# Agenda

Daniel Vitor, Petrucio Neves, Samuel Henrique

**Universidade Federal do Ceará**

25 de janeiro de 2022

UNIVERSIDADE FEDERAL DO CEARÁ

# Sumário




UNIVERSIDADE FEDERAL DO CEARÁ

# Introdução


UNIVERSIDADE FEDERAL DO CEARÁ

# Introdução

- Objetivo Geral: o desenvolvimento de um protótipo para serviços de uma agenda de contatos
- Principais atores: Usuário da Agenda
- Principais métodos: adicionar, procurar, listar, remover e editar contatos e limpar agenda

UNIVERSIDADE FEDERAL DO CEARÁ

# Introdução - Configuração

- Sistema Distribuído
- Linguagem de Programação: Java
- Arquitetura de Comunicação: Cliente-Servidor
- Socket UDP
- Protocolo no formato requisição-resposta

# Métodos Remotos

# Métodos Remotos

## Adicionar Contatos

**In:** Contato

**Out:** Boolean (true, caso o nome já esteja sendo usado; false, caso contrário)

**Exceções:** Nome não pode ser vazio

## Listar Contatos

**In:** Void

**Out:** Lista de Contatos

**Exceções:** Vazio

UNIVERSIDADE FEDERAL DO CEARÁ

# Métodos Remotos

## Buscar Contatos

**In:** Contato

**Out:** Lista de Contatos

**Exceções:** Vazio

## Editar Contato

