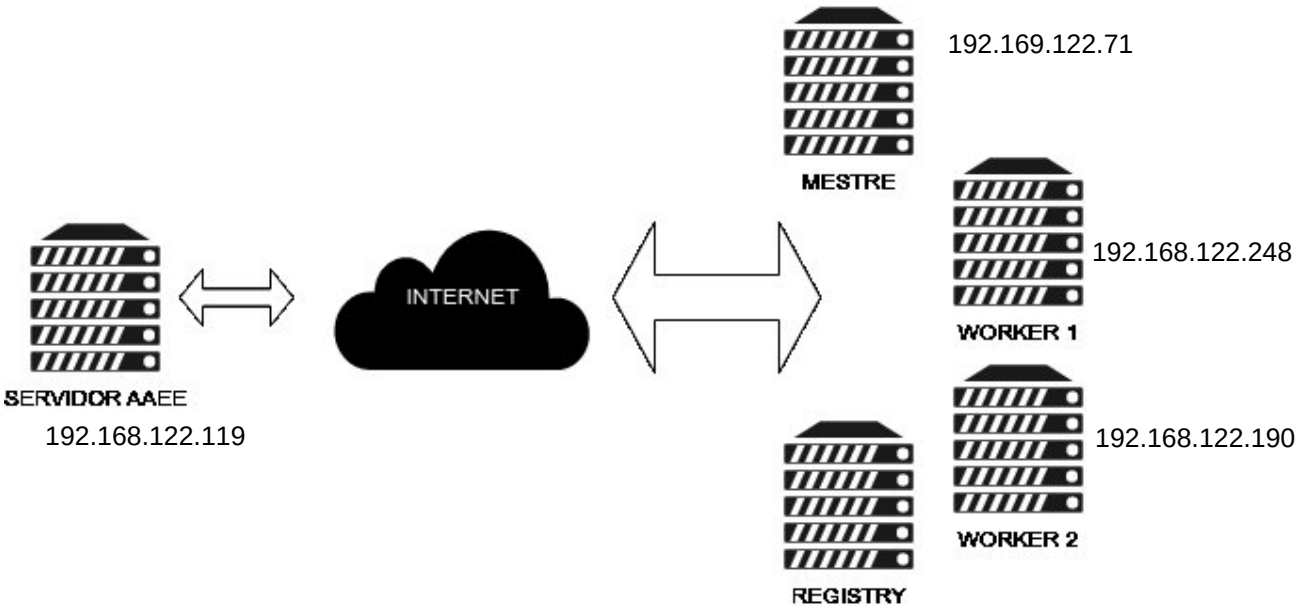
Proposta de Arquitetura de implantação dos servidores

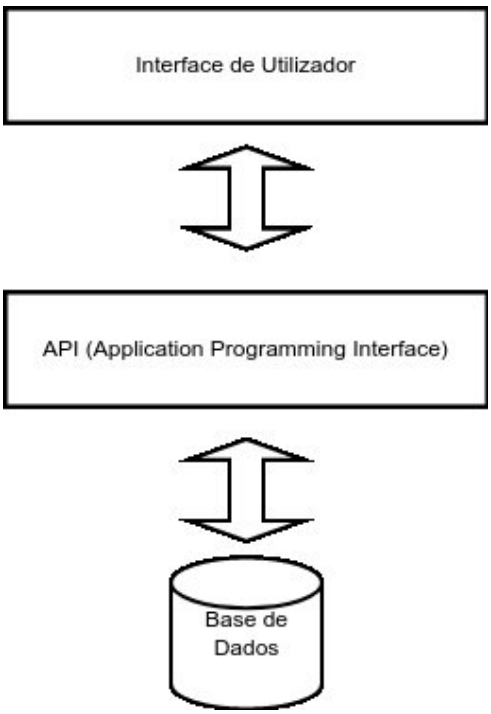
Para o funcionamento do sistema de replicação será necessário ter um cluster de servidores fora da rede da AAEE. Este cluster de servidores pode ser composto por maquinas virtuais ou maquinas fisicas que podem estar localizadas na mesma rede ou em redes destintas deste que haja conectividade entre elas.



