FAI – CENTRO DE ENSINO SUPERIOR EM GESTÃO, TECNOLOGIA E EDUCAÇÃO

CURSO DE SISTEMAS DE INFORMAÇÃO

DILTON THALES MELO DA SILVA

LUCAS DOS REIS SEVERINI

MATEUS BOCHE DANIEL

DONATE

SANTA RITA DO SAPUCAÍ - MG

2025

SUMÁRIO

[1 PROJETO DE SISTEMA DISTRIBUÍDO 3](#_Toc204949122)

[1.1 PROCEDIMENTOS PARA TRATAMENTO DOS DESAFIOS 3](#_Toc204949123)

[1.1.1 Heterogeneidade 3](#_Toc204949124)

[1.1.2 Escalabilidade 3](#_Toc204949125)

[1.1.3 Abertura 3](#_Toc204949126)

[1.1.4 Segurança 3](#_Toc204949127)

[1.1.5 Manuseio de Falhas 3](#_Toc204949128)

[1.1.6 Concorrência 3](#_Toc204949129)

[1.1.7 Transparência 3](#_Toc204949130)

[1.2 TECNOLOGIAS E ARQUITETURAS DE DISTRIBUÇÃO 4](#_Toc204949131)

# 1 PROJETO DE SISTEMA DISTRIBUÍDO

Um sistema distribuído é aquele em que os componentes localizados em um computador estão conectados uns aos outros em uma rede, comunicam-se e coordenam suas ações apenas passando mensagens. Essa definição leva as seguintes características de sistemas distribuídos: dispositivos sendo acessados por vários usuários ao mesmo tempo, falta de relógio global e falhas de componentes individuais (COULOURIS; DOLLIMORE; KINDBERG, 2007).

## 1.1 PROCEDIMENTOS PARA TRATAMENTO DOS DESAFIOS

Esta seção descreve os principais requisitos necessários para um sistema distribuído apresentar interoperabilidade funcional. Esses requisitos são: heterogeneidade, escalabilidade, abertura, segurança, manuseio de falhas, concorrência e transparência.

### 1.1.1 Heterogeneidade

Com a Internet, é possível aos usuários acessarem serviços e executarem aplicativos por meio de um conjunto heterogêneo de computadores e redes (COULOURIS; DOLLIMORE; KINDBERG, 2007).

A heterogeneidade se aplica aos seguintes aspectos:

1. redes: via protocolos de Internet é possível ser realizada a comunicação;
2. *hardwares* de computador: diferentes padrões de placas de rede fazem que aconteça diferentes implementações. Exemplo: placa de rede Ethernet têm uma implementação diferente daqueles que possuem placas de outros padrões, isto acontece também com arquiteturas de microprocessadores (Intel, ARM, por exemplo);
3. sistemas operacionais: diferentes sistemas operacionais trazem consigo diferentes formas de programação;
4. linguagens de programação: cada linguagem de programação possui particularidades, seja no tratamento de vetores, registros e variáveis. É necessário tratar essas diferenças para que não haja erros de dados e aconteça uma comunicação efetiva;

implementação de diferentes desenvolvedores: é necessário que diferentes desenvolvedores utilizem de padrões e convenções comuns durante a programação para que haja comunicação entre os diferentes sistemas.

### 1.1.2 Escalabilidade

Um sistema é classificado como escalável se, perante um significativo número de recursos e usuários, ele permanece eficiente (COULOURIS; DOLLIMORE; KINDBERG, 2007).

### 1.1.3 Abertura

De acordo com Tanenbaum (2007, p.4), um sistema distribuído aberto é um sistema que oferece serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços. No caso de sistemas distribuídos, em geral os serviços são especificados por meio de interfaces, que costumam ser descritas em uma linguagem de definição de interface (*Interface Definition Language* – IDL).

A principal característica de um sistema aberto é a documentação das principais interfaces do software com as principais funções, como utilizá-las em outras aplicações e principalmente a divulgação da API para que outros desenvolvedores possam utilizar dos recursos oferecidos pelo sistema. Um exemplo são os aplicativos que utilizam as APIs do Google Maps, conseguindo utilizar dados fornecidos pela API, sem precisar ter contato com os desenvolvedores da Google.

### 1.1.4 Segurança

Muitos recursos de informação que se tornam disponíveis e são mantidos em sistemas distribuídos têm um alto valor intrínseco para seus usuários. Portanto, sua segurança é de fundamental importância. Na segurança de recursos de informação existem quatro componentes: confidencialidade (proteção contra exposição para pessoas não autorizadas), integridade (proteção contra alteração ou dano), disponibilidade (proteção contra interferência com os meios de acesso aos recursos) e autenticidade (provar a identidade do usuário a traves de senhas, certificados digitais, assinaturas digitais) (COULOURIS; DOLLIMORE; KINDBERG, 2007).

### 1.1.5 Manuseio de Falhas

Coulouris, Dollimore e Kindberg (2007) afirmam que falhas em sistemas distribuídos são parciais, pois todos eles possuem componentes que funcionam independentemente, portanto, mesmo com a falha de um componente, outros podem continuar operando normalmente, dificultando assim o manuseio de falhas. É essencial que todo componente seja desenvolvido com o objetivo de que ele, mesmo ocorrendo falhas dos componentes que depende, funcione ou trate destas falhas apropriadamente.