**In:** Agenda

**Out:** Boolean (true, caso tenha sido editado; false, caso contrário)

**Exceções:** Nome não pode ser vazio

UNIVERSIDADE FEDERAL DO CEARÁ

# Métodos Remotos

## Remover Contatos

**In:** Contato

**Out:** Boolean (true, caso tenha sido removido; false, caso contrário)

**Exceções:** Vazio

## Limpar Agenda

**In:** Void

**Out:** Void

**Exceções:** Vazio

# Arquitetura

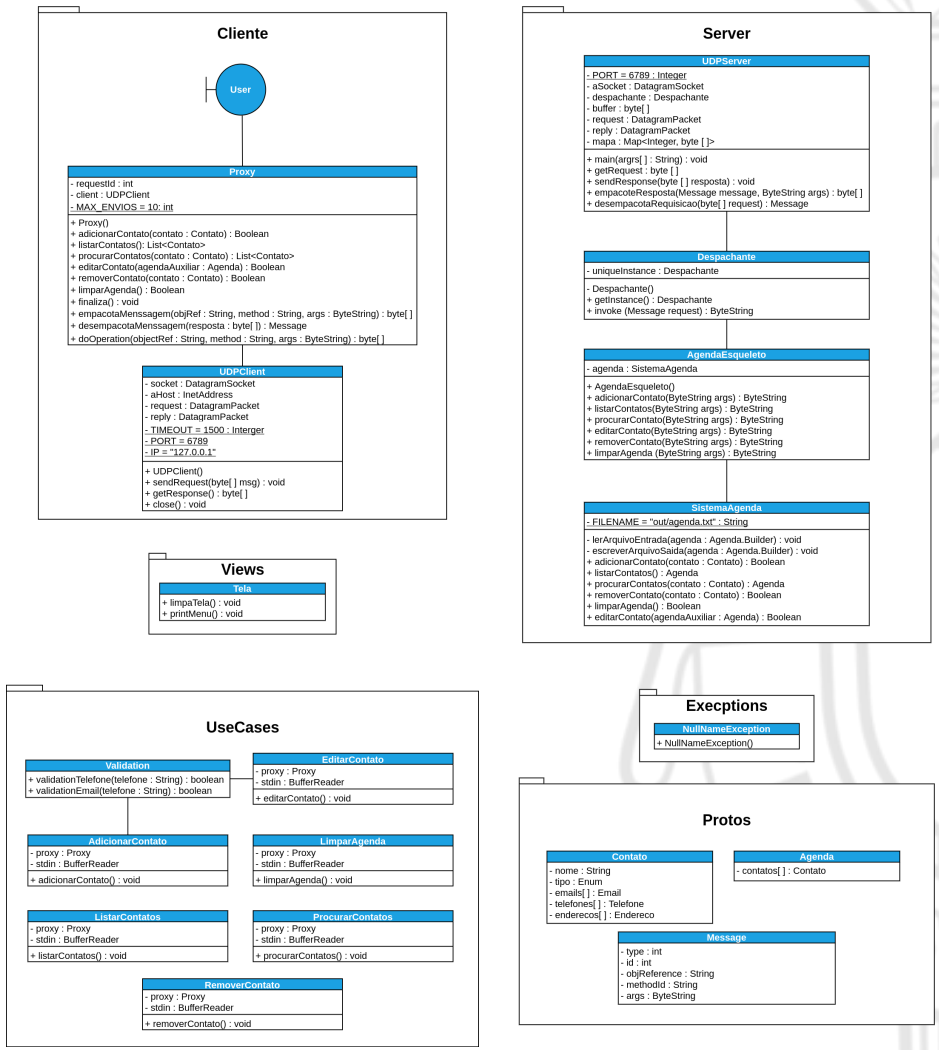


Figura: UML

# Arquitetura

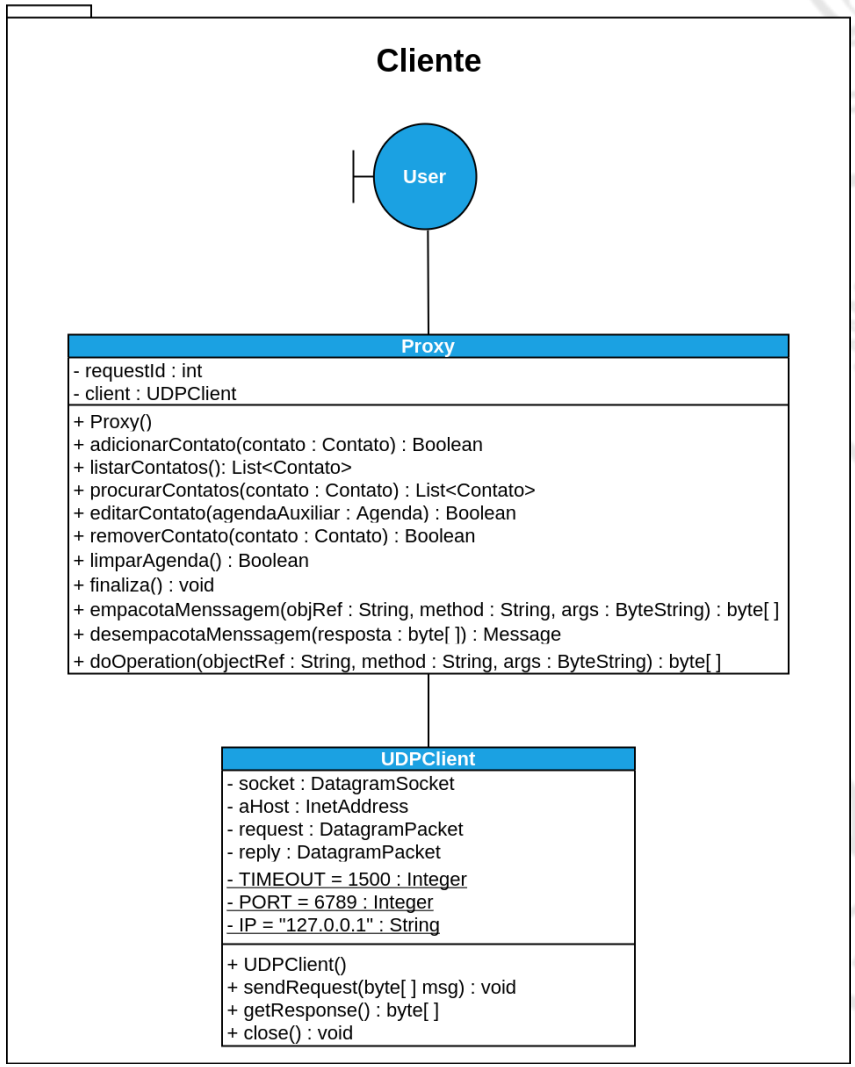


Figura: UML - Lado cliente

UNIVERSIDADE FEDERAL DO CEARÁ