Na arquitetura anterior podem existir mais de 2 maquinas worker. As maquinas Mestre e Registry são as mais importantes para a funcionamento da aplicação de replicação de aplicações web, pois, é apartir delas que a aplicação interage com a infraestrutura.

Proposta de Arquitetura para aplicação

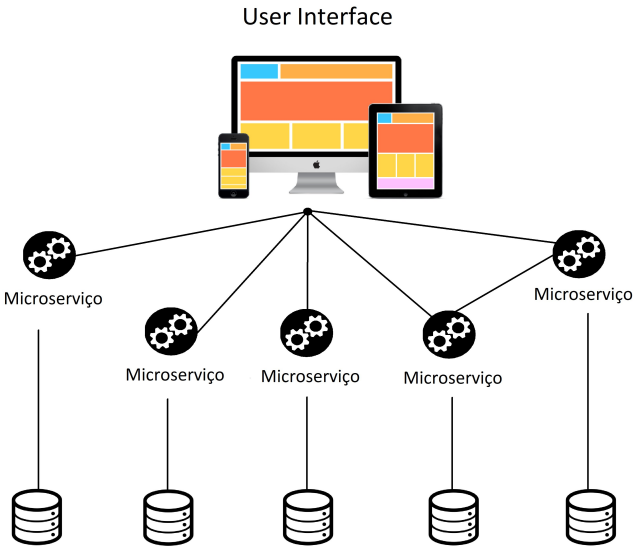
De forma generica, a arquitetura da proposta para a aplicação esta divida em 3 camadas nomeadamente cuja manutenção pode ser feita de forma independente.



- **Aplicação Cliente** - É uma aplicação web que interage com os clientes por meio de um navegador;
- **API** – Recebe os comandos enviados pela aplicação web e os envia para o cluster ou manipula a base de dados quando é o caso.
- **Base de dados** - É o local onde são armazenados os dados gerados pela aplicação.

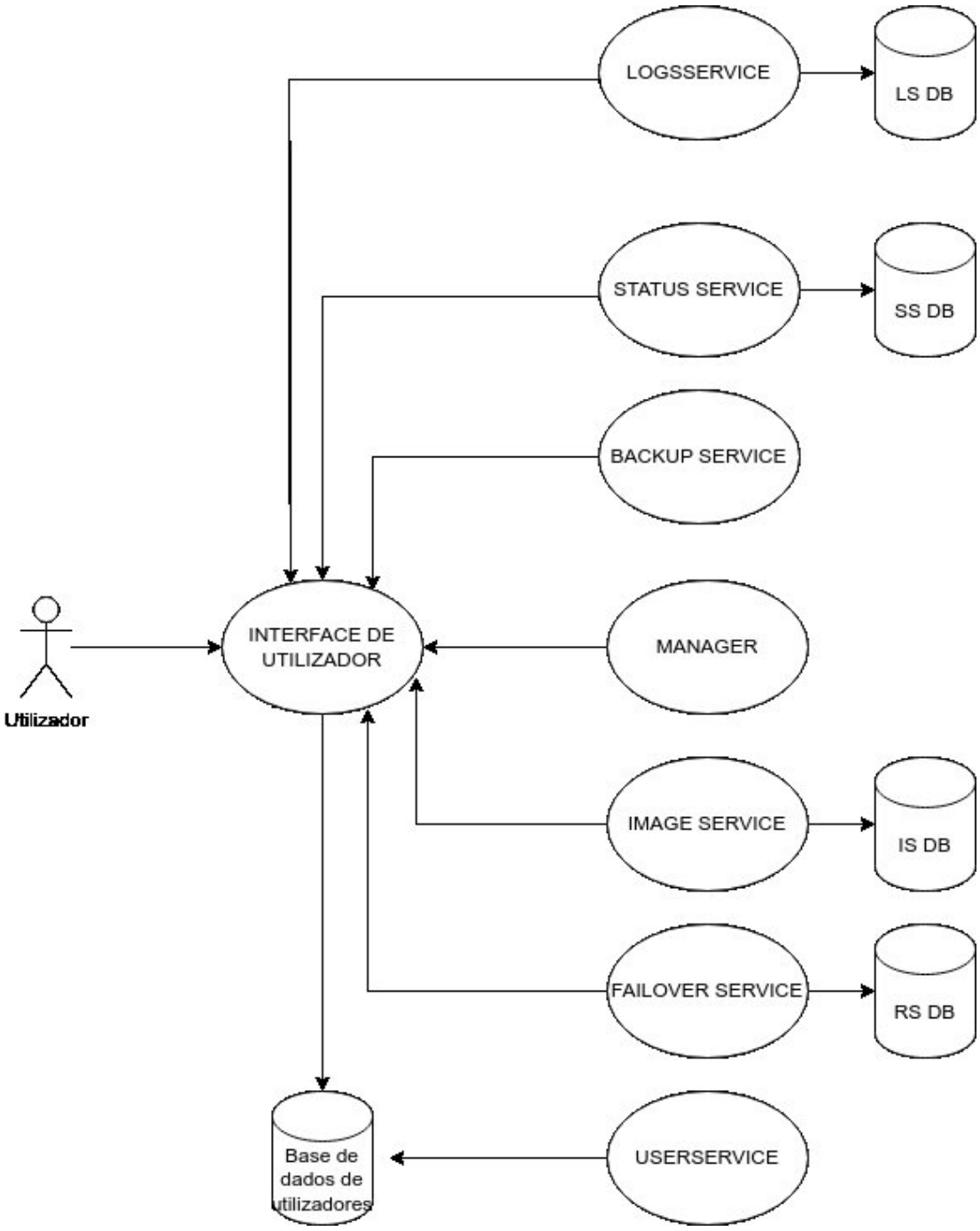
A API esta divida em partes independentes e, a sua a arquitetura foi projecta para que seja resilente, escalavel e, que possa suportar grande de trafego e processamento sem comprometer o desempenho. A arquitetura da API é baseada em microserviços.

