

Documento de Arquitetura

Versão 1.0

Histórico da Revisão

| **Data** | **Versão** | **Descrição** | **Autor** |
| --- | --- | --- | --- |
| 24/05/2017 | 2.4 | Revisão geral do documento | Jaelson Santos |
| 23/05/2017 | 2.3 | Tópico refeito parcialmente  10 – Tamanho e Desempenho | Jaelson Santos |
| 22/05/2017 | 2.2 | Tópico refeito parcialmente  8- Visão da Implementação | Jaelson Santos |
| 21/05/2017 | 2.1 | Tópico refeito parcialmente  7-Visão de Implantação | Jaelson Santos |
| 21/05/2017 | 2.0 | Tópico refeito totalmente  6- Visão de Processos | Jaelson Santos |
| 21/05/2017 | 1.9 | Tópico refeito parcialmente  4 – Visão de Casos de Uso | Jaelson Santos |
| 20/05/2017 | 1.8 | Tópico refeito totalmente  3 – Metas e Restrição da Arquitetura | Jaelson Santos |
| 19/05/2017 | 1.7 | Tópico refeito totalmente  2 – Representação Arquitetural | Jaelson Santos |
| 17/05/2017 | 1.6 | Alteração total no tópico 1 - Introdução | Jaelson Santos |
| 15/05/2017 | 1.5 | 5.Visão Lógica | Jaelson Santos |
| 15/05/2017 | 1.4 | 9.Visão de Dados (opcional, caso o seu software possua) | Jaelson Santos |
| 15/05/2017 | 1.3 | Realizado mudanças no tópico 5 do documento, foi adicionado o diagrama de pacotes e explicado o mesmo. | Jaelson Santos |
| 14/05/2017 | 1.2 | Realizado alterações no tópico 8, foi feito alguns ajustes na ortografia especifica para melhor entendimento. | Jaelson Santos |
| 13/05/2017 | 1.1 | Realizado mudanças no tópico 2 do documento. | Jaelson Santos |
| 13/05/2017 | 1.0 | Realizado mudanças no tópico 6 do documento, adicionado o diagrama de classes | Jaelson Santos |
| 17/04/2017 | 0.9 | 6.Visão de Processos | Francisco Sergio  Rafael Cidrão |
| 17/04/2017 | 0.8 | 7.Visão de Implantação | Jaelson Santos |
| 17/04/2017 | 0.7 | 8.Visão da Implementação | Daniel Carlos  Francisco Sergio |
| 11/04/2017 | 0.6 | 10.Tamanho e Desempenho | Daniel Carlos |
| 11/04/2017 | 0.5 | 11.Qualidade | Francisco Sergio  Rafael Cidrão |
| 06/03/2017 | 0.4 | 1.Introdução | Rafael Cidrão |
| 23/04/2023 | 0.3 | 10.Tamanho e Desempenho | Rayanne Machado |
| 23/04/2023 | 0.2 | 11.Qualidade | Rayanne Machado |
| 23/04/2023 | 0.1 | 1.Introdução | Rayanne Machado |

Índice Analítico

[1. Introdução 5](#_heading=h.30j0zll)

[1.1 Finalidade 5](#_heading=h.3znysh7)

[1.2 Escopo 5](#_heading=h.tyjcwt)

[1.3 Definições, Acrônimos e Abreviações 5](#_heading=h.1t3h5sf)

[1.4 Referências 5](#_heading=h.4d34og8)

[1.5 Visão Geral 6](#_heading=h.2s8eyo1)

[2. Representação Arquitetural 6](#_heading=h.17dp8vu)

[3. Metas e Restrições da Arquitetura 8](#_heading=h.3rdcrjn)

[3.1 Restrições 8](#_heading=h.26in1rg)

[4. Visão de Casos de Uso 8](#_heading=h.lnxbz9)

[4.1 Realizações de Casos de Uso 9](#_heading=h.35nkun2)

[5. Visão Lógica 11](#_heading=h.1ksv4uv)

[5.1 Visão Geral 11](#_heading=h.44sinio)

[5.2 Pacotes de Design Significativos do Ponto de Vista da Arquitetura 12](#_heading=h.2jxsxqh)

[6. Visão de Processos 13](#_heading=h.z337ya)

[7. Visão de Implantação 13](#_heading=h.3j2qqm3)

[8. Visão da Implementação 14](#_heading=h.1y810tw)

[8.1 Visão Geral 14](#_heading=h.4i7ojhp)

[8.2 Camadas 14](#_heading=h.1ci93xb)

[9. Visão de Dados (opcional) 15](#_heading=h.3whwml4)

[10. Tamanho e Desempenho 16](#_heading=h.2bn6wsx)

[11. Qualidade 17](#_heading=h.qsh70q)

Documento de Arquitetura de Software

# Introdução

O documento tem como objetivo fornecer uma visão geral do software UVS abrangendo suas funcionalidades, análise e definição das necessidades e recursos necessários aos usuários envolvidos. As informações contidas neste documento visam estabelecer subsídio para a modelagem do sistema, incluindo desde a criação de requisitos até o desenvolvimento propriamente dito. Os detalhes de como o software funcionará serão descritos no restante do documento.