# Arquitetura

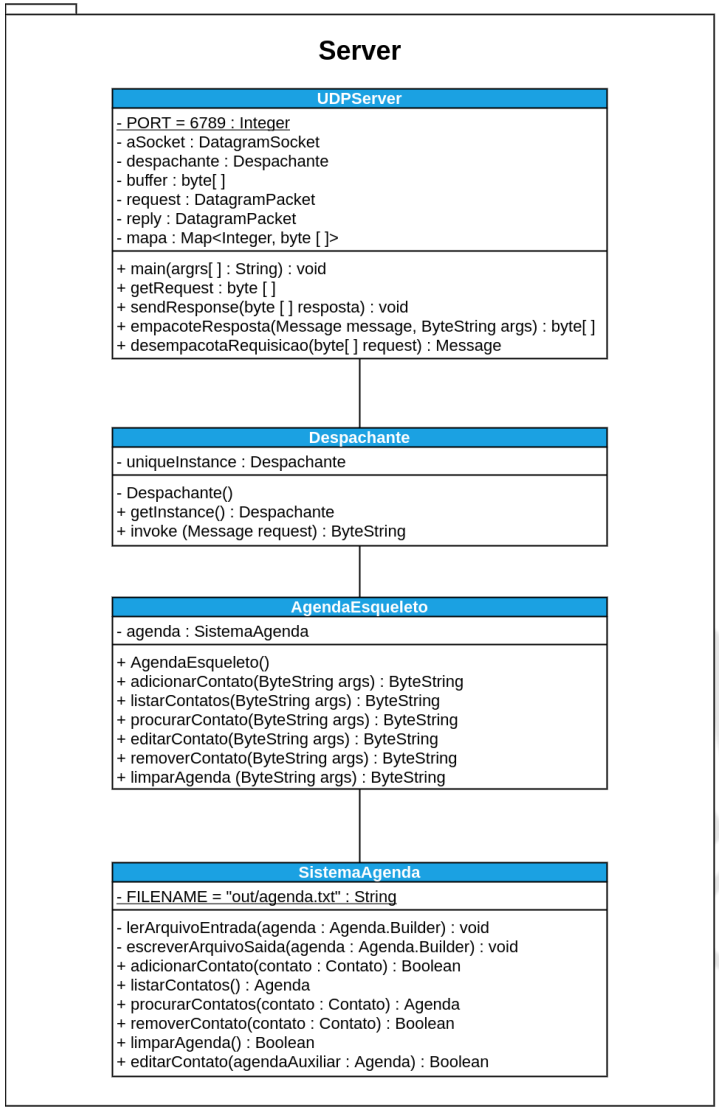


Figura: UML - Lado servidor

# Arquitetura

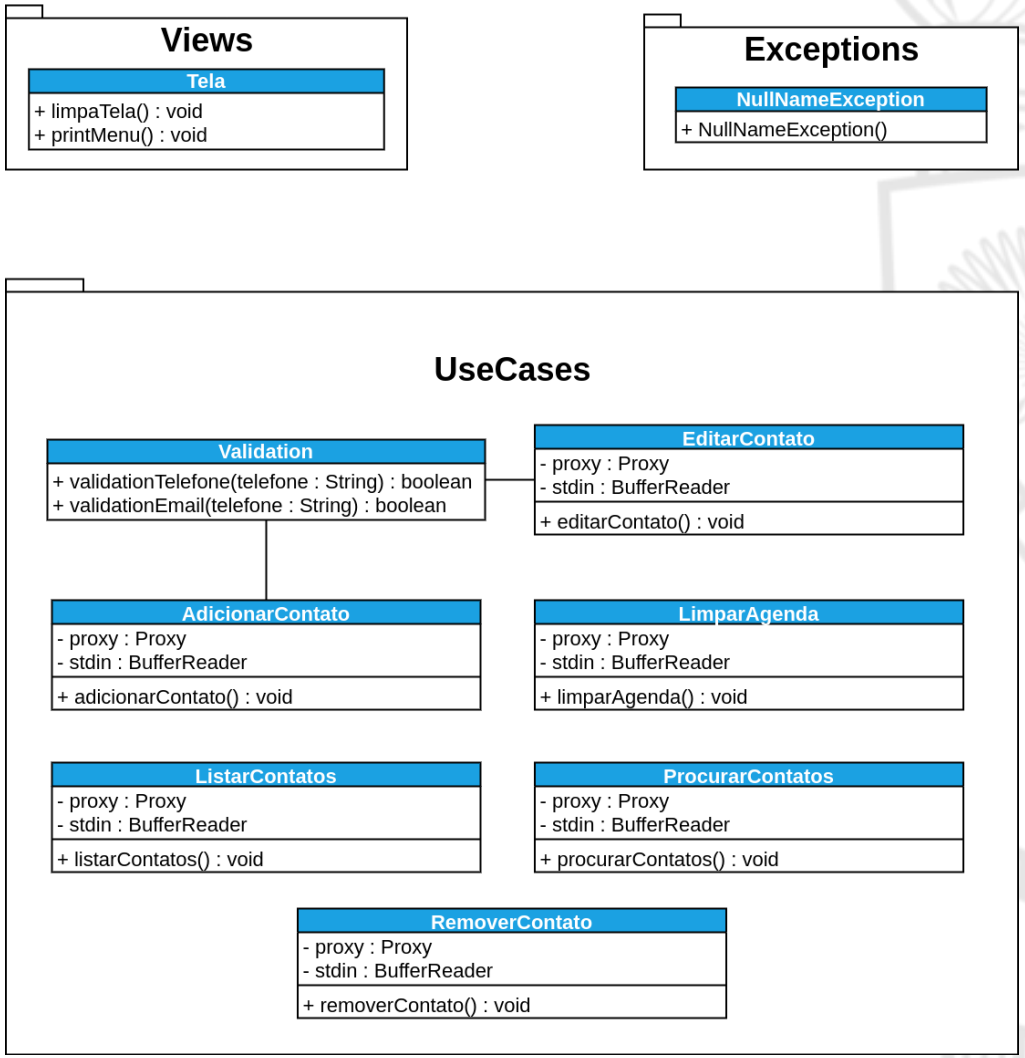


Figura: UML - Outros pacotes

# Arquitetura

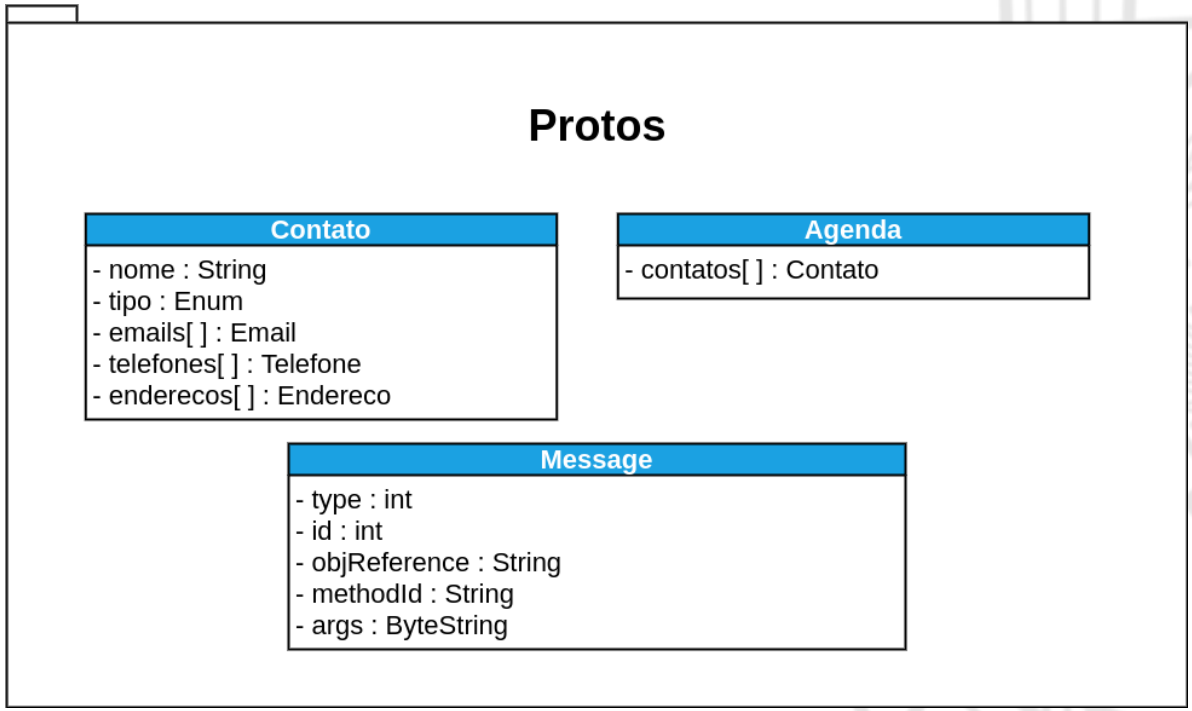


Figura: UML - Protos

# Dados Transmitidos

# Message

```
syntax = "proto3";

package com.trabalhoFinal.protos;

option java_package = "com.trabalhoFinal.protos";
option java_outer_classname = "MessageProto";

message Message {
  int32 type = 1;
  int32 id = 2;
  string objReference = 3;
  string methodId = 4;
  bytes args = 5;
}
```

