**<>**

**Documento Geral de Arquitetura de Software**

**Versão <1.0>**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <01/12/2020> | <1.0> | <Iniciado o documento, Representação Arquitetural, Metas e Restrições da Arquitetura, Visão de Casos de uso.> | <Rogerio Amorim> |

**Documento de Casos de Uso e Estilo Arquitetural**

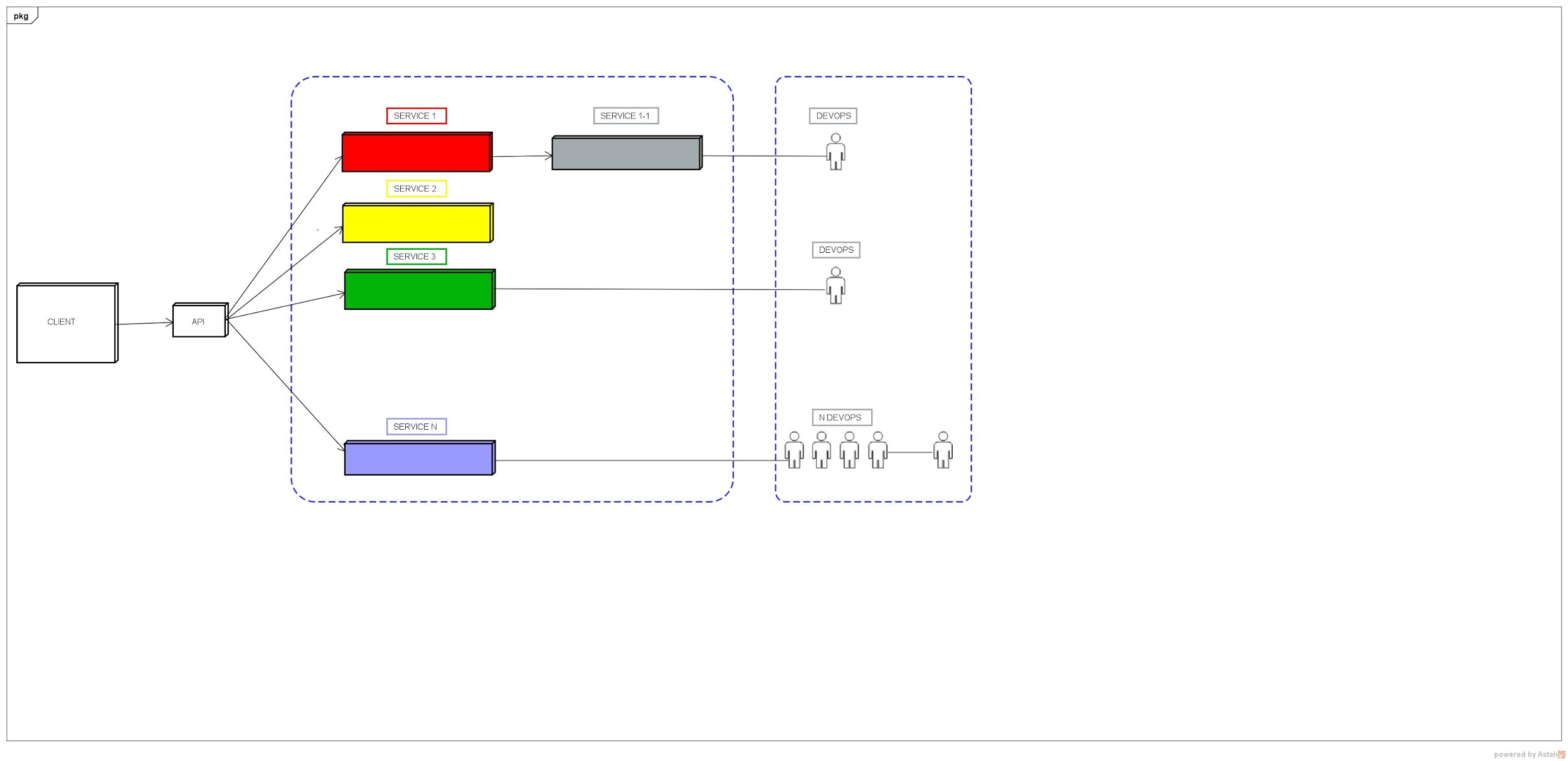
**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <31/10/2020> | <1.0> | <Iniciado o documento, Representação Arquitetural, Metas e Restrições da Arquitetura, Visão de Casos de uso.> | <Rogerio Amorim> |

# Representação Arquitetural

A arquitetura do sistema é a arquitetura baseada na arquitetura de micro serviços. Este padrão caracteriza-se pela presença de uma API que é responsável por gerenciar as requisições feitas por clientes e destiná-las aos serviços responsáveis assim como gerenciar as respostas obtidas pelos serviços e encaminhá-las aos seus respectivos clientes, e pela camada de Micro Serviços independentes que criam uma ecosistema para resolução das atividades requisitadas .

* 1. **Definição de estilo arquitetural**

****

# Metas e Restrições da Arquitetura

1 Todos os usuários devem ter conta do *Gmail*  para poder realizar o cadastro  
2 O Sistema só permitirá atualização da conta se houver acesso à internet.

