Alunos: Ewelly Fabiane Sousa Cunha, João Pedro Oliveira Silva, Miller Raycell Monteiro Correia, Pedro Aleph Gomes De Souza Vasconcelos, Rodrigo de Andrade Rolim Bem

Professor: Leandro Nelinho Balico

Disciplina: Sistemas Distribuídos

Relatório Perdidogs

Perdidogs é um projeto desenvolvido com o objetivo de servir como plataforma para a divulgação de animais perdidos, a divulgação do animal funciona da seguinte forma, pelo Telegram, a pessoa que encontrar o animal terá a opção de tirar fotos do animal, e anexar a localização atual do animal, o bot irá realizar uma postagem no Twitter indicando que um animal foi encontrado, o tweet terá as fotos do animal e a localização, a plataforma também terá um site que possui um mapa com os animais cadastrados, e também terá uma opção de ver os animais que foram dados como desaparecido, depois de 10 dias o animal será apagado do sistema, para evitar dados repetidos.

A aplicação é dividida em 4 containers, um para a gerência dos serviços relacionados as redes sociais, um para serviço de frontend web, um para o backend web e por fim um para o banco de dados, as linguagens utilizadas foram python 3.8 para o backend web e para o controle das redes sociais, para o frontend web foi utilizado NodeJs, usando Typescript e React, o banco utilizando é o Mongodb, a aplicação foi transformada em containers usando Docker e Docker compose para a orquestração dos containers.

O primeiro container a ser analisado será o que controla as redes sociais, o controle é feito através das apis, *python-telegram-bot* e python-twitter, essas apis permitem que se controle do acesso as redes, para o bot do telegram foram desenvolvidos vários gatilhos, o primeiro é o */start* que é o comando de início do bot, ele irá saudar a pessoa ao bot e indicar o fluxo de execução, depois será solicitado que a pessoa encaminhe fotos do animal, depois a pessoa deverá encaminhar a sua localização, que ativará o gatilho que adicionará o animal no

banco de dados, e por fim será pedido que a pessoa insira o comando */tweet*, para ativar o gatilho que fará a postagem no Twitter usando a api.

O container que gerencia o backend web utiliza Flask, que são bibliotecas python que fazem a parte de web, o Flask cria as rotas para o acesso do Front, o backend, possui 2 métodos, um get, que será a listagem dos animais para que sejam mostrados no frontend web, uma rota que mostrar os animais perdidos mais próximos do usuário atual.

O container que gerencia o frontend web foi desenvolvido utilizando NodeJS, utilizando o framework o React para fazer a montagem do Frontend e utilizando a linguagem de programação TypeScript, a escolha de usar o React e não seguir utilizando o Flask como frontend, foi baseada nos recursos de design do framework, que permite que se tenha uma interface mais dinâmica e é mais familiar para a equipe. Para se fazer o mapa foi utilizado a biblioteca MapBox, que oferece uma precisão e um design muito interessante, o front recebe as informações do back, e os trata para que seja mais visual para os usuários.

O banco da escolha foi o Mongodb, pela sua praticidade no uso e a forma simples que ele permite que seja transformado em container, mas também a simplificação do tratamento de posições geográficas, a coleção utilizada possui o nome animais e os seus documentos terão os campos _id, user_id: esse campo é utilizado para fazer a busca no banco para carregar as coordenadas para a criação do link do google maps, campo criado automaticamente para cada documento pelo mongo, date: que é um campo que possui a data que o animal foi encontrado, imagens: um array que possui os links das imagens que foram enviadas pelo usuário ao bot, coordenadas: a posição na qual o animal foi encontrado, é dividida em latitude e longitude, nenhum dado de localização do usuário é exposto pela aplicação, quando o usuário aceita utilizar a aplicação ele concorda em dividir a localização atual do animal, o perdidogs não faz uso nenhum da localização do usuário, apenas a localização que foi enviada.

O orquestrador de containers utilizado foi o Docker Compose, através dele podemos fazer a organização do mesmo e controlar a escalabilidade da aplicação de acordo com a necessidade pode se ampliar os recursos específicos de cada container.

Para rodar a aplicação basta clonar o repositório, abrir um terminal e inserir ./run.sh, que o aplicativo baixará todas as dependências (pode demorar bastante tempo), e rodar o aplicativo