Segundo (https://aws.amazon.com/pt/microservices/) microsserviços são uma abordagem arquitetônica e organizacional do desenvolvimento de software na qual o software consiste em pequenos serviços independentes que se comunicam usando APIs (Application Programming Interface) bem definidas. Esses serviços pertencem a pequenas equipes autossuficientes.



Fonte=(https://filipesribolli.wordpress.com/2019/05/09/introducao-a-arquitetura-de-micro-servicos-vantagens-e-desvantagens/)

O sistema de replicação esta divido em pequenas partes que funcionam de forma independente e com funções distinhas. Cada “pedaço” da aplicação, pode ser gerido por uma equipa deferente e, pode apresentar tecnologias diferentes. Na arquitetura da aplicação o processo de manutenção também será mais facil e, não será necessário interromper todo o programa para fazer manutenção de um unico microserviço.



A figura anterior mostra a arquitetura proposta para o sistema de replicação. Na arquitetura proposta, o utilizar interage com a interface do utilizador que por sua vez interage com os microserviços, ou seja, cada funcionalidade da interface do utilizador é mapeada para um microserviço e, por sua vez, os microserviços são independentes entre si. Deste modo, a indisponibilidade de um dos microserviços não afeta o funcionamente da aplicação como um todo.

Descrição dos Componentes da Arquitetura

Interface de utilizador – Os utilizadores do sistema irão interagir com um sistema por meio de uma aplicação web que será usada atravez do navegador é apartir da interace do utilizador que os comandos serão enviados para os microserviços.

Statusservice – é o microserviço responsavel pelo monitoramento dos servidores e criação dos servidores monitorados. É por meio dele que é possivel saber se um determinado servidor monitorado encontra-se offline.

Backupservice – é o microserviço responsavel por realizar a copia de dados de um servidor para um volume. É por meio dele que se são criados os volumes e containers de sincronização;

Manageservice - é o microserviço que faz a gestão do cluster e de toda infraestrutura, cria os serviços e redes do cluster.

Imageservice - faz a gestão de imagens do registry, cria containers de aplicação, guarda containers como imagens, envia imagens par ao registry.

Failoverservice - é por meio dele que se configura o failover. Este microserviço é reponsavel por activar os serviços quando o failover acontece. Para que ele funcione, é necessário que o Manageservice esteja activo.

Userservice - este microserviço é reponsavel pela gestão de utilizadores.

Logsservice - é respondavel por guardar todos os eventos do sistema.

Principais Tecnologias Usadas

Interface do utilizador

A interface do utilizador foi desenvolvida usando o Django. O Django é um framework escrito em python para desenvolvimento de aplicações web. O django tem uma arquitetura modular e que permite desenvolver aplicações web robustras. O motivo para escolha do Django é a a sua facilidade e arquitetura clara que, aliados a simplicidade e a possibilidade de usar bibliotecas do python torna o desenvolvimento web mais flexivel e rapido.

API

A API foi desenvolvida em Java e biblioteca Spring Framework. O Java é uma linguagem de programação consolidada no mercado e, é reconhecida pela sua segurança, desempenho e compatibilidade com a maioria dos dispositivos, é possivel desenvolver aplicações nativas da nuvem usando java. O Spring framework é um biblioteca que permite o desenvolvimento de Aplicações web.

O Java foi escolhido para o desenvolvimento da API porque com ele pode-se usar a biblioteca Spring-Eureka que é um framework robustro para desenvolvimento de microserviços. O Spring-Eureka tem a tecnologia Service Discover que faz com que, seja possível encontrar o endereço IP e a porta de um microservico com base no nome.

Uma parte da API foi escrita em python , especificamente a parte que faz a conecção com o Docker API. Os scrips python que enviam comandos a api docker não estão presentes em todos os microserviços.

Base de dados

Na presente proposta de solução foram usadas dois tipos de base de dados nomeadamente mongoDB e Mysql.

A base de dados depende do microserviço e existem microserviços que não usam base de dados. Para o microservico Userservice (responsável pela gestão de utilizadores) foi usada a base de dados mysql. Foi escolhido o mysql porque tem facil integração com o framework da interface de utilizador e, porque, base dados relacionais são melhores quando trata-se de persistência de dados. Para o Statusservice, imageservice, Failoverservice e Logsservice foi usada a base de dados mongoDB. A escolha do mongoDB para os microserviços mensionados anteriormente foi feita por causa da natureza não relacional dos dados gerados pelos microserviços e, por causa da facilidade que o mongodb tem em armazenar grandes volumes de dados sem comprometer o desempenho da base de dados.

Modelos do sistema proposto

Diagrama de entidades

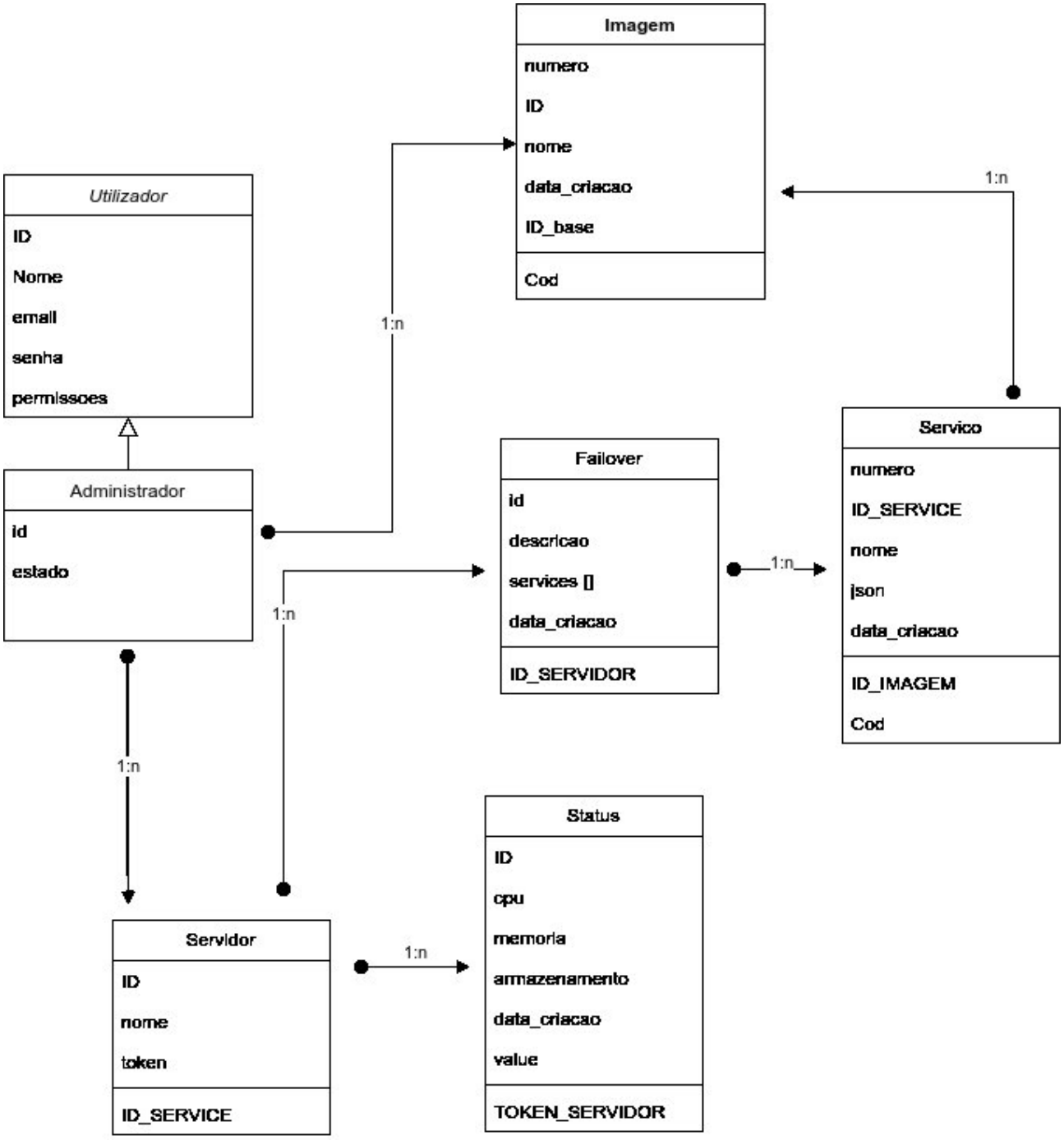
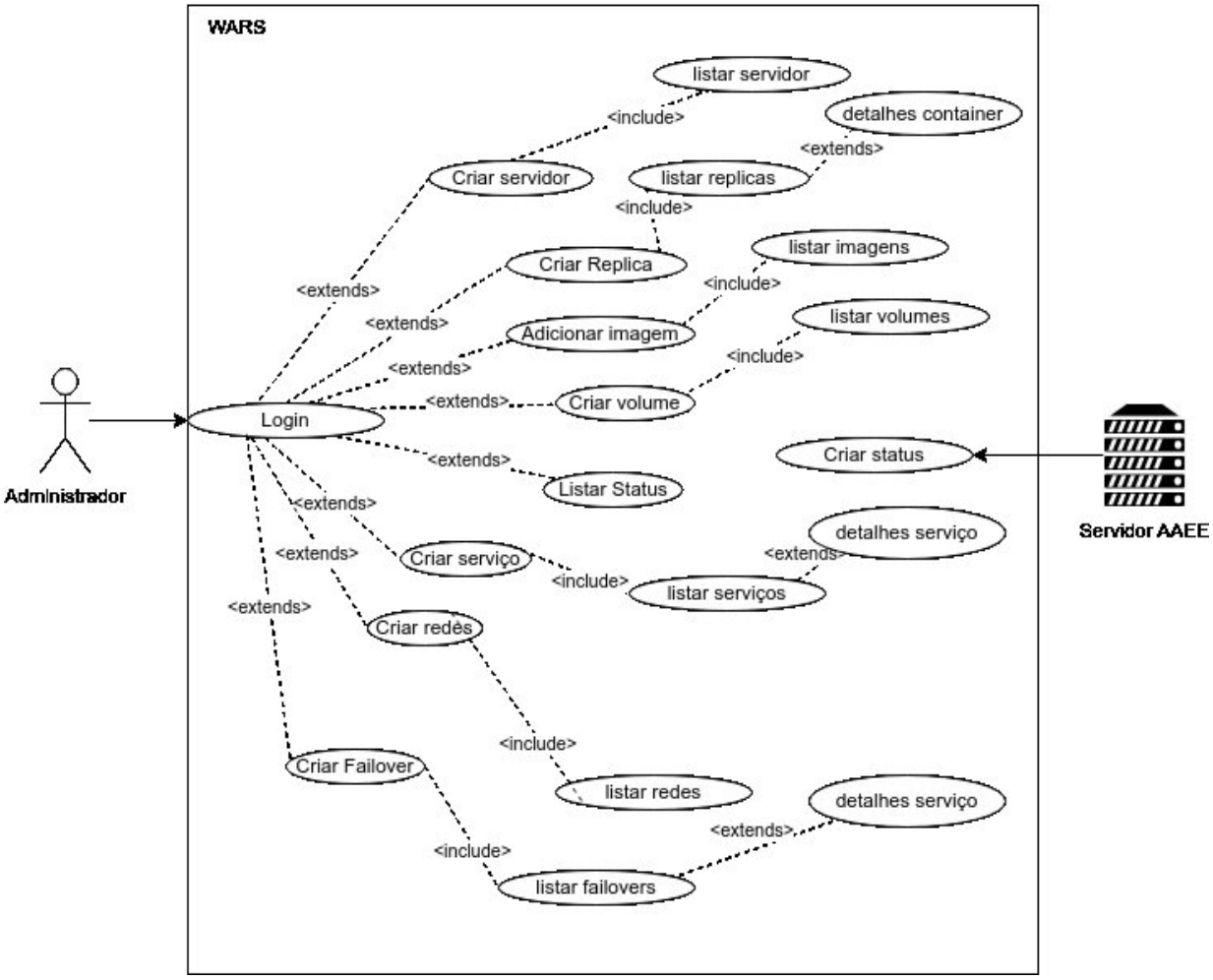
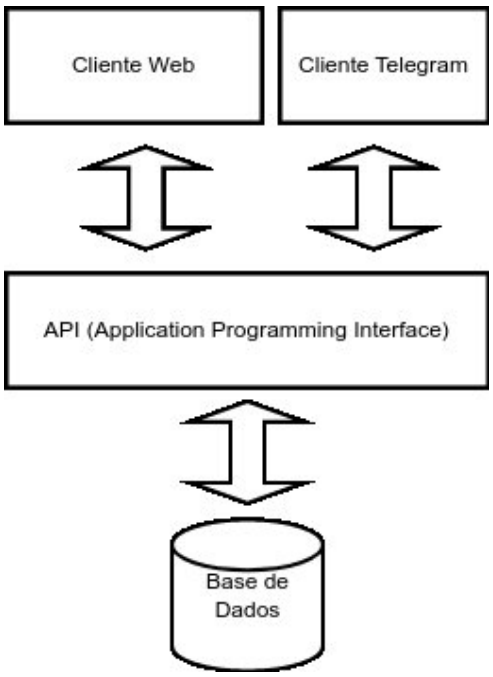


Diagrama se sequencias

Use case


WARS
listar servidor
<include>
detalhes container
<extends>
Criar servidor
listar replicas
<include>
Criar Replica
listar imagens
<include>
<extends>
<extends>
Adicionar imagem
listar volumes
<include>
<extends>
<extends>
Criar volume
Administrador
Login
<extends>
Criar status
Servidor AAEE
Listar Status
<extends>
<extends>
detalhes serviço
<extends>
Criar serviço
<include>
listar serviços
<extends>
Criar redes
<include>
Criar Failover
listar redes
detalhes serviço
<extends>
<include>
listar failovers


Cliente Web
Cliente Telegram
API (Application Programming Interface)
Base de
Dados