3 O Sistema deverá contar criptografar todos os dados dos usuários e garantir a segurança do mesmo.

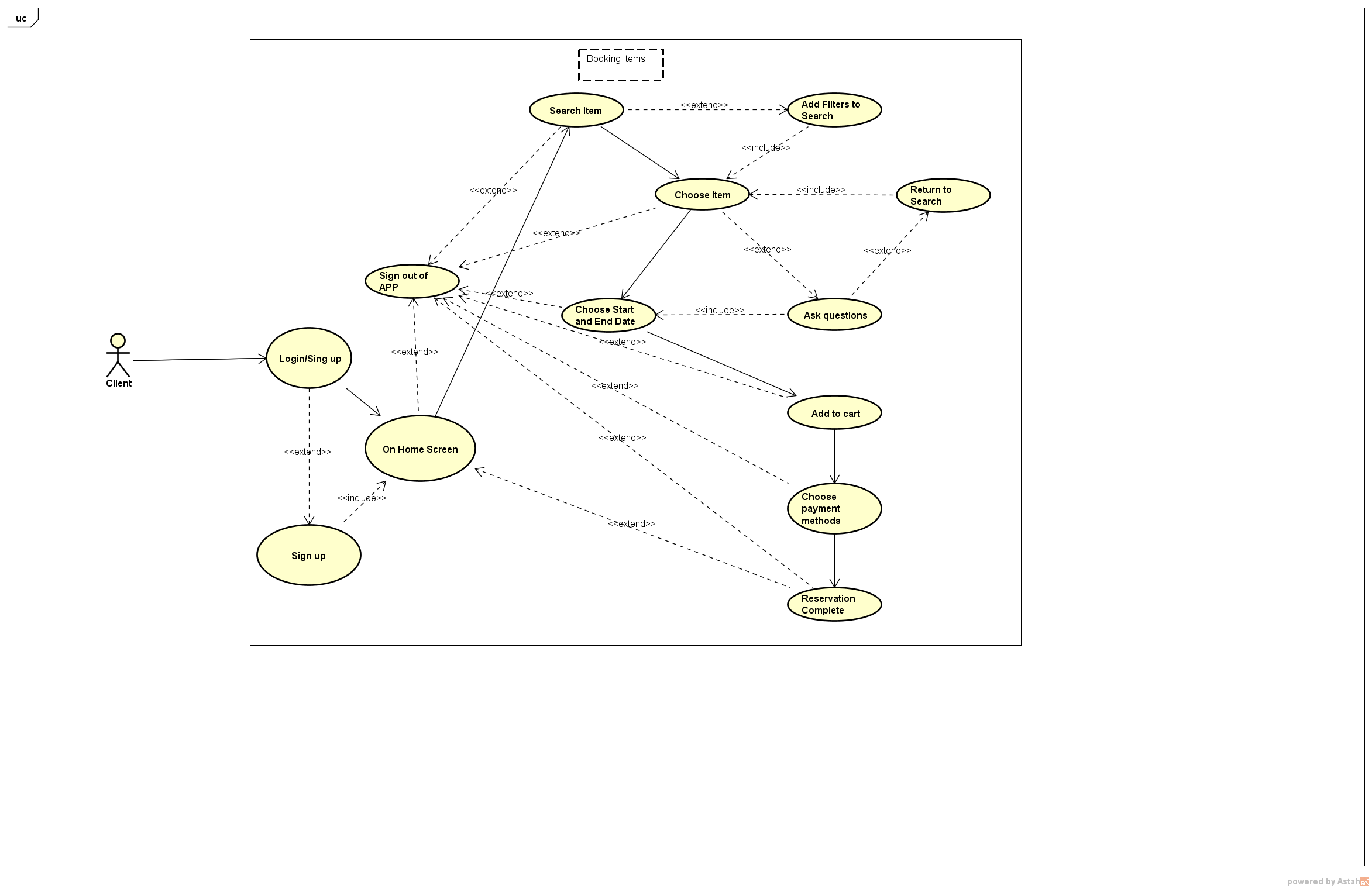
4 O Sistema deverá ser disponível para iOS, Android.

5 O Sistema deverá contar com uma interface simples e de fácil entendimento.

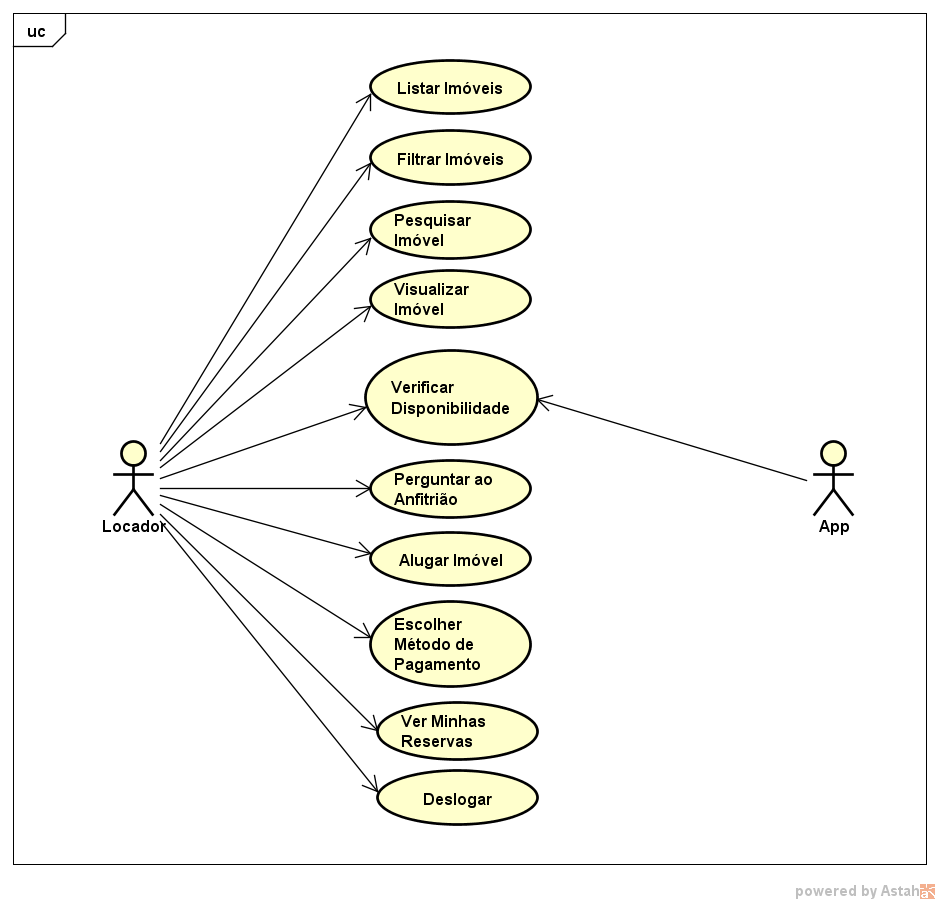
6 O Sistema precisará estar ativo sempre.