## Finalidade

Este documento oferece uma visão geral arquitetural abrangente do sistema, usando diversas visões arquitetônicas para representar diferentes aspectos do sistema. O objetivo deste documento é capturar e comunicar as decisões arquitetônicas significativas que foram tomadas em relação ao sistema.

## Escopo

Este documento trata da implementação arquitetural do EPYMAPS com base nos diagramas de casos de uso, diagramas de classe, diagrama de pacote, diagrama de implantação, além da visão lógica do sistema com o detalhamento da arquitetura MVP que será utilizada como padrão arquitetural no desenvolvimento.]

## Definições, Acrônimos e Abreviações

**MVP**: Padrão arquitetural no qual se divide em 3 camadas: model, view, presente.

**Android**: Sistema operacional para dispositivos móveis.

**IDE**: Ambiente de desenvolvimento.

**Java**: Linguagem de programação orientada a objetos.

**SO**: Sistema Operacional.

**Google**: Empresa desenvolvedora do SO Android e outras ferramentas.

**EpyMaps**: Aplicativo mobile, na qual o documento se trata.

## Referências

* Diagrama de Casos de uso.
* Especificações
* Requisitos
* Visão
* Teste

## Visão Geral

Este documento representa a arquitetura do sistema EpyMaps, logo todo o contexto de solução que a arquitetura necessita, assim esclarecendo dúvidas e facilitando na hora do desenvolvimento.

Segue a estrutura do documento:

* Introdução
* Representação Arquitetural
* Metas e Restrição da Arquitetura
* Visão de Casos de Uso
* Visão Lógica
* Visão de Processos
* Visão de Implantação
* Visão de Implementação
* Visão de Dados
* Tamanho e Desempenho
* Qualidade

# Representação Arquitetural

**Descrição do cenário**

**App:** O sistema Epymaps estará instalado no dispositivo em que o usuário esteja utilizando, logo após feito login poderá desfrutar das funcionalidades do mesmo e fazer as requisições a view.

**DB Local:** Base de dados local que está contida no sistema, o usuário enquanto estiver off-line poderá utilizar o sistema, mapeando locais e salvando-os estes dados na memória do dispositivo, onde posteriormente quando estiver conectado à internet o sistema poderá fazer a sincronização com o banco de dados externo.

**Apis Login:** Estas apis serão utilizadas para o processo de cadastro e login do usuário a plataforma, assim o usuário não precisará efetuar toda a burocracia que é o preenchimento dos dados cadastrais a partir do sistema.

**Web Service:** O Web Service será a ponte entre a aplicação e o servidor de dados externo, a camada presenter fará a requisição a camada model que será a camada responsável por fazer as requisições no formado .json para o webservice, logo o webservice se comunicará com o servidor de dados na nuvem e posteriormente responderá as requisições solicitadas.

**Model:** Camada responsável por toda regra de negócio da aplicação, nela conterá as entidades do que modelam o projeto, classes validadoras, classe DAOs que são as classes responsáveis pela manipulação dos dados, por exemplo: Usuario, UsuarioDAO, Mapeamento, MapeamentoDAO entre outras.

**View:** Nesta camada conterá os layouts(Activity ou Fragment) responsáveis pela comunicação visual do app, como também capturar as interações do usuário com o app, por exemplo: Tela Login, Tela de Cadastro, Ambiente do usuário, entre outras.

**Presenter:** Será a camada responsável por responder as invocações das camadas model e view, em nenhum momento quando se utiliza a arquitetura MVP as camadas model e view se comunicam diretamente, ou seja, como foco principal a separação da regra de negócio da camada de apresentação ao usuário. E resta ao presenter trabalhar a comunicação de dados que entra em ambas as camadas citadas, vale lembrar que é possível também incluir partes de logica de negócio na camada presenter.

**DB Nuvem:** Base de dados na nuvem que estará contido todos os dados armazenados através da aplicação.

# Metas e Restrições da Arquitetura

A meta deste documento de arquitetura tem como principal objetivo possibilitar a melhor compreensão entre os desenvolvedores e dessa maneira designar quais os elementos serão implementados no sistema.