Figura: message

# Agenda

```
syntax = "proto3";

package com.trabalhoFinal.protos;

option java_package = "com.trabalhoFinal.protos";
option java_outer_classname = "AgendaProto";

message Contato {
  string nome = 1;

  enum Type {
    MOBILE = 0;
    PERSONAL = 1;
    HOME = 2;
    WORK = 3;
  }

  message Email {
    optional string email = 1;
    optional Type type = 2;
  }
```

Figura: agenda

# Agenda

```
  message Telefone {
    optional string telefone = 1;
    optional Type type = 2;
  }

  message Endereco {
    optional string endereco = 1;
    optional Type type = 2;
  }

  repeated Email emails = 2;
  repeated Telefone telefones = 3;
  repeated Endereco enderecos = 4;
}

message Agenda {
  repeated Contato contatos = 1;
}
```

Figura: agenda

# Classes Implementadas

# Classes Implementadas - views -> Tela

- A classe Tela é chamada pela classe User e é responsável pela interface do usuário

# Classes Implementadas - exceptions -> NullNameException

- A classe NullNameException é responsável por retornar um erro, caso o usuário esqueça de colocar o nome do Contato.

# Classes Implementadas - useCases

## AdicionarContato

A classe AdicionarContato é responsável por receber os dados do usuário, e assim enviar as informações nescessárias para adicionar o contato na Agenda.

## ListarContatos

A classe ListarContatos é responsável por receber uma lista dos Contatos por meio do Proxy, e assim imprimir na tela para o Usuário.

## ProcurarContatos

A classe ProcurarContatos é responsável por pedir uma informação de um Contato que o Usuário digita, e assim enviar por meio do Proxy, para o SistemaAgenda, e recebe como resposta, um tipo Agenda com os possíveis contatos e imprimir na tela.

UNIVERSIDADE FEDERAL DO CEARÁ

# Classes Implementadas - useCases

## EditarContato

A classe EditarContato é responsável pegar o Contato que o Usuário quer editar, depois pegar as informações que irão mudar, e enviar para que possa ser alterado

## RemoverContato

A classe RemoverContato é responsável por pegar o contato que o Usuário escolheu para remover, e enviar esse pedido.

## LimparAgenda

A classe LimparAgenda é responsável por mandar essa requisição, pelo Proxy, para remover todos os Contatos da Agenda.

UNIVERSIDADE FEDERAL DO CEARÁ

# Classes Implementadas - useCases -> Validation

- A classe Validation é responsável pela validação dos números de telefone e os emails.

# Classes Implementadas - client -> User

- A classe User é responsável por fazer a interface do usuário com o sistema, nessa classe temos uma instância do Proxy o qual vamos utilizar para fazer todas as chamadas das funções.

# Classes Implementadas - client -> Proxy

- A classe Proxy é responsável por abstrair para o User as chamadas ao servidor principal.
- Além de empacotar e desempacotar as mensagens a serem enviadas/recebidas via UDP.
- Serializar as requisições e desserializar as respostas

UNIVERSIDADE FEDERAL DO CEARÁ

# Classes Implementadas - client -> UDPClient

- A classe UDPClient é responsável por efetuar a comunicação com o servidor trocando os dados que foram empacotados anteriormente.

UNIVERSIDADE FEDERAL DO CEARÁ

# Classes Implementadas - server -> UDPServer

- A classe UDPServer é responsável por desempacotar e desserializar as mensagens, atender as requisições, empacotar e serializar as respostas e depois enviar de volta a resposta daquela requisição, além disso faz o tratamento de falhas.

# Classes Implementadas - server -> Despachante

- A classe Despachante é responsável por decifrar qual o método está sendo chamado através daquela requisição e utilizar a classe Esqueleto para executar tais métodos. Além disso, ele é responsável por retornar a saída que os métodos invocados retornaram.

# Classes Implementadas - server -> AgendaEsqueleto

- A classe AgendaEsqueleto é responsável por fazer o intermédio entre o Despachante e o SistemaAgenda. O AgendaEsqueleto executa o método do sistema e após isso retorna em um ByteString para que seja enviado posteriormente como resposta da requisição solicitada.

# Classes Implementadas - server -> SistemaAgenda

- A classe SistemaAgenda é responsável por implementar todos os métodos do sistema da Agenda para serem executadas em um servidor.
- Toda a parte lógica está na classe.

UNIVERSIDADE FEDERAL DO CEARÁ

# Tratamentos de Falhas

# Tratamentos de Falhas

- Perda da Mensagem
  - Timeout: 1,5s
  - Reenvio
- Mensagem Duplicada
  - TreeMap<Integer, byte[]>

# Obrigado(a) pela Atenção!


UNIVERSIDADE FEDERAL DO CEARÁ