# Visão de Casos de Uso

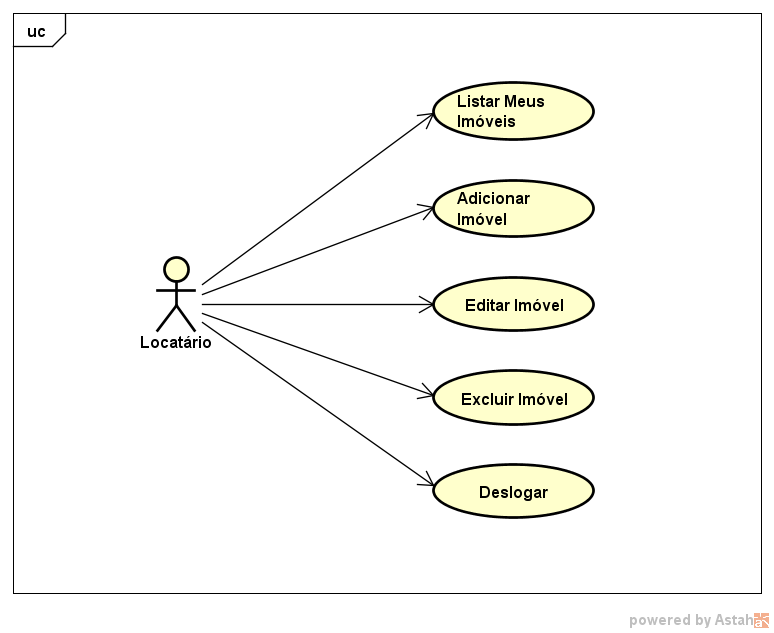
* 1. **Caso de uso generalizado**



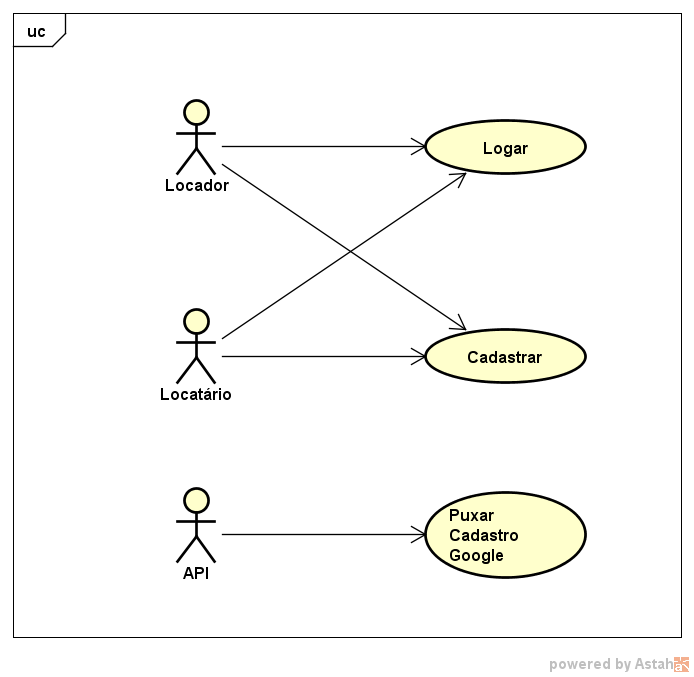
* 1. **Caso de uso tela inicial locador**



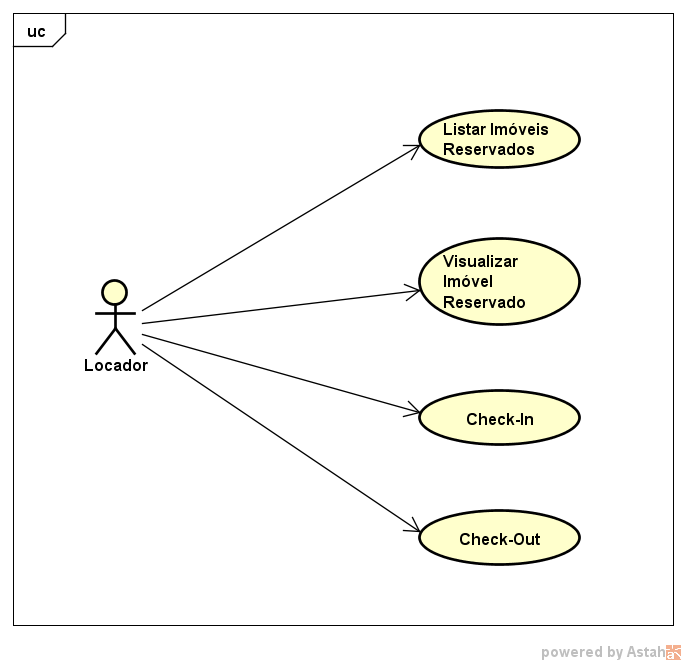
* 1. **Caso de uso tela inicial locatário**



* 1. **Caso de uso Login/Cadastro**



* 1. **Caso de uso Minhas Reservas**



# 

**Documento de Visão Arquitetural de Software**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <31/10/2020> | <1.0> | <Iniciado o documento, Visão lógica, Visão Processos, Visão Implantação, Visão Implementação> | <Rogerio Amorim> |

# Visão Lógica

As partes significativas do ponto de vista da arquitetura do modelo de design, são :

Visão de segurança: é responsável pela garantia de que o sistema é seguro, para que o usuário não tenha dúvidas sobre a segurança do sistema ou deixe de usar o sistema por sentir que não é seguro.

Dentro dessa visão há um Subsistema de design de segurança, dentro desse subsistema os pacote de segurança e dentro deste pacote as classe:

* SERVIÇO: é a classe responsável pelo desempenho do sistema, sendo assim é responsável pela funcionalidade, segurança e eficiência do sistema.

Visão de usabilidade: é responsável pela interface do sistema se mostrar usual ao cliente, e que ele não tenha dificuldade na hora que for usar o sistema.

Dentro dessa visão há um Subsistema de design de usabilidade, dentro desse subsistema os pacotes de usabilidade e dentro deste pacote as classes:

* API: é a classe responsável pelo que o usuário requisita e vê do sistema, por isso ela é responsável pela usabilidade do sistema.

Visão de confiança: é responsável pela confiabilidade do sistema, assim sendo específica as características que o sistema deve possuir para ser considerado confiável.