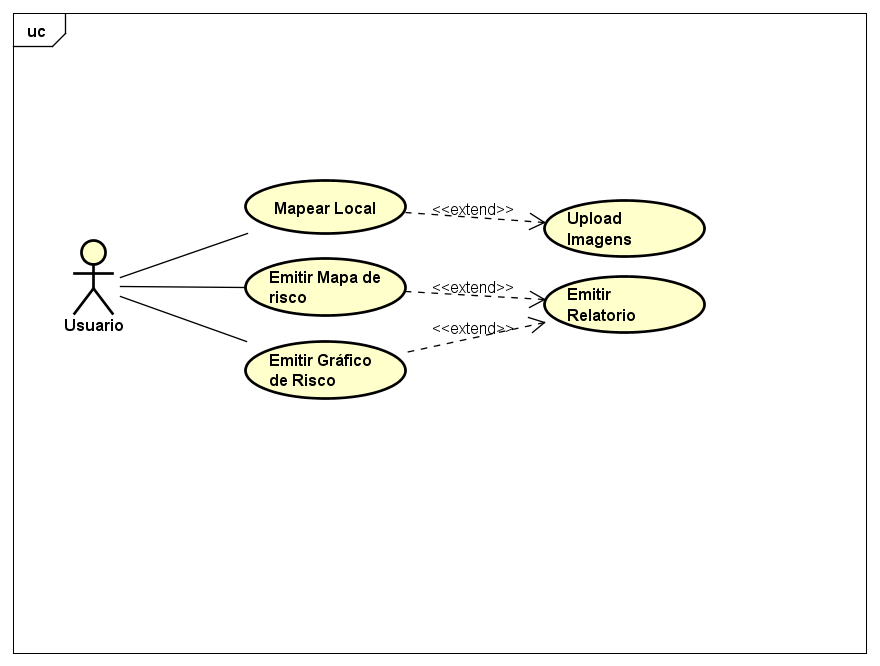
Esta arquitetura tem como proposta arquitetural do sistema utilizar o padrão MVP(Model-View-Presenter) que já foi citado anteriormente no documento, esta arquitetura irá favorece o desenvolvimento como também as manutenções que venham a ter futuramente no sistema.

## Restrições

* O aplicativo será desenvolvido utilizando a linguagem de programação Java.
* A base da dados local que utilizaremos será o SQLite na qual será armazenado dados temporários no dispositivo do usuário enquanto ele estiver usando a aplicação em modo off-line. Quando o usuário conectar o seu dispositivo a uma rede móvel ou wi-fi o app fará a sincronização destas informações com a base de dados na nuvem.
* A base de dados na nuvem será o MySql.
* A ferramenta para o desenvolvimento será a IDE Android Studio que é a ferramenta oficial para desenvolvimento Android, ela foi desenvolvida pela empresa Google que é a empresa responsável pela disponibilização das versões do sistema operacional Android.

# Visão de Casos de Uso

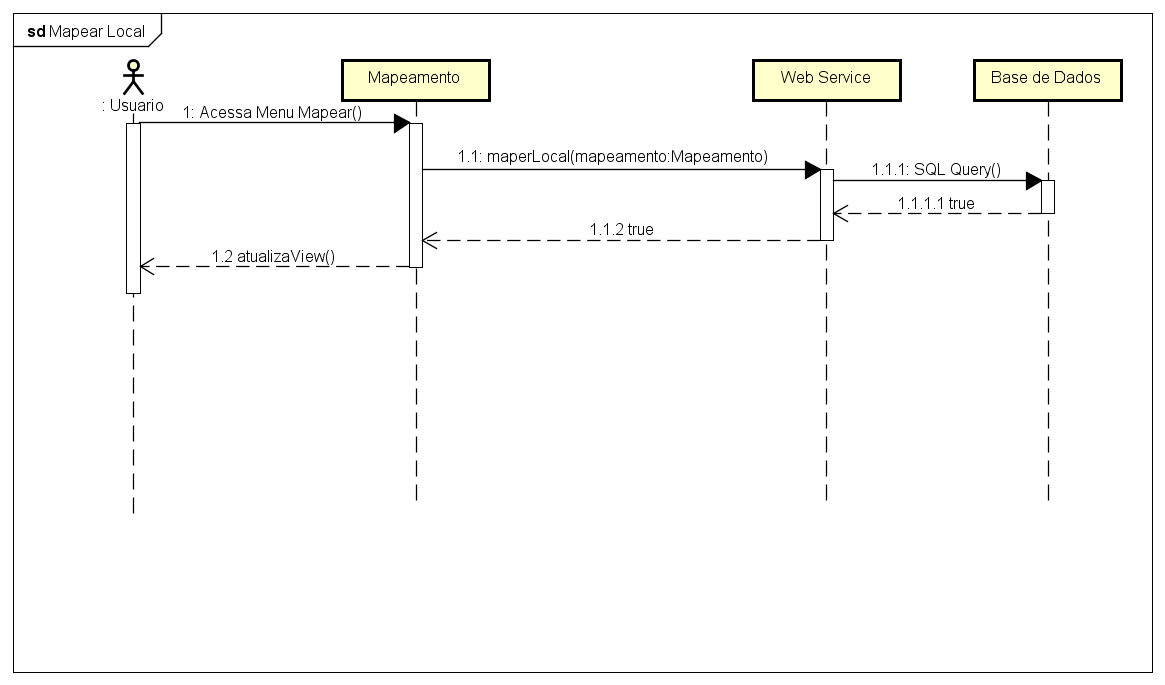
Abaixo é apresentado os principais casos de uso que são os pontos fortes da aplicação na qual o usuário poderá utilizar:



* **Mapear Local** – Esta funcionalidade permite que o usuário mapeie locais onde possuem índice de contaminação e risco para sociedade.
* **Emitir Mapa de Risco** – Esta funcionalidade permite que o usuário emita relatórios contendo índices de mapeamentos naquela determinada área.
* **Emitir Gráfico de Risco** – Esta funcionalidade permite que o usuário emita gráficos referentes aos bairros que ele deseja saber como está o índice de mapeamento.

## Realizações de Casos de Uso

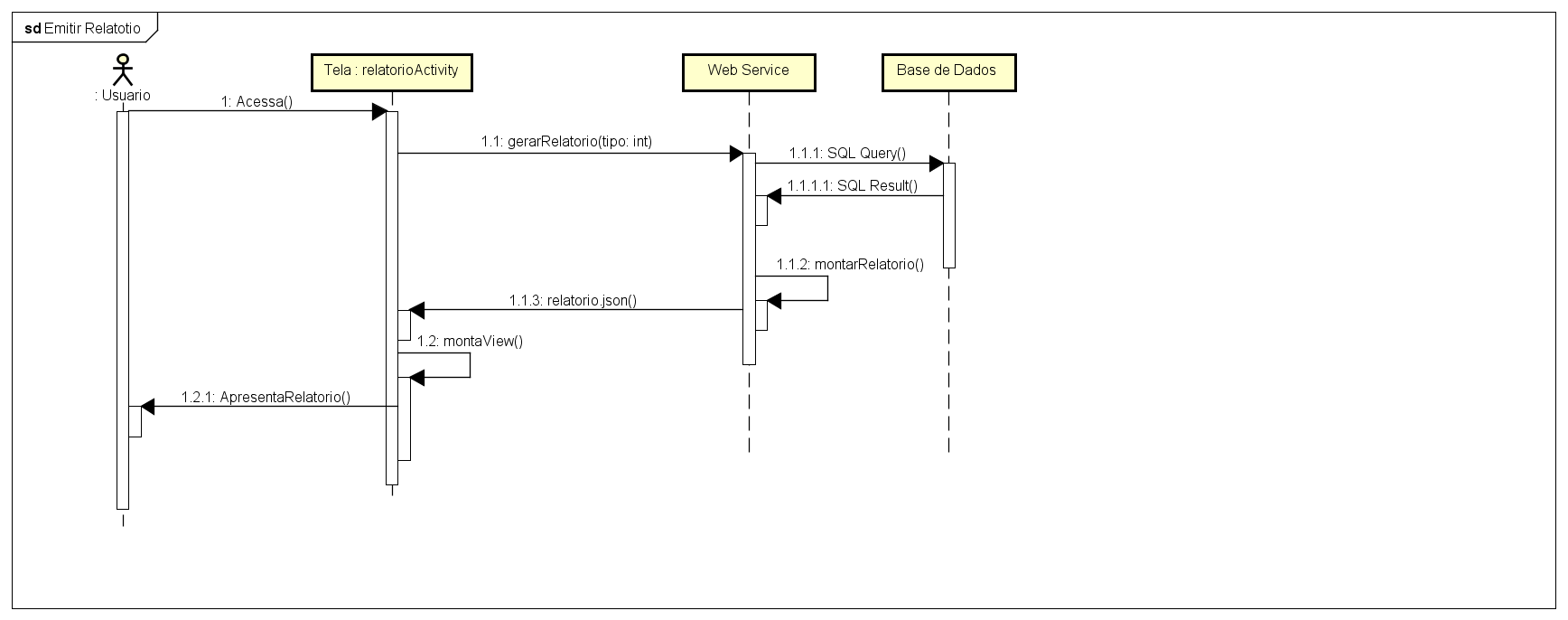
1. **Mapear Local**



***Figura 3 Diagrama Sequencia Mapear Local***

O usuário acessará o menu mapear, no qual será redirecionado a tela de mapeamento, onde terá o formulário para preenchimento, após clicar no botão mapear a requisição será feita ao web service onde o mesmo enviara a requisição através de comandos sqls para a base de dados, e retornará caso tenho sucesso ou não da operação para que seja atualizada a tela do usuário.

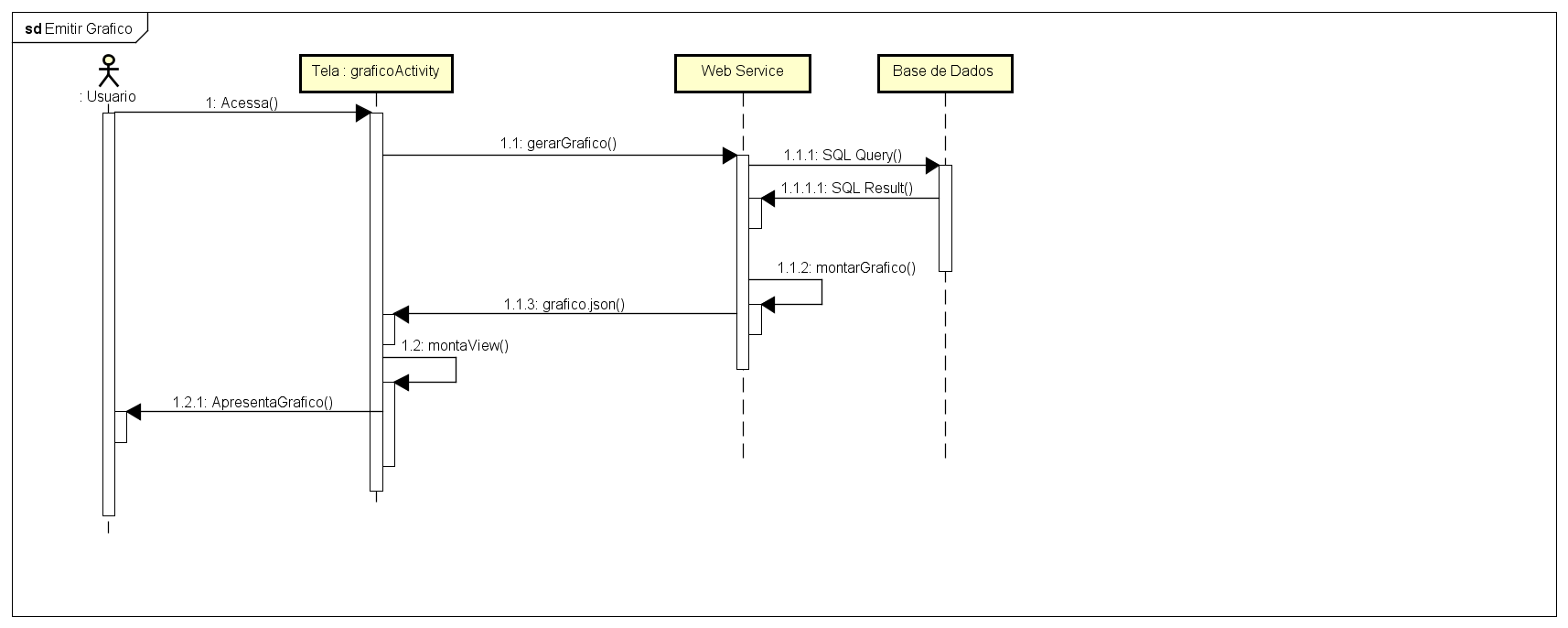
1. **Emitir Relatório**



***Figura 4 Diagrama de Sequencia Emitir Relatório***

O usuário acessa o menu relatório, onde informará os dados que deseja no relatório, logo o web service fara a requisição ao banco de dados através de comandos sql, logo o banco de dados retornara o resultado da filtragem, o web service montará o relatório da busca no formato JSON e posteriormente montar a tela com o relatório requisitado pelo usuário.

1. **Emitir Gráfico**



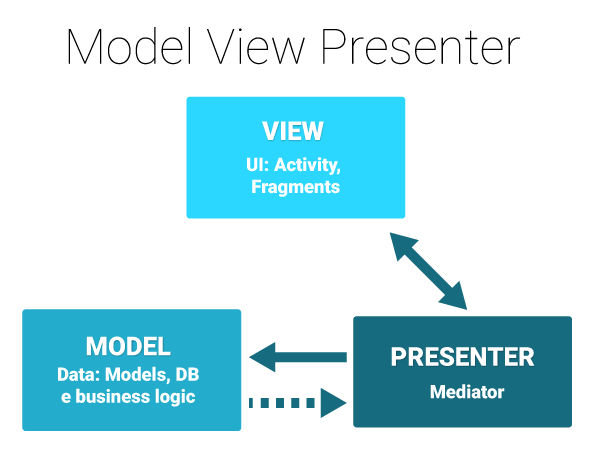
***Figura 5 Diagrama de Sequencia Emitir Gráfico***

O usuário acessa o menu estatísticas, onde informará os dados que deseja no gráfico, logo o web service fara a requisição ao banco de dados através de comandos sql, logo o banco de dados retornara o resultado da filtragem, o web service montará o gráfico da busca no formato .JSON e posteriormente o gráfico será montado e apresentado na tela do usuário.

# Visão Lógica

## Visão Geral

* **MVP**



***Figura 6 Disponível em: <https://code.tutsplus.com/tutorials/an-introduction-to-model-view-presenter-on-android--cms-26162>***

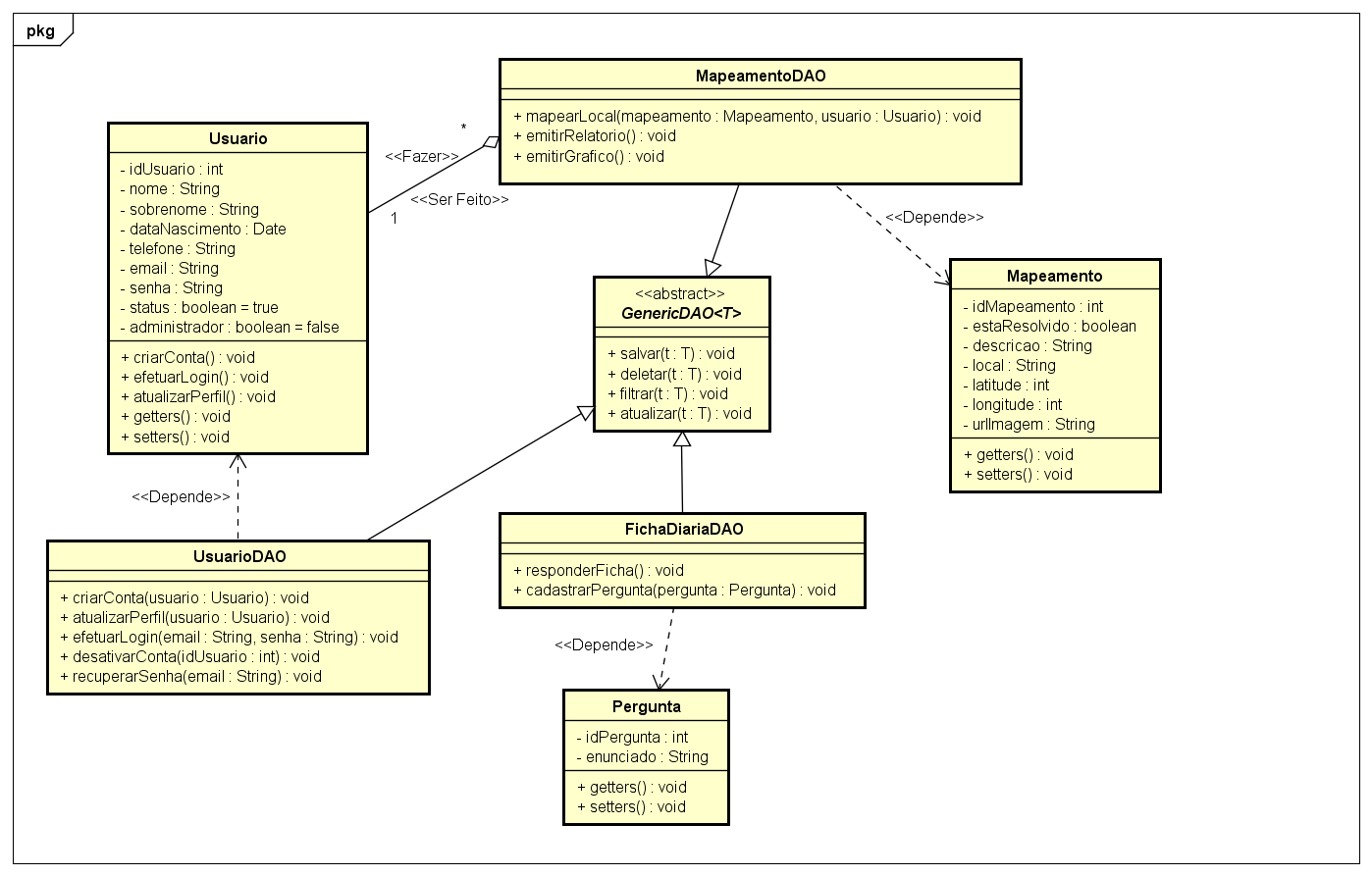
A figura acima apresenta como é feita a comunicação entre as camadas na arquitetura MVP. O model mantém a lógica de negócios da aplicação. Ele controla como os dados podem ser criados, armazenados e modificados. A view é responsável por toda interface (Activities ou Fragments) que exibe os dados e direciona as ações do usuário para o presenter. O presenter será o mediador entre o model e a view, ele é responsável por recuperar os dados do model e apresentar na view, vale lembrar que o presenter também processa as ações do usuário enviadas para ele através da view.

## Pacotes de Design Significativos do Ponto de Vista da Arquitetura

No diagrama acima foi apresentado a estrutura dos pacotes na aplicação mobile, na qual foi construída baseado na arquitetura MVP para dispositivos Android.

# Visão de Processos

Segue abaixo o digrama de classes referente ao projeto:



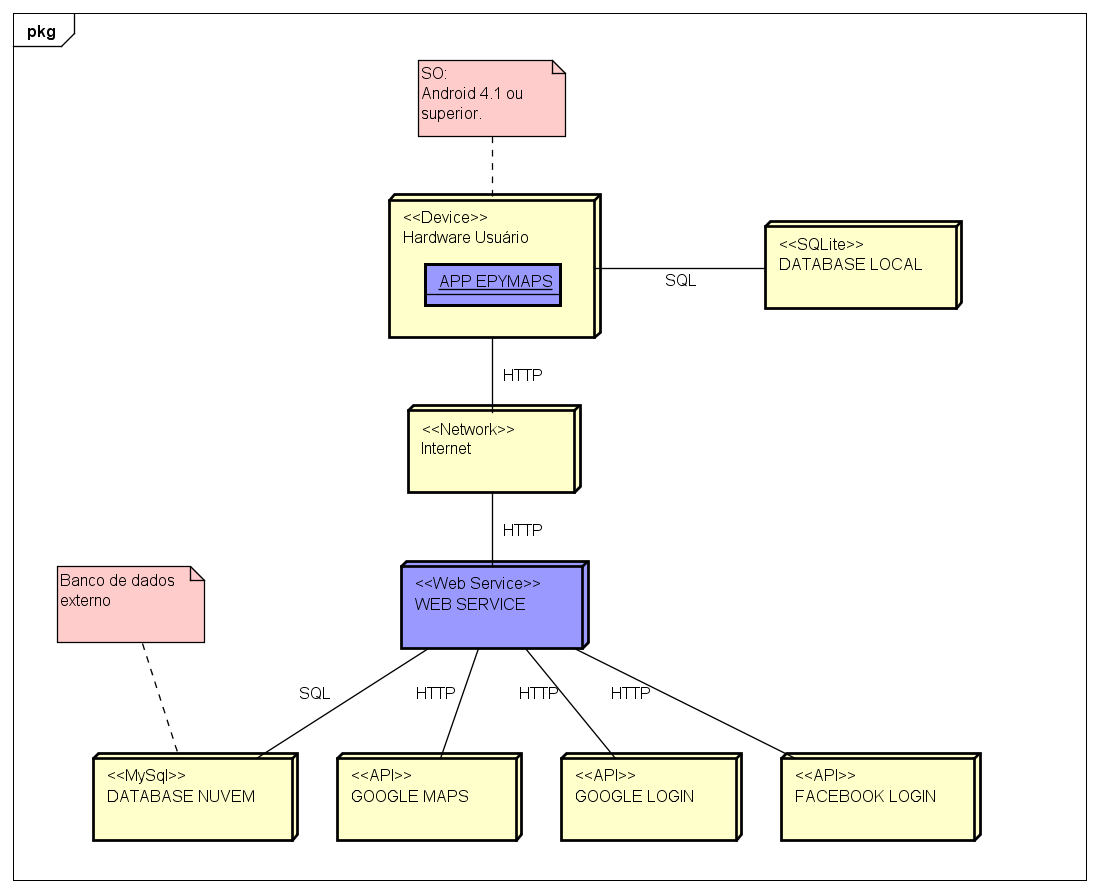
***Figura 8 Diagrama de Classe***

# Visão de Implantação

O EPYMAPS utilizará das seguintes distribuições para a comunicação dos dispositivos dos usuários com o aplicativo.

* Web Service.
* Banco de dados na nuvem(MySql).
* API Google Maps.
* API Google Login.
* API Facebook Login.

Segue abaixo a representação gráfica do diagrama de implantação:



# Visão da Implementação

# Visão Geral

Seguindo o padrão arquitetural MVP, o aplicativo está dividido em 3 camadas; Model, View, Presenter. Cada camada com funções distintas e com a sua devida importância dentro da arquitetura e do aplicativo em si. As classes da camada Model são as classes entidades da aplicação, classes validadoras, classes genéricas de persistências e também as classes de manipulação de dados da aplicação. As classes da camada View são responsáveis pelos layouts do aplicativo, ou seja, e a parte que interage com o usuário. As classes da camada Presenter são as classes responsáveis por mediar a comunicação do Model como o View.

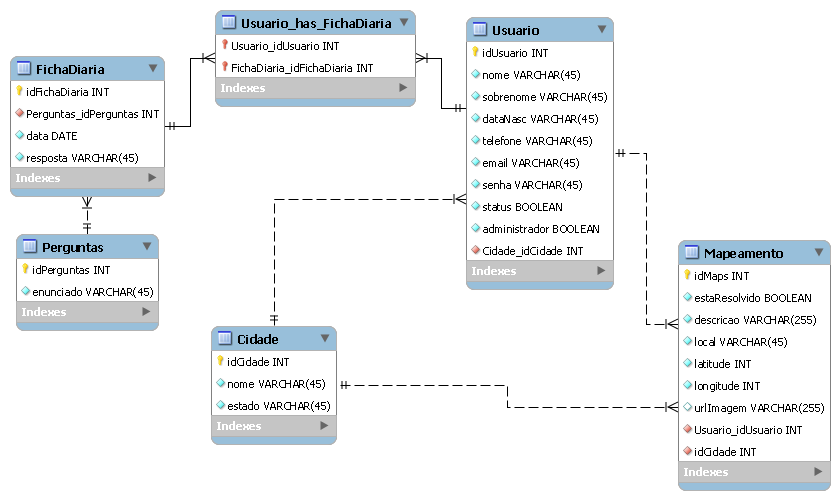
## Camadas

**View:** A camada View é responsável pela interação do usuário com a aplicação, nessa camada são realizadas algumas etapas como o desenvolvimento de interface do usuário e a interação para a configuração das funcionalidades. Nessa camada estão presentes os layouts por exemplo: MapeamentoActivity, VizualizarMapaActivity, CadastroActivity, entre outras.

**Model:** A camada Model é a responsável pelo armazenam das classe entidades do sistema, classe para manipulação dos dados, classes validadoras, classes de conexão. Ex: Usuario, Mapeamento, FichaDiaria, UsuarioDAO, MapeamentoDAO, FichaDiariaDAO, entre outras.

**Presenter:** A Camada Presenter representa o ponto forte da arquitetura MVP, pois ele será responsável por mediar a comunicação da camada Model com a View, assim estas camadas não se comunicarão diretamente, o presenter processará as requisições feitas pela view, se comunicará com o model e ira recuperar os dados do model e apresentara na view, assim proporcionando maior segurança, organização do código e posteriormente facilidade para adicionar novas funcionalidades ou manutenções.

# Visão de Dados (opcional)





A seguir é apresentado o dicionário de dados de acordo com o MER- Modelo Entidade Relacionamento:

| **Dicionário de dados** | | | |
| --- | --- | --- | --- |
| **Tabela** | | **Atributos** | **Descrição** |
| 01 | Cidade | idCidade INT  nome VARCHAR(45)  estado VARCHAR(45) | Registra uma cidade na base de dados |
| 02 | Usuario | idUsuario INT  nome VARCHAR(45)  sobrenome VARCHAR(45)  dataNascimento Date  telefone VARCHAR(20)  email VARCHAR(100)  senha VARCHAR(45)  status BOOLEAN  administrador BOOLEAN  Cidade\_idCidade INT | Registra um usuário na base de dados, relaciona a tabela Cidade |
| 03 | Mapeamento | idMaps INT  estaResolvido BOOLEAN  descrição VARCHAR(255)  local VARCHAR(100)  latitude INT  longitude INT  urlImagemVARCHAR(255)  Usuario\_idUsuario INT  Cidade\_idCidade INT | Registra um mapeamento na base de dados, relaciona a tabela Usuário e Cidade |
| 04 | Perguntas | idPergunta INT  enunciado VARCHAR(45) | Registra perguntas na base de dados |
| 05 | FicharDiaria | idFichaDiaria INT  data Date  resposta VARCHAR(45)  Perguntas\_idPergunta INT | Registra uma resposta da ficha diária, relaciona a tabela Perguntas |
| 06 | Usuario\_has\_FichaDiaria | Usuario\_idUsuario INT  FichaDiaria\_idFichaDiaria INT | Relacionamento entre as tabelas FichaDiaria e Usuario |

***Tabela 1Dicionário de Dados***

# Tamanho e Desempenho

O Modelo arquitetural usado para o desenvolvimento desta aplicação, trata-se do MVP (Model ,View, Presenter), que irá possibilitar a expansão do software sobre diversas circunstâncias tanto para atualização para correção de bugs como ampliação do sistema adicionando novas funcionalidades . No começo o software será desenvolvido somente para a cidade de Caririaçu. A arquitetura possibilitará que alterações de carga possam ser realizadas e essa é uma característica que o software precisa ter, já que poderá ser implantado em regiões com maior quantidade de usuários.

# Qualidade

A arquitetura utilizada possibilita um melhor desenvolvimento da aplicação, pois pelo fato de ser dividida em camadas, torna mais fácil o processo de manutenção e extensão do software desenvolvido. De fato, a manutenção é um recurso importante em qualquer software, visto que é através desta que se é corrigido bugs que estejam atrapalhando o seu funcionamento. Outro recurso importante que deve estar presente em todo projeto de software é a possibilidade de extensão, que consiste no acréscimo de novas funcionalidades ao software, e através da arquitetura escolhida esse processo pode ser realizado facilmente.