Acadsystem

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 por usabilidade e fornecimento de informações fieis os resultados obtidos pelos usuários em suas diversas atividades na academia. Possuindo fácil acesso ao sistema, em dispositivos comumente utilizado pelos usuários, como dispositivos moveis.

O banco de dados utilizado será o SQlite, trazendo facilidade de manutenção e de utilização, por parte dos desenvolvedores, acelerando o processo de produção.

A linguagem de programação também possui como qualidade a facilidade de uso, integridade e segurança.

# Premissas e Dependências

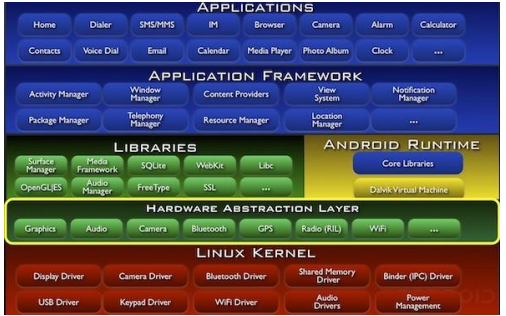
# Requisitos Críticos da Arquitetura

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

# Decisões, Restrições e Justificativas

* **Utilizar o SQlite para administração da persistência**: Este SGBD funciona em mais de 20 plataformas proporcionando consistência, alta performance confiabilidade e fácil utilização, além de flexibilidade e controle.
* **Linguagem de programação:** Android é a plataforma escolhida, por apresentar o paradigma adequado a solução, além da facilidade de uso e de integração com outras tecnologias.

# Mecanismos Arquiteturais



* **Linux Kernel**

O Kernel atua como uma camada de abstração entre o hardware e o resto do sistema

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

* **Libraries**

Nesta camada estão as bibliotecas nativas escritas em C/C++ que integram a plataforma, APIs como o OpenGL ES (para renderização 3D), Parse (gerenciador de bancos de dados) e suporte a diversos formatos de áudio e vídeo. A utilização destas bibliotecas é possível através do Framework Android.

* **Android Runtime**

Substituta da DVM, ART é a nova máquina virtual do Android que possui como principal característica o uso de compilação Ahead-of-tme (AOT). Fornece 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 (Evolução da JVM para ser usada em dispositivos moveis)*, que é uma máquina virtual para suporte à execução de aplicações.

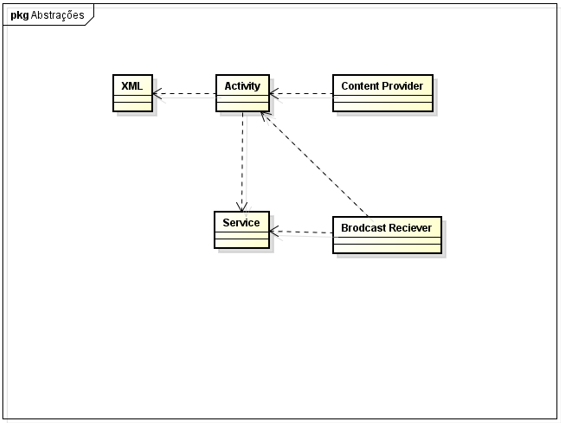
* **Applications Framework**

APIs do Android que são normalmente aplicações que executam sobre a plataforma. Os gerenciadores de serviços de telefonia, localização e notificação são alguns exemplos do que este framework disponibiliza.

* **Applications**

Na camada de aplicação está localizada uma lista de aplicações que incluem um cliente de e-mail, programa sms/mms, calendário, mapas, navegador, gerenciador de contatos, entre outros**.** Elas podem ser tanto aplicações nativas - como o gerenciador de contatos - 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

**Activity**: Gerenciador do ciclo de vida das aplicações e fornece uma pilha de navegação entre elas (Activities). Normalmente, cada Activity se refere a uma tela da aplicação e é implementada como uma única classe que deriva da classe base Activity. Essa classe deve exibir uma interface com o usuário e tratar os eventos a ela relacionados.

**Broadcast Receiver**: Esse bloco é usado quando você deseja que a aplicação reaja a algum evento externo, como por exemplo, o telefone tocar, Internet disponível ou então em um determinado momento (alarme). Esse bloco não exibe nenhuma interface com o usuário, mas pode através da classe NotificationManager avisar o usuário que tal evento ocorreu.

**Service**: Um service é o código que é executado durante toda a aplicação e sem a necessidade de uma interface com o usuário. Um bom exemplo disso são tocadores de música ou players de vídeo. Pois, após o usuário escolher qual música deseja ouvir ele não quer que sua música pare por causa de algum outro evento.

**Content Provider**: Utilizado para acessar dados de outras aplicações (Lista de Contatos) ou compartilhar seus próprios dados. São provedores de conteúdo que gerenciam estruturas de dados, eles encapsulam as informações e fornecem mecanismos para definir a segurança dos dados.

**Notification** **Manager**: Gerenciador de alertas customizados na barra de status. As notificações podem assumir diferentes formas como: ícone persistente localizado na barra de status acessível através do lançador, ligar o LED no dispositivo ou alertar o usuário com luz de fundo em conjunto com som e vibração.

**Resource Manager**: O sistema de recursos do Android mantém o controle de todos os ativos não só de códigos ligados ao aplicativo mas fornece acesso a recursos como textos localizados, gráficos e arquivos de layout.

**Views**: Esta classe representa o bloco de construção de componentes de interface do usuário, ela funciona como base para os Widgets, que são usados para criar componentes de interface com o usuário, ou seja, um sistema que contém listas, grids, caixas de texto, botões, web browser, entre outros.

# Visões Arquiteturais

• Lógica:

•operacional:

• Caso de uso:

