



Doguito Petshop

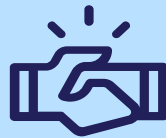
Sistemas Distribuídos

Apresentação do projeto final

Equipe



Natália Guimarães
496300



Rafael Gonçalves
495519



Roteiro

01

Ideia

03

Estrutura

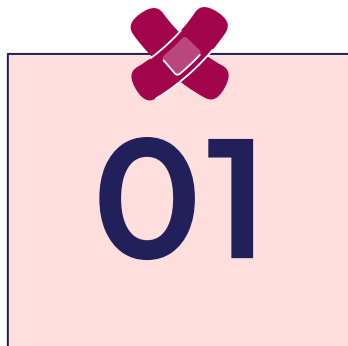
02

Arquitetura

04

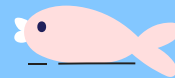
Código





Ideia





Projeto

A ideia do projeto consiste em um sistema de agendamento de consulta para pets. O sistema possui 2 tipos de usuários:

1. Colaboradores
2. Donos

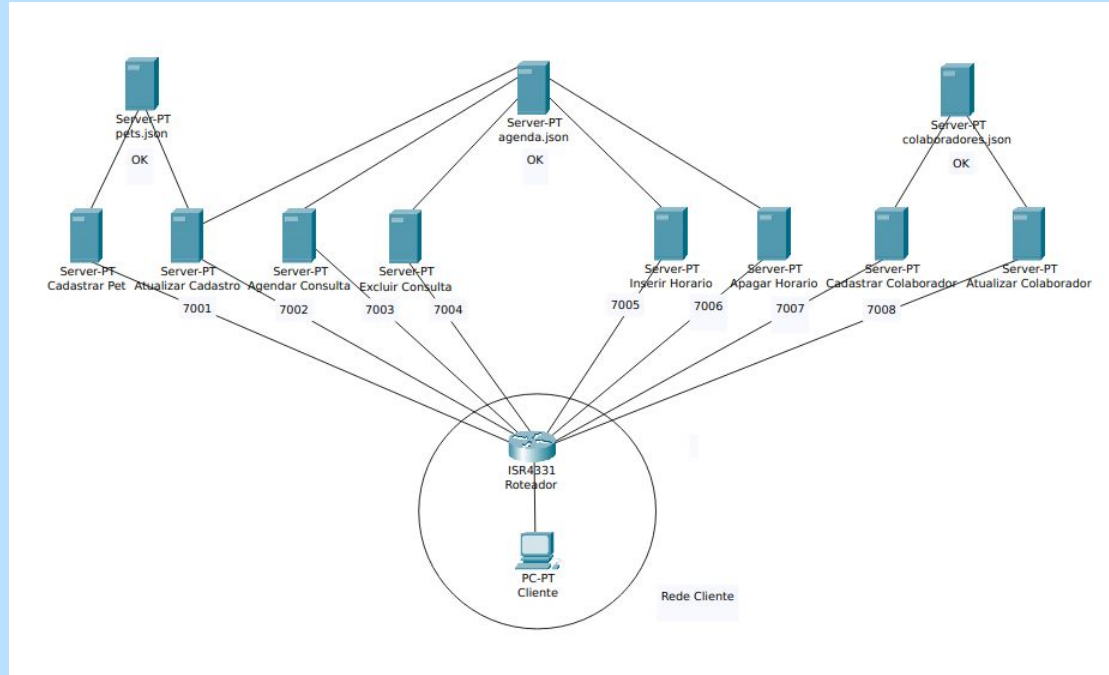


02

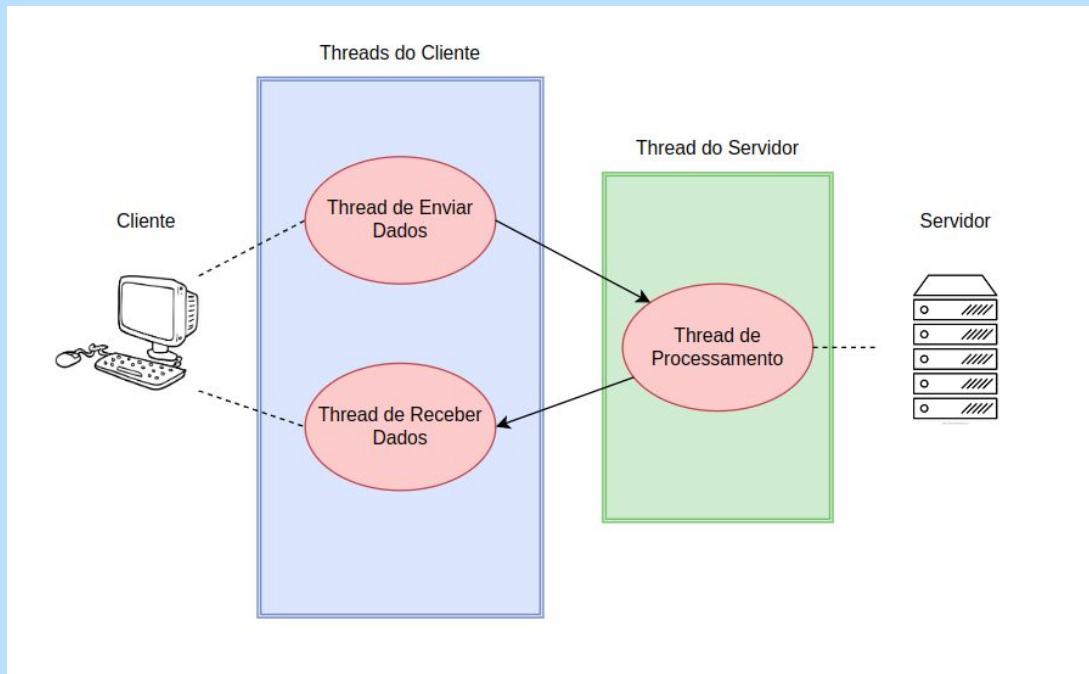
Arquitetura



Infraestrutura



Distribuição de Threads



Funcionalidades do Cliente

Uma breve explicação sobre as ações que são permitidas pelo cliente através dos servidores.



Cadastrar Pet

- Esse servidor tem a funcionalidade de cadastrar clientes no sistema.
- Ele está ativado na porta 7001.
- O cliente envia dados fundamentais para o cadastro como nome, CPF, CEP e etc.
- Esse servidor utiliza a API ViaCEP para preencher o endereço automaticamente por meio do CEP digitado.
- Cada cliente terá um ID gerado pela classe UUID do Java.



Atualizar Pet

- Esse servidor tem a funcionalidade de atualizar o cadastro já criado pelo cliente.
- O servidor está conectado na porta 7002.
- Esse servidor faz uma procura pelo cadastro do cliente no arquivo pets.json e atualiza os campos solicitados.
- A procura é feita através do login e senha enviados pelo cliente.
- O sistema não permite que 2 ou mais clientes possuam login iguais.
- Os dados permitidos que podem ser atualizados são: nome do pet, tipo do pet, telefone, email, endereço, número da casa e senha.
- O login não pode ser alterado.



Agendar Consulta

- A função desse servidor é permitir que um cliente solicite uma consulta para o seu pet.
- Para isso, o cliente deve enviar login, senha e CPF, mas também o código de consulta do horário desejado. Os dados do cliente são necessários para validar se o cadastro existe para marcar a consulta com o nome do pet, nome do cliente e CPF do cliente.
- O sistema gera códigos entre 1000 e 10000 e eles não se repetem para evitar que 2 ou mais horários disponíveis possuam o mesmo código.
- O servidor está conectado na porta 7003.
- Os horários disponíveis estão no arquivo agenda.json.



Excluir Consulta

- A função desse servidor é permitir que um cliente exclua um horário que ele marcou.
- Para isso, o cliente deve enviar login, senha e CPF, mas também o código de consulta do horário que deseja desmarcar a consulta.
- O servidor está conectado na porta 7004.



Funcionalidades do Colaborador

Uma breve explicação sobre as ações que são permitidas pelo colaborador através dos servidores.



Inserir Horário



- Esse servidor permite que o colaborador disponibilize um horário no sistema.
- Para isso, o cliente deve enviar login, senha e CPF para validação, uma vez só os colaboradores cadastrados é que podem inserir consultas. Além disso, deve ser enviado também os dados da nova consulta.
- Os registros são inseridos no agenda.json.
- O servidor está conectado na porta 7005.





Apagar Horário



- Esse servidor permite que o colaborador apague um horário do sistema.
- Assim como no servidor de inserir horário, o colaborador deve enviar para o servidor os dados da consulta que deseja excluir, mas também o seu login, senha e CPF para validação.
- O servidor está conectado na porta 7006.
- Observação: se o horário for agendado por algum cliente, não é possível excluí-lo.





Cadastrar Colaborador

- O servidor permite que um novo colaborador seja criado.
- Vale ressaltar que os servidores que realizam funcionalidades do colaborador foi pensado como servidores internos da rede interna da clínica.
- Esse servidor está conectado na porta 7007.



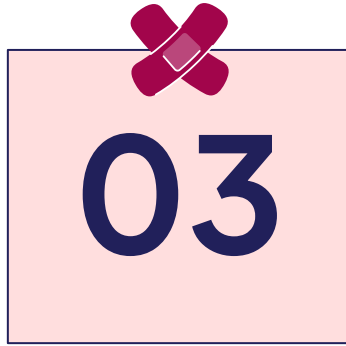


Atualizar Colaborador



- Esse servidor permite que os dados de um colaborador já existente no sistema seja atualizado.
- Os dados que podem ser atualizados são: cargo, telefone, email, endereço, login e senha.
- Já o nome e a matrícula não podem ser atualizados para não tornar os dados da consulta inconsistentes.
- Esse servidor está conectado na porta 7008.





Estrutura



Escopo do Projeto

Linguagem de Programação

- Java

Representação Externa de Dados

- JSON

Arquitetura

- Microsserviço

Requisitos Não-Funcionais

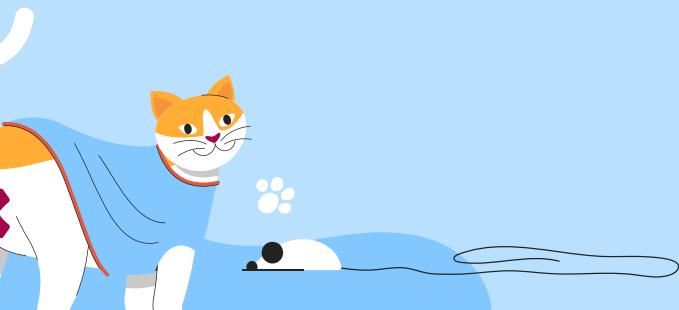
- Segurança
- Escalabilidade

Protocolo de Comunicação

- TCP

Protocolo Requisição-Resposta

- Implementado




Java

- Java é uma linguagem de programação de propósito geral, orientada a objetos e de alto nível.
- Os motivos de escolha da linguagem foi devido a sua portabilidade em vários sistemas operacionais, o fato da sintaxe ser baseada em C e C++, tornando-a mais familiar para a equipe e o seu ecossistema robusto.



JSON



- O JSON (JavaScript Object Notation) é um formato de dados leve e de fácil leitura e escrita. Ele é amplamente utilizado para a troca de dados entre aplicativos e sistemas, especialmente em ambientes web.
 - Já o GSON é uma biblioteca Java desenvolvida pelo Google que permite a conversão de objetos Java em JSON (serialização) e vice-versa (desserialização).
 - A versão do GSON utilizada no projeto foi a 2.10.
 - No projeto, além de ser utilizada para representação externa de dados, ela foi usada para organizar a forma de armazenamento dos dados para simplificar o código.
- 



Microserviço

- Um microserviço é uma abordagem arquitetural para desenvolver e implantar aplicativos como um conjunto de serviços independentes e autônomos.
- Utilizar esse modelo facilita a manutenibilidade, mas também garante tolerância a falhas.
- Geralmente um sistema implementado utilizando microserviço pode se tornar altamente escalável.



Segurança



- Para realizar segurança no sistema do Doguito Petshop usou o keytool que é uma ferramenta de linha de comando fornecida pelo JDK. Ela é usada para gerenciar certificados digitais, chaves criptográficas e keystores (arquivos que armazenam chaves e certificados).
- Além disso, utilizou o TLS (Transport Layer Security) que é um protocolo criptográfico de segurança que fornece comunicação segura em redes de computadores. Ele é projetado para garantir a privacidade e a integridade dos dados transmitidos entre aplicativos em uma rede, protegendo-os contra interceptação, adulteração e falsificação.





Segurança



- Tanto os servidores quanto o cliente são habilitados para trabalhar com a cifra AES.
- A fim de garantir autenticação por parte dos servidores, utilizou-se um certificado autoassinado.
- Desse modo, foi possível fazer conexões com o cliente utilizando um canal de comunicação seguro.



Escalabilidade

- Para garantir escalabilidade decidiu-se dividir a aplicação em vários servidores a fim de evitar que tráfego fique centralizado.
- Além disso, foi uma forma primitiva de fazer uma espécie de balanceamento de carga.
- Uma característica marcante do projeto é a utilização de um pool de threads para conectar vários clientes ao mesmo tempo.
- Em vez de criar e destruir threads repetidamente, um pool de threads mantém um conjunto fixo de threads disponíveis para executar tarefas.
- Esse mecanismo é bastante utilizado em programação concorrente para gerenciar e reutilizar um grupo de threads.



TCP



- TCP (Transmission Control Protocol) é um dos principais protocolos de comunicação da camada de transporte da arquitetura de rede TCP/IP. Ele fornece uma comunicação confiável e orientada à conexão entre dispositivos em uma rede.
- Uma característica marcante do TCP é que ele permite uma garantia de entrega, uma vez que utiliza um mecanismo de reenvio de pacotes.
- Como o sistema do Doguito trabalha com operações críticas como agendamento de consulta e cadastro, faz-se necessária a utilização desse protocolo.





Requisição Resposta



- O protocolo de requisição-resposta é um modelo de comunicação utilizado em sistemas de rede, onde um dispositivo chamado cliente envia uma solicitação (requisição) a outro dispositivo chamado servidor, que por sua vez processa a solicitação e retorna uma resposta ao cliente.
- No projeto foi implementado um protocolo requisição resposta para satisfazer as necessidades do sistema.



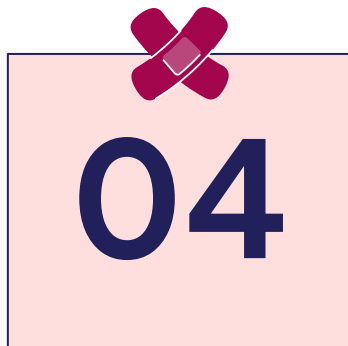


Requisição Resposta



```
4 usages
public final static int OP_CADASTRAR_PET = 1;
3 usages
public final static int OP_ATUALIZAR_PARTE_DONO = 2;
6 usages
public final static int OP_AGENDAR_DONO = 3;
5 usages
public final static int OP_EXCLUIR_AGENDAMENTO_DONO = 4;
4 usages
public final static int OP_INSERIR_HORARIO = 5;
4 usages
public final static int OP_APAGAR_HORARIO = 6;
4 usages
public final static int OP_CADASTRAR_COLABORADOR = 7;
3 usages
public final static int OP_ATUALIZAR_COLABORADOR = 8;
```





Código





Armazenamento



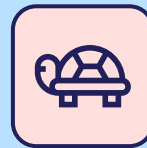
pets.json

Arquivo onde estão guardados os cadastros dos clientes.



agenda.json

Arquivo onde se encontra os horários disponibilizados pelo colaborador e agendados pelo cliente.

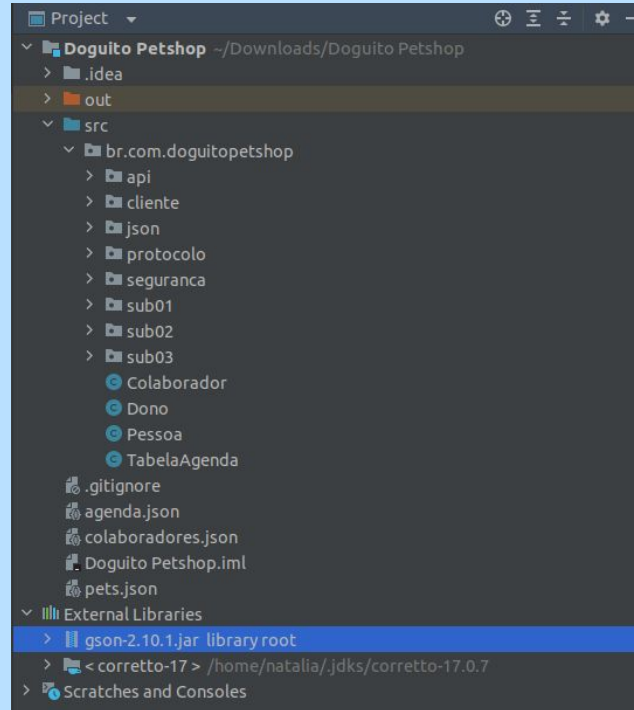


colaboradores.json

Arquivo criado para armazenar o cadastro de todos os colaboradores.



Organização do Projeto



Fim

Alguma dúvida?

