Agenda

QXD0043 - SISTEMAS DISTRIBUÍDOS

Equipe:
471047 — PEDRO HENRIQUE MAGALHÃES BOTELHO
475664 — DAVID MACHADO COUTO BEZERRA
493470 — PAULO ARAGÃO DE AZEVEDO NETO

UFC CAMPUS QUIXADÁ

UNIVERSIDADE FEDERAL DO CEARÁ

Sumário

Introdução

Introdução

- Objetivo Principal: Desenvolvimento de uma Agenda de Contatos;
- Arquitetura: Cliente-Servidor;
- Principais Funções:
  - Adicionar Contatos;
  - Listar Contatos;
  - Procurar Contatos;
  - Apagar Contatos;
  - Editar Contatos;
  - Limpar Agenda;

Introdução

- Linguagem Utilizada: JAVA;
- Comunicação: via Socket UDP;
- Protocolo no formato requisição-resposta;

Métodos

Métodos

1. **Adicionar Contatos**:
    - **In**: Contato;
    - **Out**: Boolean;
    - **Exceções**: Nome ou telefone vazios;

2. **Listar Contatos**:
    - **In**: Void;
    - **Out**: Lista de Contatos;
    - **Exceções**: Nenhuma;

Métodos

3. **Buscar Contatos**:
    - **In**: Nome;
    - **Out**: Contato;
    - **Exceções**: Nome ou telefone vazios;

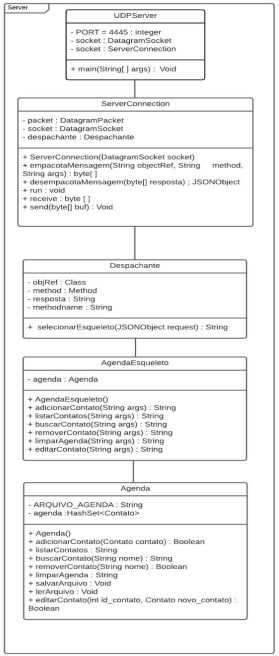
4. **Editar Contato**:
    - **In**: Id;
    - **Out**: Boolean;
    - **Exceções**: Nome ou telefone vazios;

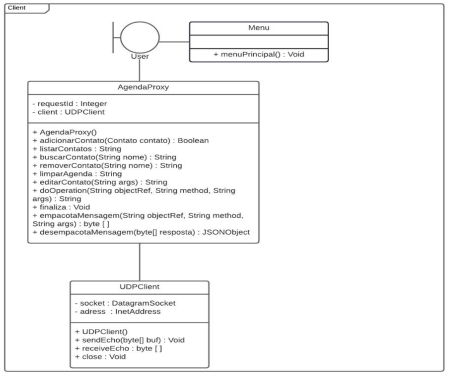
Métodos

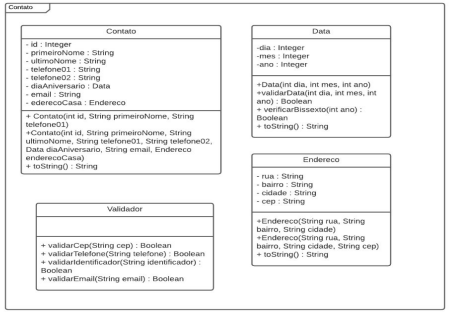
5. **Remover Contato**:
    - **In**: Nome;
    - **Out**: Boolean;
    - **Exceções**: Nenhuma;

6. **Limpar Agenda**:
    - **In**: Void;
    - **Out**: Void;
    - **Exceções**: Nenhuma;

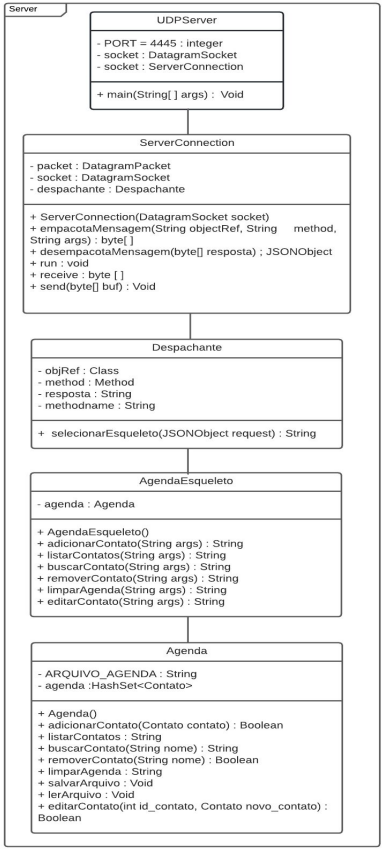
Arquitetura



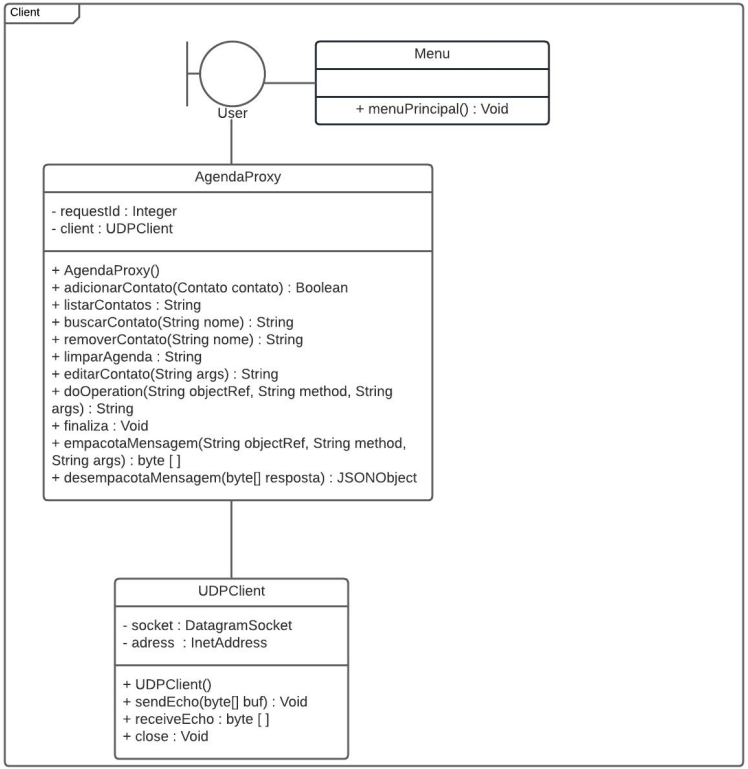




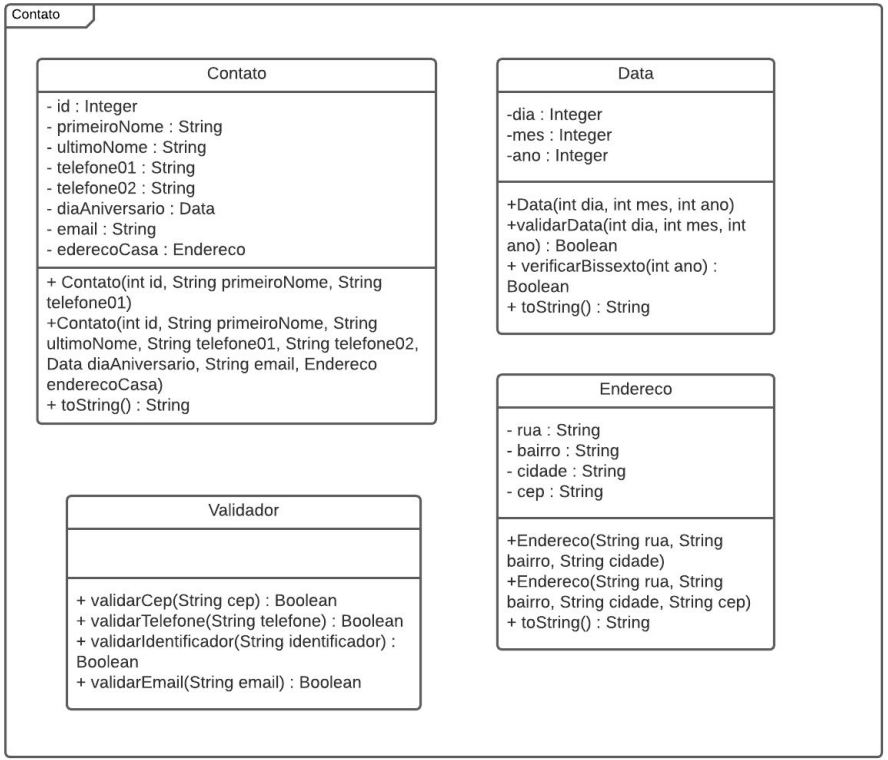
Arquitetura



Arquitetura



Arquitetura



Transmissão de Dados

Empacotar Mensagem

```java
public byte[] empacotaMensagem(String objectRef, String method, String args) {
    JSONObject mensagem = new JSONObject();
    mensagem.put(key: "type", value: 1);
    mensagem.put(key: "id", value: 0);
    mensagem.put(key: "objReference", objectRef);
    mensagem.put(key: "methodId", method);
    mensagem.put(key: "arguments", args);

    return mensagem.toString().getBytes();
}
```

Contato

```java
public Contato(int id, String primeiroNome, String telefone01) throws IllegalArgumentException {
    if(id < 0) {
        throw new IllegalArgumentException("Erro: Número de ID inválido!");
    }
    this.id = id;
    setPrimeiroNome(primeiroNome);
    setTelefone01(telefone01);
}

public String getPrimeiroNome() {
    return this.primeiroNome;
}
public String getTelefone01() {
    return this.telefone01;
}

public void setPrimeiroNome(String primeiroNome) throws IllegalArgumentException {
    if(!Validador.validarIdentificador(primeiroNome)) {
        throw new IllegalArgumentException("Erro: Primeiro Nome inválido!");
    }
    this.primeiroNome = primeiroNome;
}

public void setTelefone01(String telefone01) throws IllegalArgumentException {
    if(!Validador.validarTelefone(telefone01)) {
        throw new IllegalArgumentException("Erro: Telefone 01 inválido!");
    }
    this.telefone01 = telefone01;
}

public String toString() {
    String endereco_str = String.format("{\"id\":%d,\"primeiroNome\":\"%s\",\"telefone01\":\"%s\"}",
                                        this.id, this.primeiroNome, this.telefone01);

    return endereco_str;
}
```

Métodos Implementados

Adicionar Contato

```java
public boolean adicionarContato(Contato contato) throws Exception {
    if(this.agenda.containsKey(contato.getId())) {
        return false;
    }
    if(contato.getPrimeiroNome().isEmpty() || contato.getTelefone01().isEmpty()) {
        throw new Exception("Campos 'Primeiro Nome' e 'Telefone 01' devem estar ambos preenchidos!");
    }

    lerArquivo();
    this.agenda.put(contato.getId(), contato);
    salvarArquivo();
    return true;
}
```

Listar Contato

```
public String listarContatos() {
     lerArquivo();

     if(agenda.isEmpty()) {
          return "Agenda Vazia!";
     }

     StringBuilder contatos = new StringBuilder();
     for(Contato cont_agenda : agenda.values()) {
          contatos.append("ID: " + cont_agenda.getId() + ", Nome: " + cont_agenda.getPrimeiroNome() + "\n");
     }

     return contatos
     .replace(contatos.length() - 1, contatos.length(), "")
     .toString();
}
```

Buscar Contato

```java
public Contato[] buscarContato(String nome){
    lerArquivo();

    ArrayList<Contato> contatos = new ArrayList<>();
    for (Contato cont_agenda : agenda.values()){
        if (cont_agenda.getPrimeiroNome().equals(nome)){
            contatos.add(cont_agenda);
        }
    }

    return (Contato[]) contatos.toArray();
}
```

Remover Contato

```java
public boolean removerContato(int id_contato){
    lerArquivo();
    boolean cont = (agenda.remove(id_contato) != null);
    System.out.println(cont);
    salvarArquivo();
    return cont;
}
```

Limpar Agenda

```
public void limparAgenda(){
        lerArquivo();
        agenda.clear();
        salvarArquivo();
}
```

Editar Contato

```java
public boolean editarContato(int id_contato, Contato novo_contato) throws Exception {
    if(this.agenda.containsKey(id_contato)) {
        throw new Exception("O contato requerido não existe!");
    }
    if(novo_contato.getPrimeiroNome().isEmpty() || novo_contato.getTelefone01().isEmpty()) {
        throw new Exception("Campos 'Primeiro Nome' e 'Telefone 01' devem estar ambos preenchidos!");
    }
    lerArquivo();
    boolean resultado = (this.agenda.put(id_contato, novo_contato) != null);
    salvarArquivo();
    return resultado;
}
```

Tratamento de Falhas

Tratamento de Falhas

- Perda de Mensagem:
  - Timeout 1s;
  - Retransmissão;

```java
//construtor
public UDPClient() throws SocketException, UnknownHostException {
    socket = new DatagramSocket();
    address = InetAddress.getByName("localhost");
    socket.setSoTimeout(1000);
}
```

Tratamento de Falhas

- Mensagem Duplicada:
  - HashMap;

```java
public class ServerConnection extends Thread {
    private DatagramPacket packet;
    private DatagramSocket socket;
    private Despachante despachante;
    private HashMap<Integer, byte[]> historicoMensagens;
```

```java
public void run() {
    while(true) {
        byte[] m = receive();

        // ============== Mensagem Recebida ====================
        System.out.println("RECEBIDA:" + (new String(m)));

        JSONObject mensagem = desempacotaMensagem(m);

        int id = (int) mensagem.get(key: "id");

        if(this.historicoMensagens.containsKey(id)) {
            send(this.historicoMensagens.get(id));
            continue;
        }

        String resultado = despachante.selecionaEsqueleto(mensagem);

        byte[] buf = empacotaMensagem(mensagem, resultado);

        this.historicoMensagens.put(id, buf);

        // ============== Mensagem Enviada ==================
        System.out.println("ENVIADA: " + (new String(buf)));

        send(buf);
    }
}
```

Perguntas?