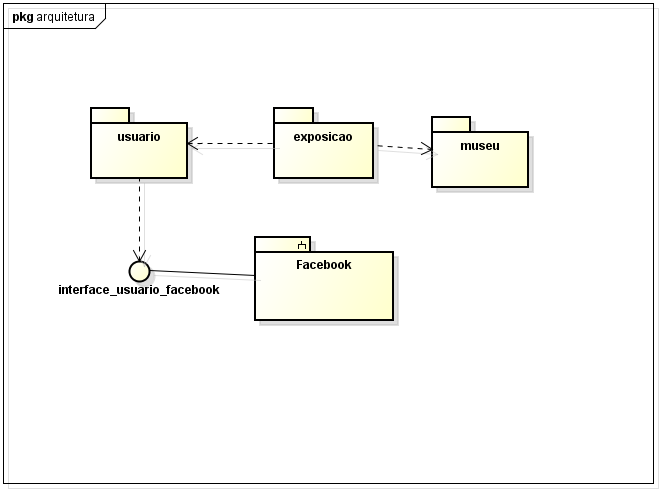
Os **Content Providers** (provedores de conteúdos) são a maneira utilizada pela plataforma para compartilhar dados entre as aplicações que executam no dispositivo. Um exemplo bem claro disto é a aplicação de gerenciamento de contatos do Android, que é nativa. Aplicações desenvolvidas por terceiros podem utilizar um content provider a fim de ler os contatos armazenados no dispositivo de forma simples.

Os **Broadcast Receivers** são componentes que ficam "escutando" a ocorrência de determinados eventos, que podem ser nativos ou disparados por aplicações. Uma aplicação pode, por exemplo, utilizar um *broadcast receiver* para ser avisada quando o dispositivo estiver recebendo uma ligação e, com base nessa informação, realizar algum tipo de processamento.

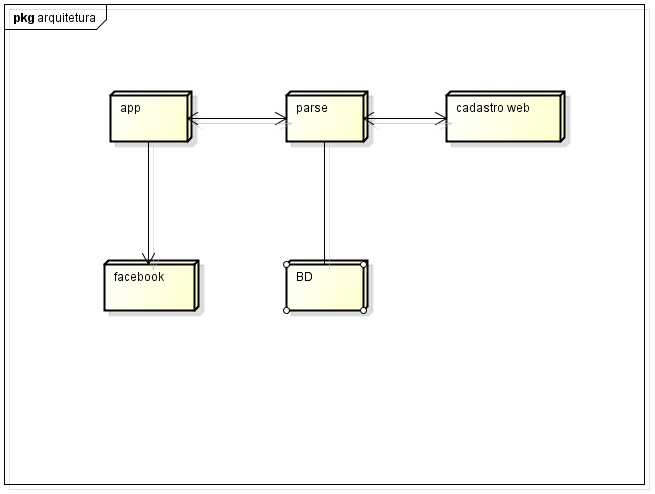
Junto os estes componentes, existe o arquivo de manifesto **AndroidManifest.xml**. Ele é obrigatório e único para cada aplicação. É nele que são feitas as configurações gerais da aplicação e dos componentes que fazem parte dela. E, juntando tudo isto, existe a figura do **Android Core**, que na verdade não é um componente específico, mas sim a plataforma Android propriamente dita. É ele quem proporciona a interação entre os componentes e as aplicações e torna possível a execução do código.

# Visões Arquiteturais

• Lógica:



• Operacional:



• Caso de uso:

