|  |
| --- |
| Histórico de Versões |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** | **Revisor** | **Aprovado por** |
| 12/11/2018 | 1.0 |  | Gustavo Sousa | Rubens Batista |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| Índice |

[**Histórico de Versões**](#_gjdgxs) **1**

[**Índice**](#_1fob9te) **2**

[**Documento de Arquitetura de Software**](#_3znysh7) **2**

1. [**Objetivo do Documento**](#_2et92p0) **2**
2. [Objetivos e Restrições da Arquitetura](#_tyjcwt) 3
3. [**Descrição da Arquitetura**](#_1t3h5sf) **5**
4. [**Camadas e Subsistemas**](#_4d34og8) **5**
   1. [**Padrões e Mecanismos Arquiteturais**](#_2s8eyo1) **6**
   2. [**Outras Visões**](#_2s8eyo1) **7**
5. [**Decisões e Justificativas**](#_2s8eyo1) **8**

|  |
| --- |
| Documento de Arquitetura de Software |

# Objetivo do Documento

O Documento de Arquitetura de Software provê uma visão geral da arquitetura através de diferentes tipos de visões para descrever os diferentes aspectos do sistema.

## Objetivos e Restrições da Arquitetura

O sistema de decisões objetiva permitir aos administradores, minutadores e apoio, a elaboração de atos offline, padronizada e com recursos avançados de editoração. Além destes objetivos, o sistema visa proporcionar a publicação destes atos em base pública, bem como a posterior pesquisa destes online, a partir do conteúdo de seus campos.

Para a especificação de uma arquitetura condizente com os objetivos do sistema, é importante considerar os requisitos significativos e um conjunto de objetivos e restrições arquiteturais. Nesta seção, estes fatores são descritos com intuito de clarificar as premissas que conduziram à especificação da arquitetura.

Os principais objetivos arquiteturais considerados foram:

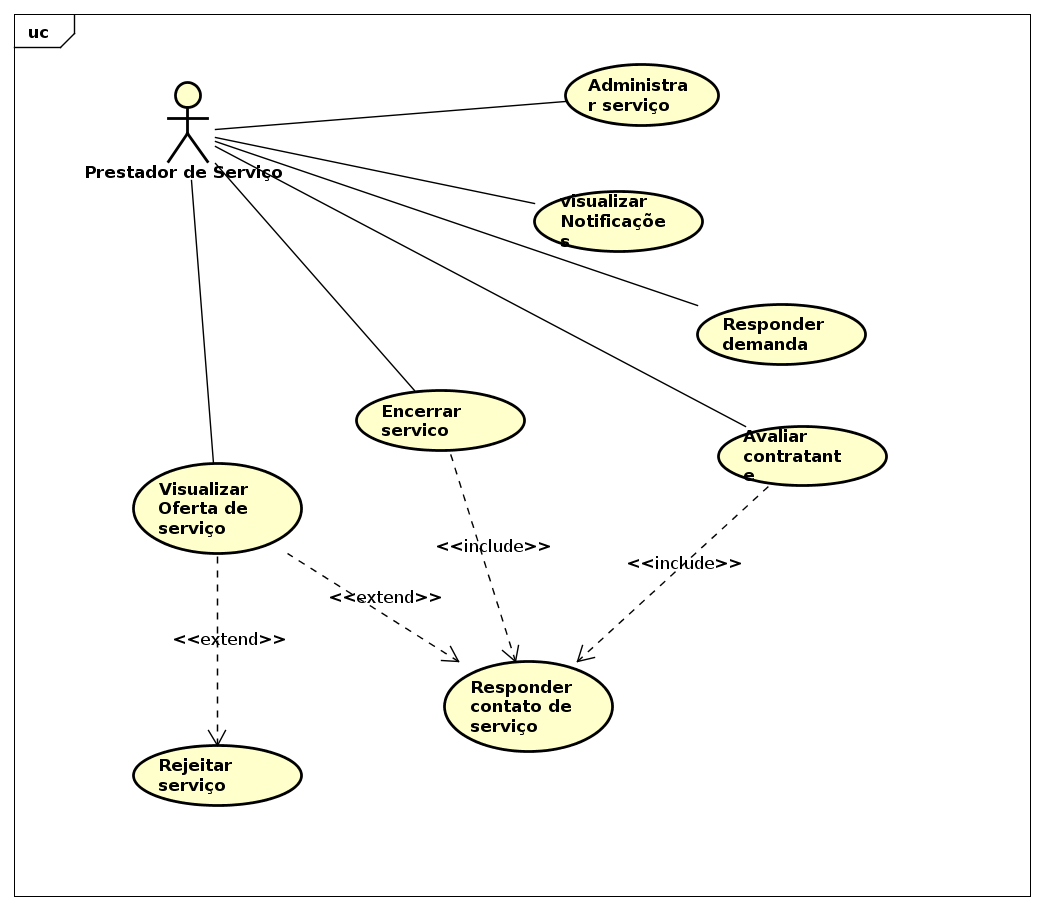
* **Manutenibilidade:** o sistema de decisões deve ser facilmente mantido. Para apoiar este objetivo, pretende-se modularizar as funcionalidades, adotar padrões de projeto e o GitLab – de modo que as responsabilidades do sistema sejam equilibradamente distribuídas e os conhecimentos básicos para evolução estejam facilmente disseminados na equipe;
* **Performance:** tanto a elaboração de atos quanto à pesquisa não devem extrapolar os tempos de resposta definidos na especificação de requisitos não funcionais. Pretende-se adotar um conjunto de práticas preventivas durante o desenvolvimento : a execução de testes rotineiros de desempenho para verificar se o tempo de resposta está adequado;
* **Escalabilidade:** O sistema começará apenas com o básico para funcionar, mas diante do aumento do número de usuários o sistema deve manter o tempo de resposta dentro dos padrões definidos.

As principais **restrições** que direcionaram a definição da arquitetura do sistema, descrita neste documento, são:

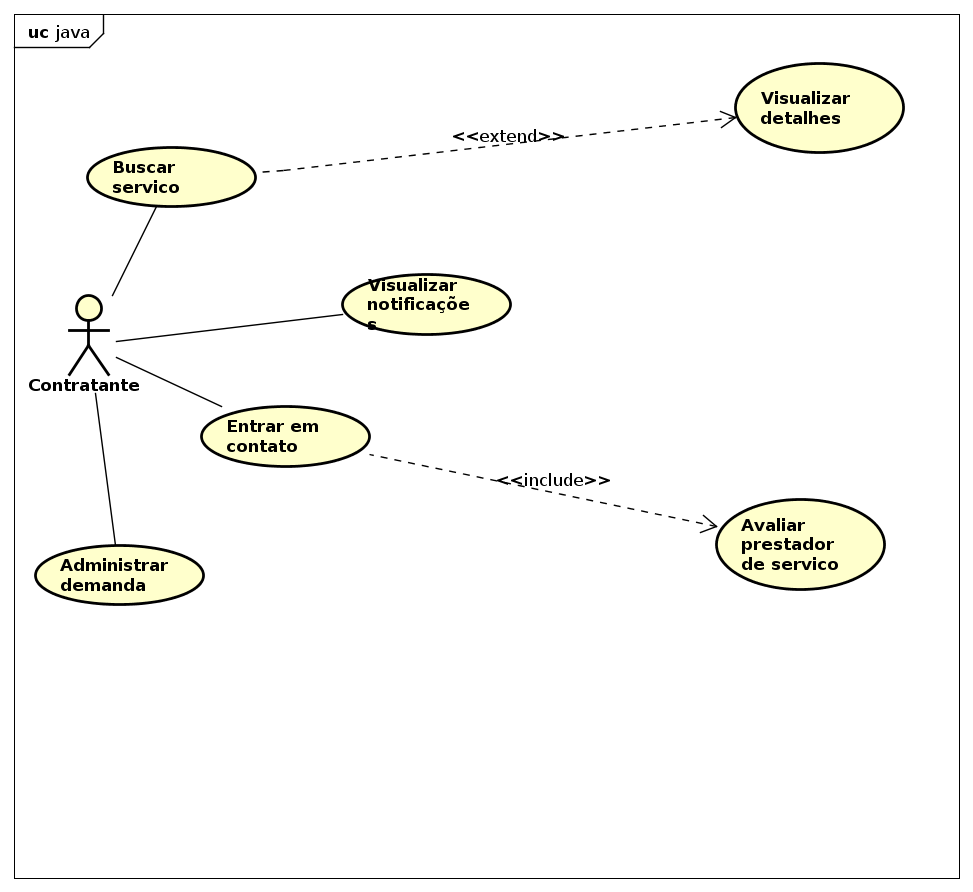
* Necessidade de estar conectado a internet para realizar as buscas no sistema;
* Necessidade de estar online para que uma mensagem seja enviada;
* Necessidade de que o usuário instale o app para ter acesso ao cadastro e todas as outras funcionalidades do sistema.

Diante dessas premissas citadas, definiu-se que os sistema deverá ser desenvolvido para a plataforma smartphones através de framework Ionic para plataformas Android e IOS. O qual será responsável por cadastro de serviços, cadastro de currículos, cadastro de demandas, cadastro de usuários, contato entre o empregador e prestador de serviço.

Os diagramas de casos de uso são apresentados, para os principais requisitos que influenciaram nas decisões arquiteturais são apresentados a seguir, acompanhados pelas justificativas para a escolha dos mesmos:



* **Administrar serviço**:a realização deste cenário permitiu identificar como deveria ser a arquitetura do banco de dados e outros requisitos necessários para o bom funcionamento;
* **Visualizar Notificações**: A realização deste cenário permitiu identificar como dever ser a estrutura que ajudará no envio das mensagens;
* **Visualizar Notificações:** A realização deste cenário permitiu definir o serviço de banco de dados a ser utilizado;
* **Visualizar oferta de serviço**: a realização deste cenário permitiu identificar questões arquiteturais de integração entre as funcionalidades, como notificações, aceitação de serviço, avaliação de contratante e encerramento do serviço.

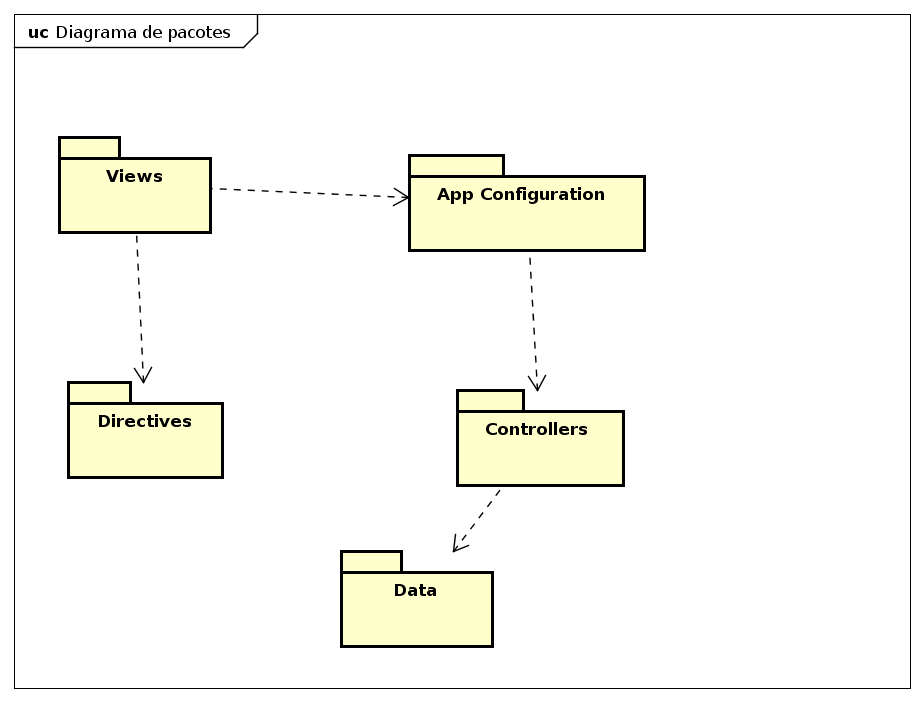
****

* **Buscar serviço**: a realização deste cenário permitiu identificação de questões arquiteturais importantes, tais como: Integração entre as funcionalidades, banco de dados, visualização da informação;
* **Administrar Demanda**:a realização deste cenário permitiu identificar como deveria ser a arquitetura do banco de dados e outros requisitos necessários para o bom funcionamento;
* **Entrar em contato**:a realização deste cenário permitiu identificar o tipo e o serviço de banco de dados a ser utilizado para a execução desta tarefa.

# Descrição da Arquitetura

# Camadas e Subsistemas

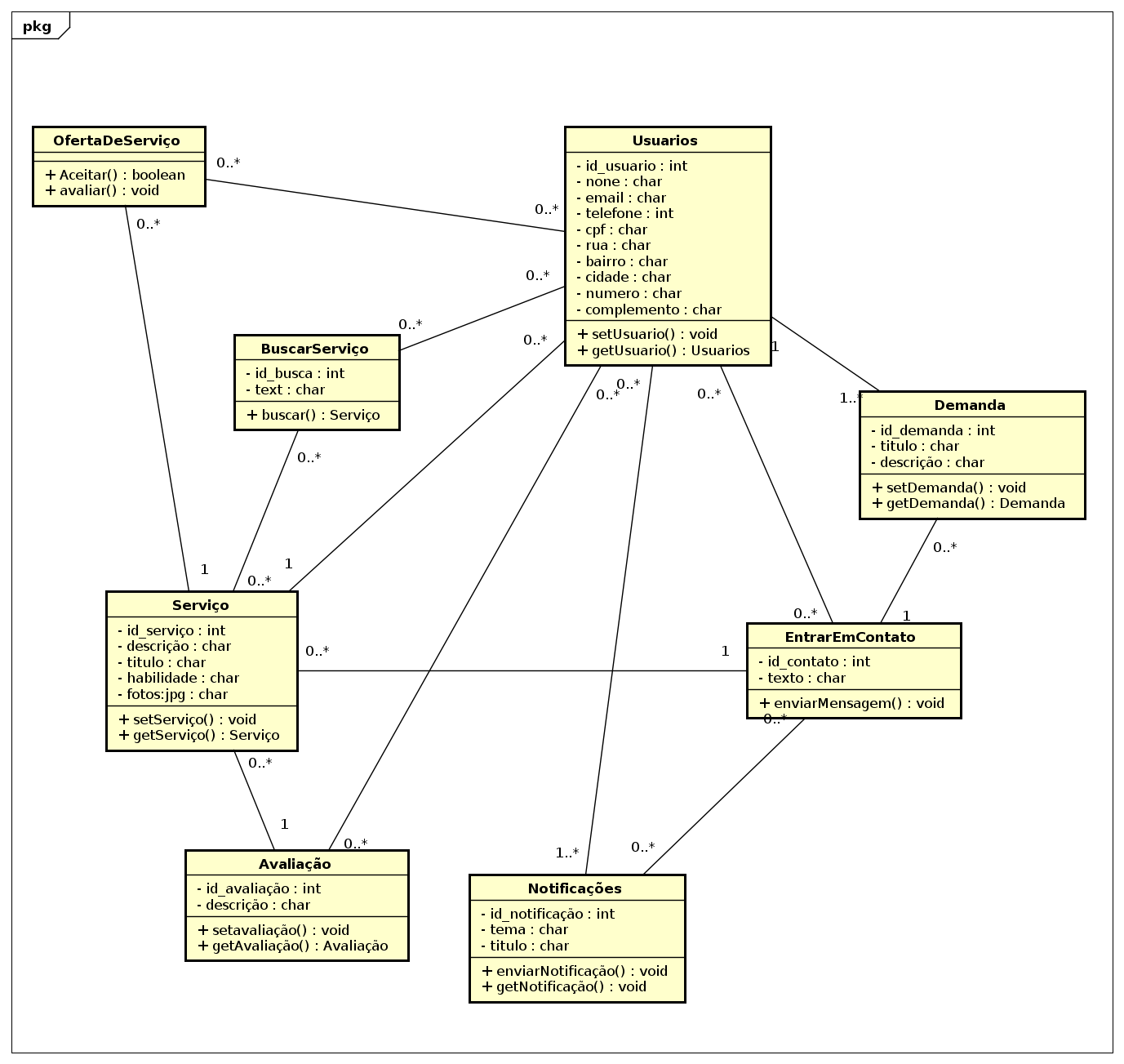
O sistema é dividido em cinco partes de acordo com a estrutura do ionic: Views, App Configuration, Controllers, Directives, Data. O diagrama de pacotes representa esta visão.



# Padrões e Mecanismos Arquiteturais

Para exemplificar, estruturalmente e comportamentalmente, os cenários mais significativos são apresentados os diagramas de classe a seguir.

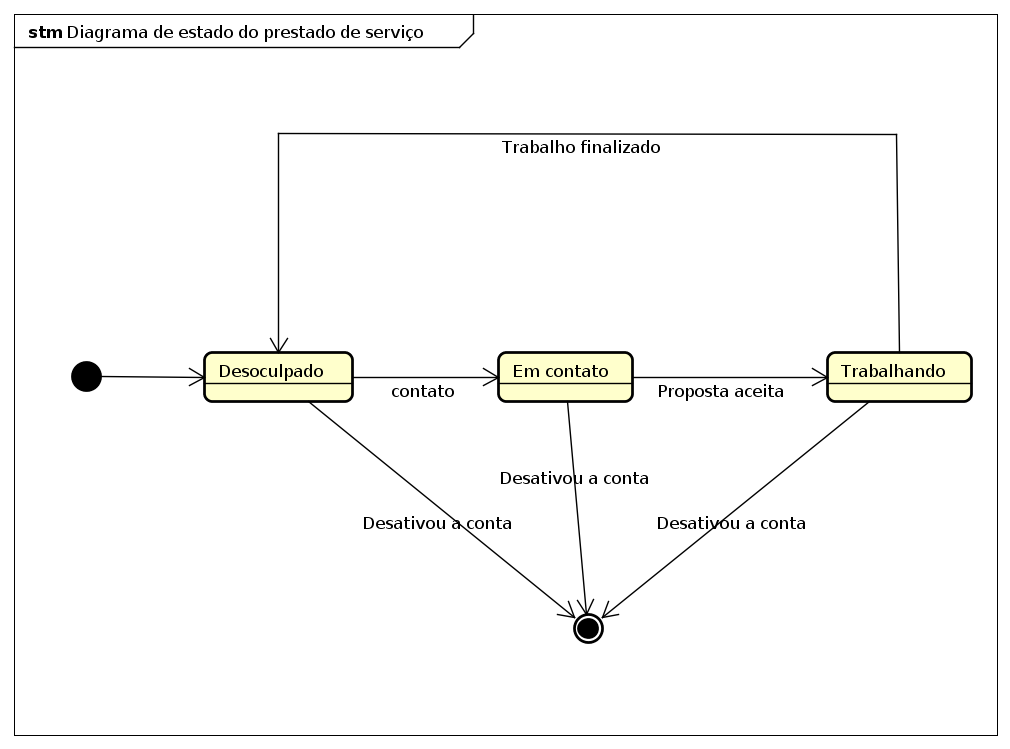
* **Visão estrutural**

****

# Outras Visões

* **Transição de Estados**

Considerou-se relevante representar as transições de estado relativas ao cenário de prestador de serviço, pois, é um fluxo com mais significância. Abaixo segue o diagrama. oculpado



O diagrama ilustra o estado inicial de um prestador de serviço em “desocupado”. A única ação que conduz um modelo a um estado final é a de “Desativar a conta”. Os demais estados são reversíveis/transitórios.

# Decisões e Justificativas

As principais decisões e justificativas são apresentadas no quadro abaixo:

|  |  |
| --- | --- |
| **Decisão** | **Justificativa** |
| Fazer um sistema destinado a dispositivos móveis | Devido ao fato de o sistema necessitar de atualizações e interação constantes com o usuário, optou-se por destinar o sistema a dispositivos móveis. |
| Adotar o Ionic como framework para desenvolvimento | O ionic permite a construção de aplicativos híbridos, o que permite a utilização do sistema em mais de um sistema operacional. |
| Utilizar o banco de dados FireBase | Optou-se por utilizar esse banco de dados pelo fato de ele ser prático para a construção de aplicações com interações em tempo real. |
| Utilizar o banco de dados SQLite | O banco de dados SQLite será utilizado para o armazenamento local das informações. |