### 1.1.6 Concorrência

Em um ambiente concorrente, cada recurso deve ser projetado para oferecer consistência nos estados de seus dados. É essencial que todos os recursos estejam disponíveis, com o maior desempenho possível e para o maior número de usuários possíveis simultaneamente (COULOURIS; DOLLIMORE; KINDBERG, 2007).

### 1.1.7 Transparência

O desafio da transparência tem o objetivo de tornar aspectos da distribuição invisíveis para o desenvolvedor e usuários, a fim de que ele se preocupe apenas com o projeto de seu sistema em particular. O objetivo de tornar ocultos certos aspectos da distribuição é para que este seja percebido como um sistema único em vez de uma coleção de componentes independentes (COULOURIS; DOLLIMORE; KINDBERG, 2007).

## 1.2 TECNOLOGIAS E ARQUITETURAS DE DISTRIBUÇÃO

O sistema Donate possui uma arquitetura baseada em Cliente-Servidor. Para cada requisição de um cliente é o servidor que irá fazer todo o processamento e retornar as respostas, por meio de protocolos de rede e com passagens de métodos Rest.

As funcionalidades do sistema Donate estão diretamente relacionadas aos componentes de um sistema distribuído.

Por exemplo, o cliente (P5) envia uma solicitação REST para o front-end (P4), que por sua vez se comunica com o back-end (P3) desenvolvido em Spring Boot.

Este back-end (P3) realiza o processamento, acessa o gerenciador de banco de dados relacional PostgreSQL (P1) para armazenar ou recuperar informações e pode acionar o servidor de e-mail (P2) para notificações automáticas.

Além disso, o (P3) pode consultar o servidor de mapas (P7) para exibir localizações, integrando todos esses processos de maneira transparente para o usuário.

Essa comunicação entre múltiplos processos e dispositivos exemplifica o funcionamento de um sistema distribuído dentro do contexto do projeto Donate.

O (P1) responsável por validar as requisições do cliente, processar, persistir os dados e retornar o resultado. O cliente, (P5), (um navegador) faz a interface com o usuário, sendo por intermédio dela que ele terá acesso a todo sistema.

O diagrama de distribuição do sistema Donate está apresentado abaixo ela explica como funciona o sistema da perspectiva de um sistema distribuído. Nela está presente os processos clientes, que são constituídos pelos processos:

1. P1 - processo SGBDR PostgreSQL: gerenciador de banco de dados relacional
2. P2 - processo Servidor de e-mail: recebimento e envio de e-mails
3. P3 - processo *back-end* (Spring Boot): conecta APIs com o banco de dados, gerencia conexões de usuários e alimenta os aplicativos Web e mobile
4. P4 - processo *front-end* (Angular): interface gráfica do projeto
5. P5 - processo do navegador cliente: processamento do aplicativo Web
6. P6 - processo Donate mobile: processamento do aplicativo mobile
7. P7 - processo Servidor de mapas: conecta API de mapa

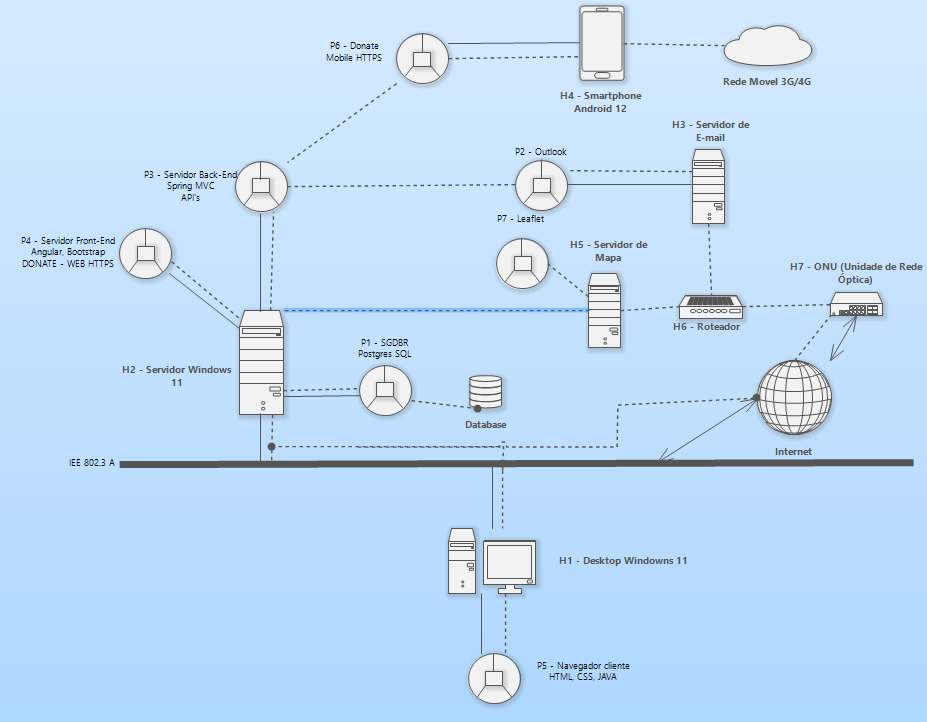


Figura - Diagrama do Sistema Distribuído do Projeto DONATE

FONTE: Autoria Própria