Dentro dessa visão há um Subsistema de design de usabilidade, dentro desse subsistema os pacotes de usabilidade e dentro deste pacote as classes:

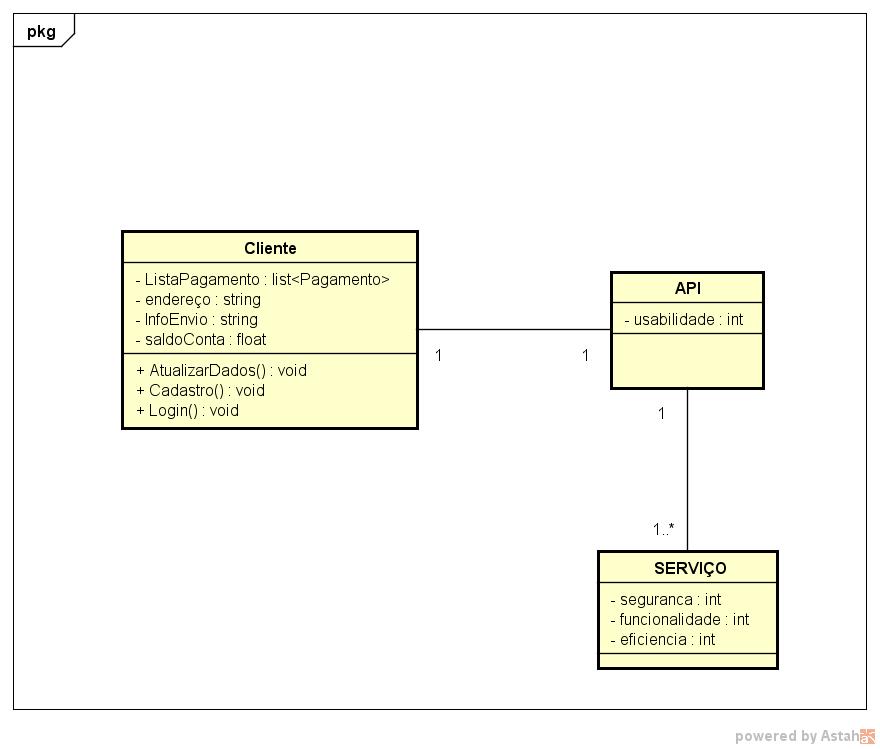
* SERVIÇO: é a classe responsável pelo desempenho do sistema, sendo assim é responsável pela funcionalidade, segurança e eficiência do sistema.

## Visão Geral

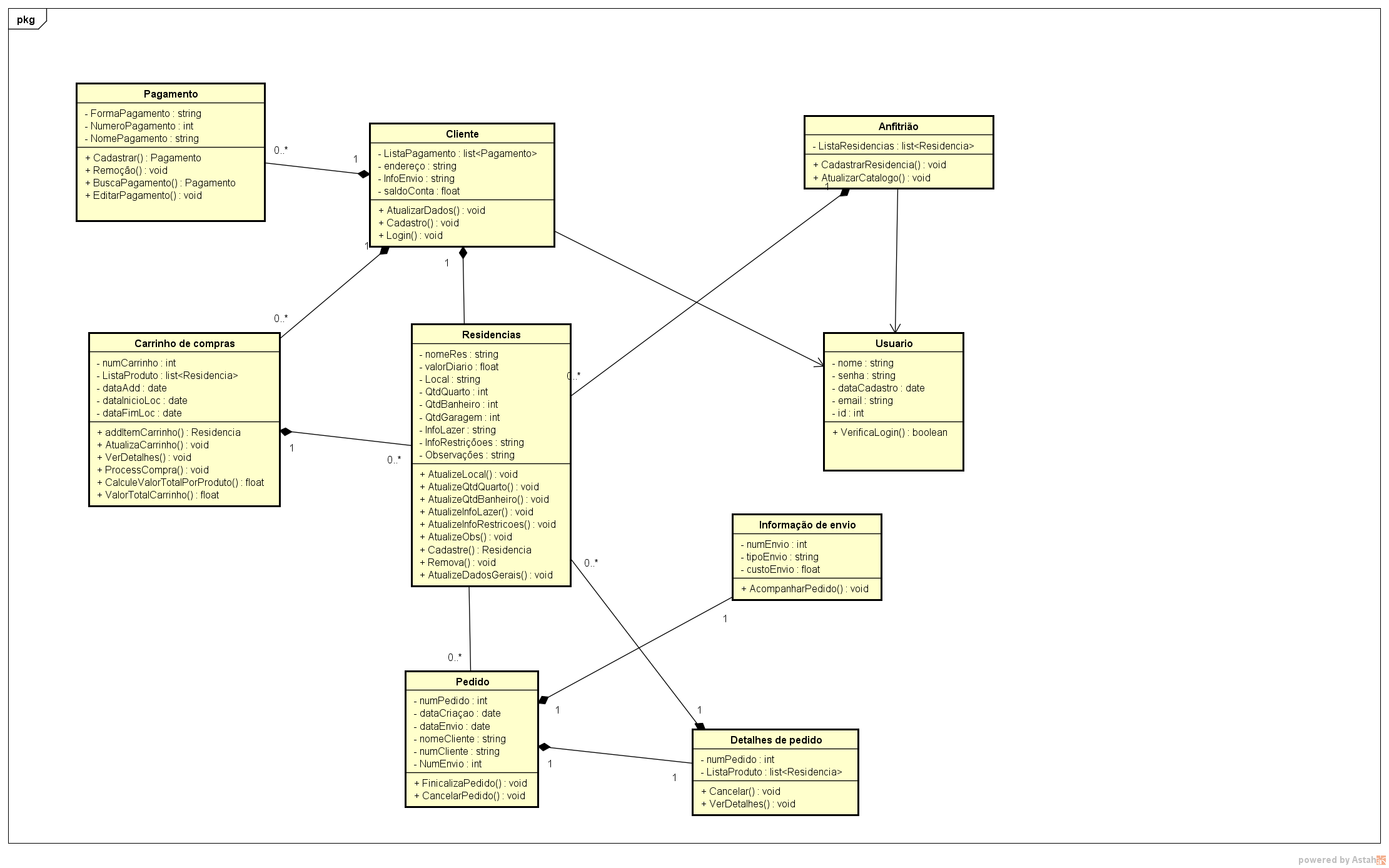
Visão geral do design é baseado na usabilidade, segurança e confiança, desta forma são divididos em três pacotes de nível superior, o pacote da visão de usabilidade, o pacote da visão de confiança e o pacote da visão de segurança.

Cada um deles tem o seu subsistema de design o subsistema da usabilidade, o subsistema da confiança e o subsistema da segurança, dentro de cada subsistema tem estão os pacotes, os quais são, respectivamente : pacote de segurança, que tem a classe SERVIÇO; pacote de usabilidade, que tem a classe API; pacote de confiança, que tem a classe API.

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

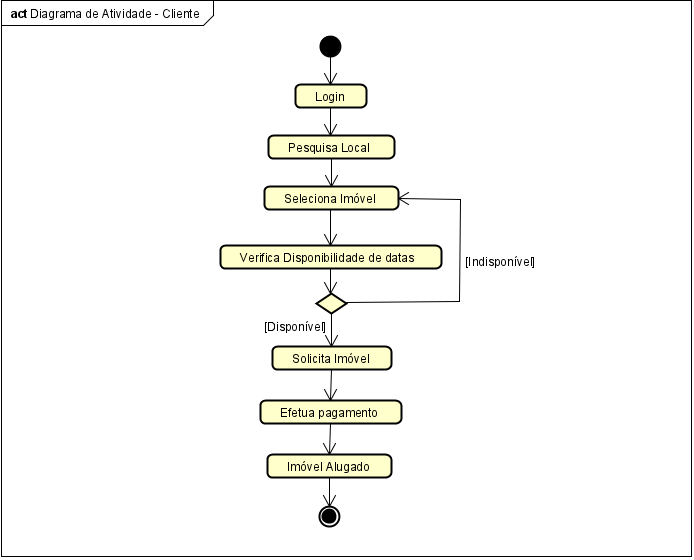


* 1. **Diagrama de classes**

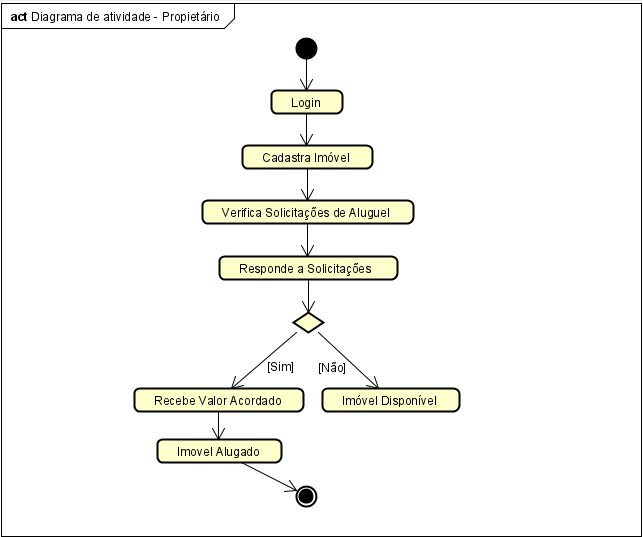
****

# Visão de Processos

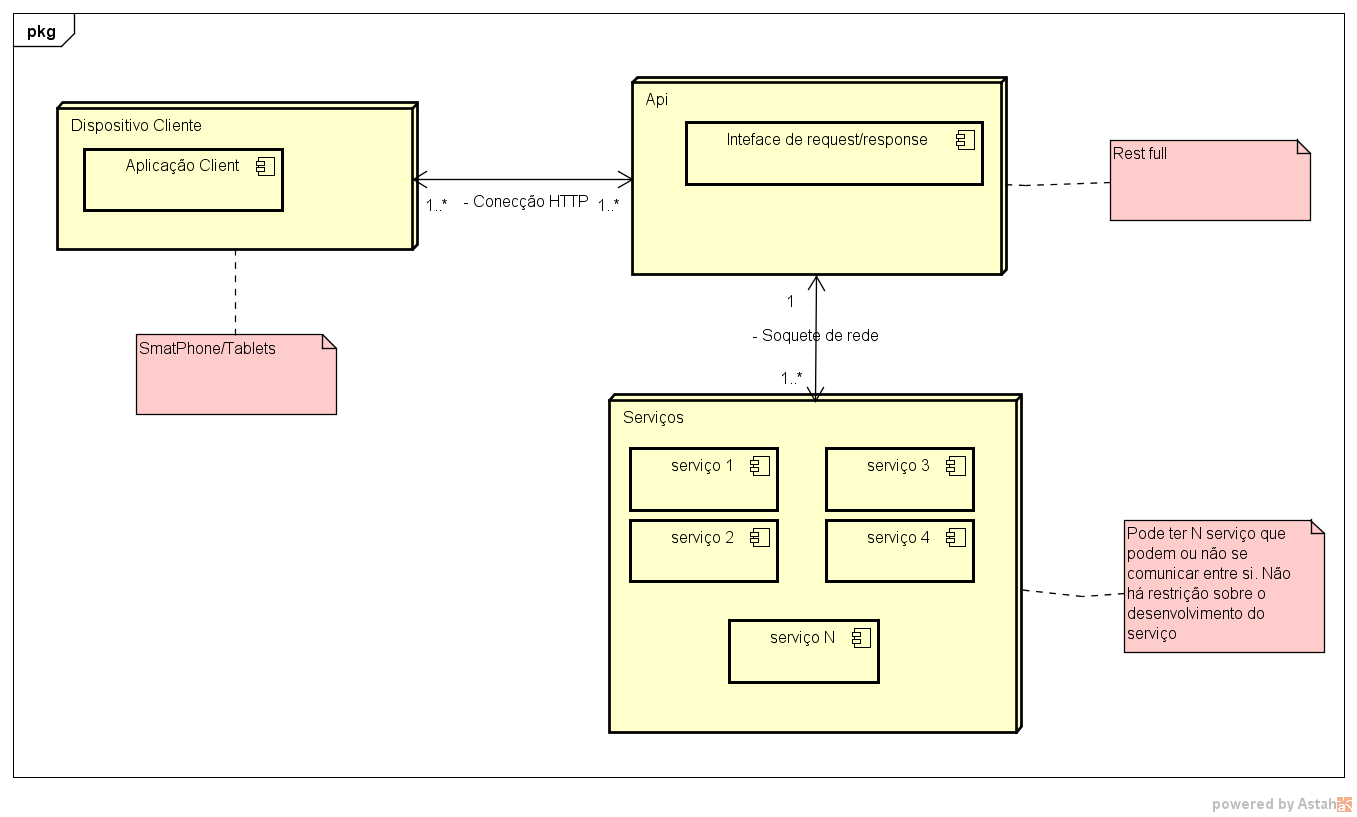
* 1. **Visão do locador**

**

* 1. **Visão do locatário**

****

# Visão de Implantação

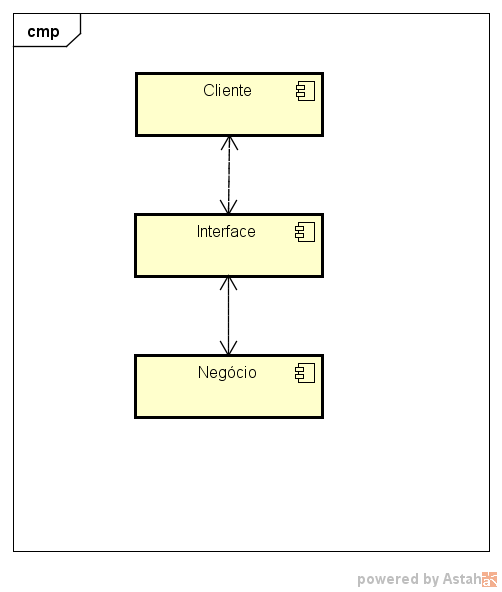
**

# Visão da Implementação

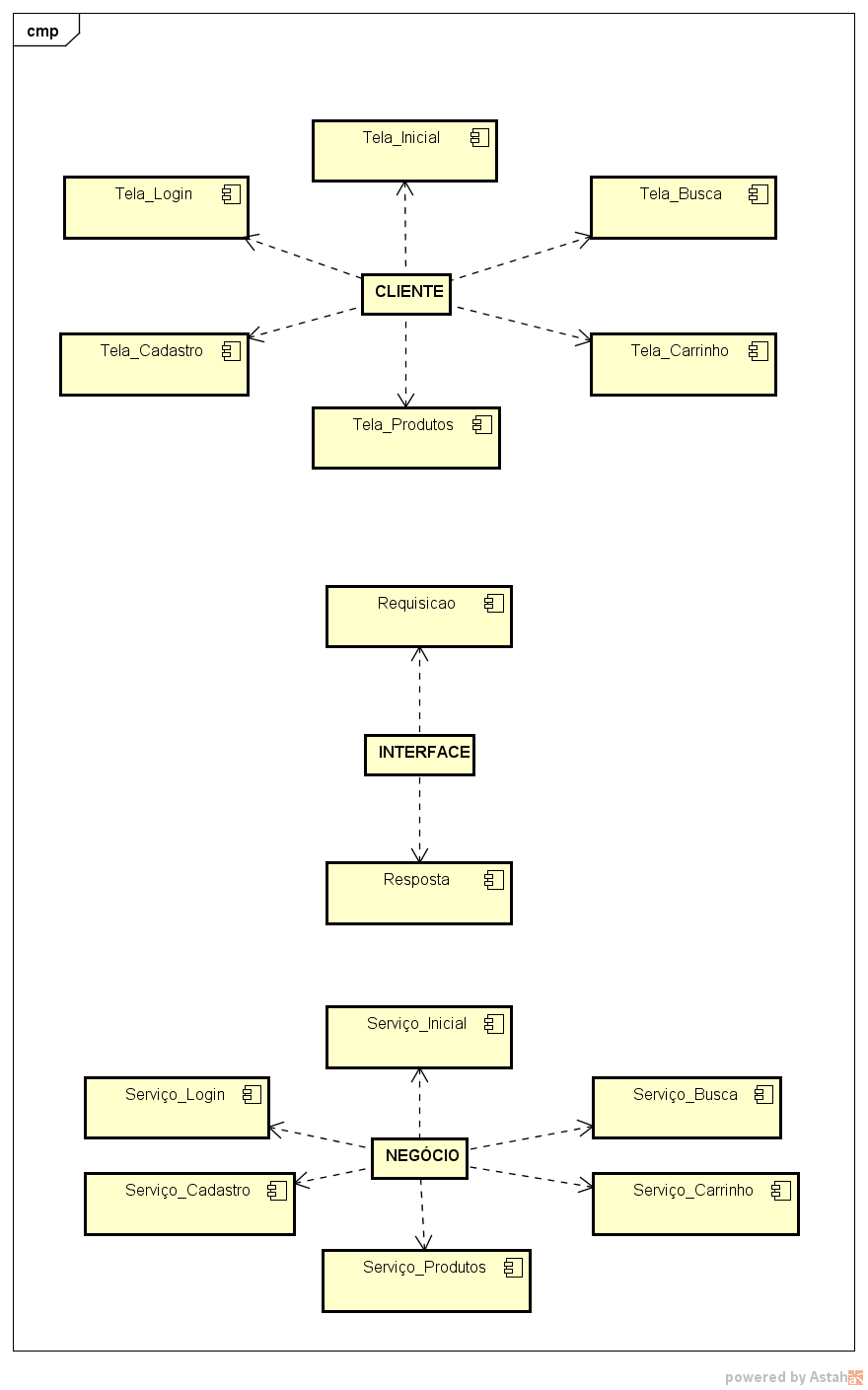
O modelo geral foi divido em 3 camadas, sendo elas: camada cliente, camada de interface e camada de negócio . Os componentes mais significativos são aqueles que possuem total foco do sistema, são aqueles que desempenham a função principal do sistema.

## Visão Geral

* Camada de Cliente - Camada de Interface : onde acontece a interação “principal”, a camada de cliente é onde o cliente faz as requisições dos serviços para camada de interface que é onde fica a api de redirecionamento de requisições e respostas dos respectivos serviços .
* Camada de Interface - Camada de Negócio : é a interação onde a camada de interface fornece as requisições aos devidos serviços da camada de negócio, onde acontece a troca de dados, dados salvos são “recuperados” para utilizar durante a interação com o usuário e dados onde dados gerados são armazenados.

**

## Camadas



**Documento de Design Arquitetural**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <31/10/2020> | <1.0> | <Iniciado o documento, descrição dos design, modelo de design arquitetural> | <Rogerio Amorim> |

# Descrição do Design

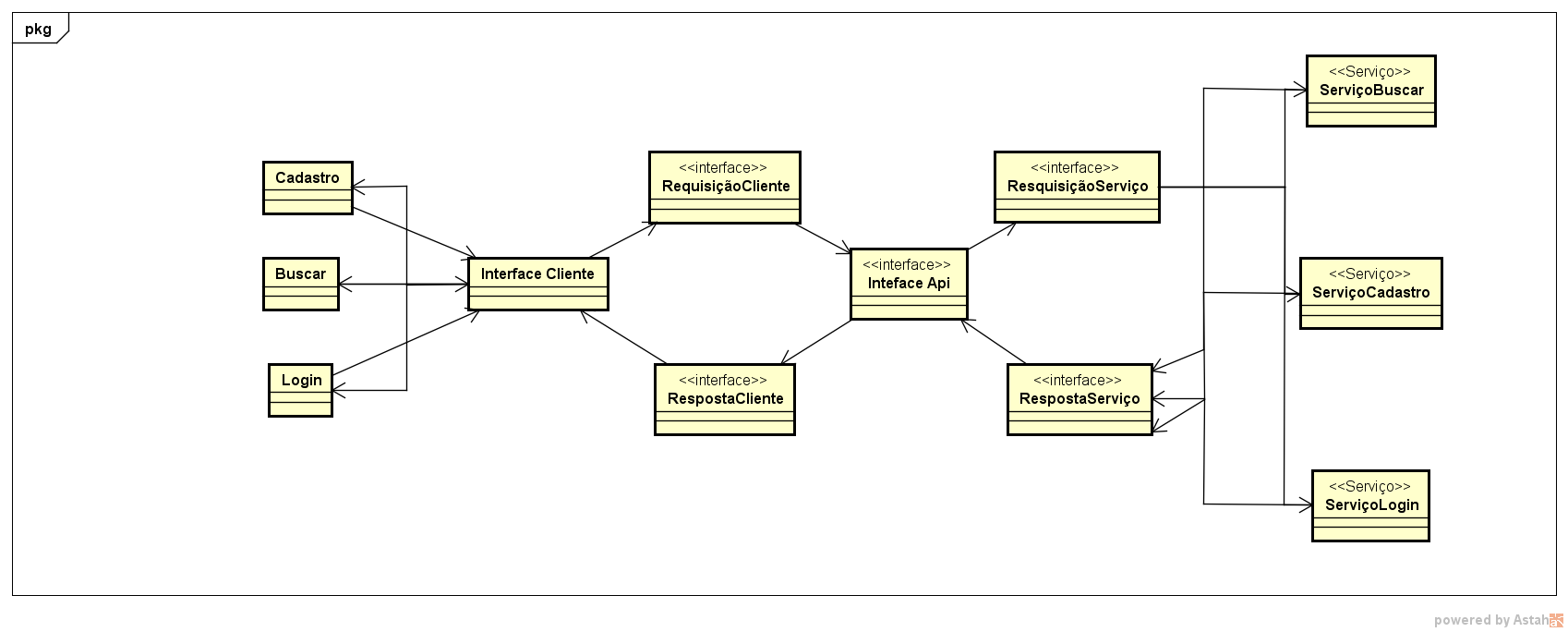
O software foi projetos para atender as demandas e evoluir sem problemas futuros. A arquitetura escolhida foi a de Micro Serviço.

Temos o lado cliente onde é responsável por fazer as requisições para resoluções das atividade definidas.

Em seguida temos a Api de gerenciamento, que tem a responsabilidade de receber as requisições do cliente e redirecionar para o/os serviços detentores do conhecimento para resolução da atividade, outra responsabilidade da Api é direcionar as respostas fornecidas pelos serviços até o cliente que fez as requisições.

Por fim temos a parte dos serviços, cada serviços tem uma unica reposabilidade para diminuir a complexidade, melhorar o reaproveitamento de código e facilitar manutenções. Caso necessário um serviço pode fazer requisições internas a outros serviços para poder entregar o resultado da requisição passada pela Api.

1. **Modelo de Design Arquitetural**

****

**Documento de Decisões Arquiteturais**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <01/11/2020> | <1.0> | <Iniciado o documento, apresentação do negócio, apresentação de tecnologia.> | <Rogerio Amorim> |

# Apresentação do Negócio

O ideia de negócio proposta, trata-se de um sistema móvel para locação de residências/flats/studios de forma ágil e sem burocracias, onde o locatário se comunica direto com lacador.

A arquitetura proposta é a de micro serviços, devido a sua praticidade, flexibilidade e escalabilidade.

Para o desenvolvimento da ideia proposta, foram levantadas 2 estilos arquiteturais;Orientada a objetos; Restful API.

Orientada a objetos agrega os seguintes atributos ao projeto. Usabilidade, Manutenibilidade

Recuperabilidade,Tolerância a falhas e Confiabilidade.

Restful API agrega os seguintes atributos ao projeto, Portabilidade, Disponibilidade, Eficiência de desempenho, Compatibilidade.

# Apresentação de tecnologias

Neste projeto usaremos Dart/Flutter como linguagem principal e Firebase como via de armazenamento de arquivos e banco de dados.

**Documento de Análise Arquitetural**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <31/10/2020> | <1.0> | <Iniciado o documento,.> | <Rogerio Amorim> |

# Apresentação do Negócio

O ideia de negócio proposta, trata-se de um sistema móvel para locação de residências/flats/studios de forma ágil e sem burocracias, onde o locatário se comunica direto com lacador.

1. **Apresentação da Arquitetura**

A arquitetura proposta é a de micro serviços, devido a sua praticidade, flexibilidade e escalabilidade.

1. **Identificação de Estilos Arquiteturais**

Para o desenvolvimento da ideia proposta, foram levantadas 2 estilos arquiteturais:

* Orientada a objetos
* Restful API

1. **Atributos de Qualidade**

A seguir os atributos de qualidade que a arquitetura suporta:

* **Eficiência de desempenho**: Desempenho em relação à quantidade de recursos usados nas condições declaradas.
  + Latência
  + Perda de dados
* **Compatibilidade:** Grau com o qual um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes e / ou executar suas funções necessárias, enquanto compartilha o mesmo ambiente de hardware ou software; Envolve interoperabilidade
  + Disponível em várias aparelhos
* **Usabilidade**: Grau com que um produto ou sistema pode ser usado por usuários específicos para atingir metas especificadas com eficácia, eficiência e satisfação em um contexto específico de uso.
  + UX
  + Tratamento de erros/exceção
* **Confiabilidade:** Grau com que um sistema executa funções especificadas sob condições especificadas por um período de tempo especificado.
  + Segurança de dados (LGPD)
* **Disponibilidade:** Grau em que um sistema, produto ou componente está operacional e acessível quando necessário para uso.
  + Tempo disponível
  + Tempo de inicialização
* **Tolerância a falhas:** Grau para o qual um sistema, produto ou componente opera conforme pretendido, apesar da presença de falhas de hardware ou software.
  + Registro de logs
  + Tratamento
* **Recuperabilidade**: Grau com que, no caso de uma interrupção ou falha, o sistema pode recuperar os dados diretamente
  + Recuperação de estado
* **Manutenibilidade:**
  + Responsabilidade única
  + Baixo acoplamento
* **Portabilidade:**
  + Dispositivo móvel (smartphone)

1. **Análise de estilos arquiteturais**

* **Orientada a objetos**
  + Usabilidade
  + Manutenibilidade
  + Recuperabilidade
  + Tolerância a falhas
  + Confiabilidade
* **Restful API**
  + Portabilidade
  + Disponibilidade
  + Eficiência de desempenho
  + Compatibilidade

1. **Apresentação de Resultados**

Forem levantados os riscos e benefícios da arquitetura escolhida, assim como mapeados os atributos exigidos e estilos arquiteturais que os suportam.

**Documento de DTD**

**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <01/12/2020> | <1.0> | <Iniciado o documento, xml-residência, xml-cliente, xml-schema-cliente, xml-schema-residência> | <Rogerio Amorim> |

# Xml Cliente

<?xml version="1.0"?>

<Cliente>

<nome>

Nome aqui

</nome>

<dataNascimento>

Data nascimento aqui

</dataNascimento>

<endereco>

Endereço aqui

</endereco>

<AlugarCasa>

</AlugarCasa>

<MeusPedidos>

</MeusPedidos>

</Cliente>

# Xml Schema Cliente

<?xml version="1.0"?>

<schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com" elementFormDefault="qualified">

<element name="Cliente">

<complexType>

<sequence>

<element name="nome" type="xs:string" />

<element name="dataNascimento" type="xs:date" />

<element name="endereco" type="xs:string" />

</sequence>

<element name="AlugarCasa">

</element>

<element name="MeusPedidos">

</element>

</complexType>

</element>

</schema>

# Xml Residência

<?xml version="1.0"?>

<Residencia>

<endereco>

Endereço aqui

</endereco>

<dataDisponibilidade>

Data de disponibilidade aqui

</dataDisponibilidade>

<preco>

Preço aqui

</preco>

<VerificarDisponibilidade>

</VerificarDisponibilidade>

</Residencia>

# Xml Schema Residência

*<?xml version="1.0"?>*

*<schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.w3schools.com" xmlns="http://www.w3schools.com" elementFormDefault="qualified">*

*<element name="Residencia">*

*<complexType>*

*<sequence>*

*<element name="endereco" type="xs:string" />*

*<element name="dataDisponibilidade" type="xs:date" />*

*<element name="preco" type="xs:double" />*

*</sequence>*

*<element name="VerificarDisponibilidade">*

*</element>*

*</complexType>*

*</element>*

*</schema>*

**Documento de Modelagem Web Services**

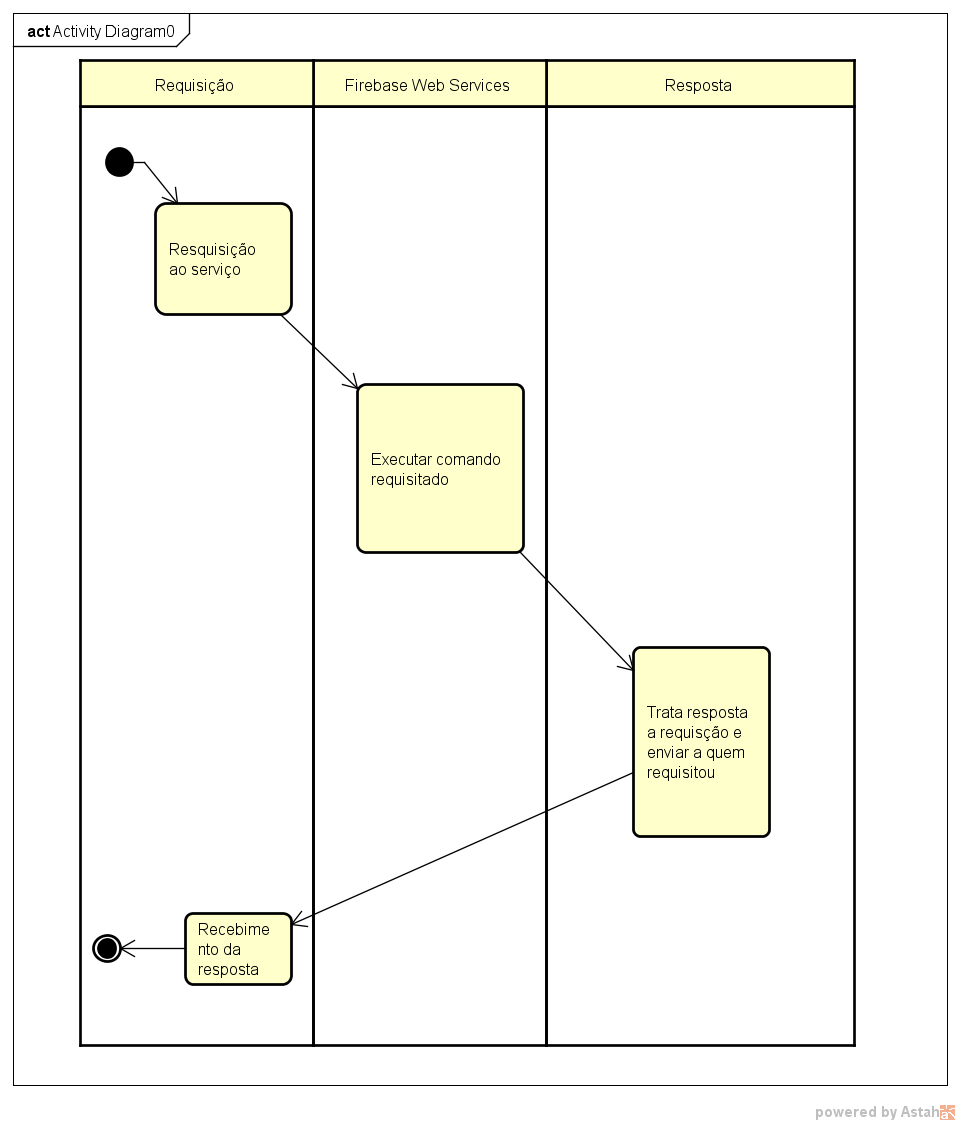
**Histórico da Revisão**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <01/12/2020> | <1.0> | <Iniciado o documento,descrição modelagem, diagrama modelagem> | <Rogerio Amorim> |

# Descrição modelagem

Será usado o Web Service Firebase para Manipulação e Armazenamento dos dados. No diagrama a seguir mostramos como será feita as requisições, tratamentos de requisições e respostas.

# Diagrama modelagem

****