## 6.5 Projeto do Sistema Distribuído

Sistema distribuído é aquele em que os componentes localizados em um computador estão conectados uns aos outros em uma rede, se comunicam 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 (DOLLIMORE e KINDBERG 2007).

### **6.5.1 Procedimentos para Tratamentos 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.

6.5.1.1 Heterogeneidade

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

A heterogeneidade se aplica aos seguintes aspectos:

1. redes: através de 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, etc.);
3. sistemas operacionais: diferentes sistemas operacionais trazem consigo diferentes formas de programação;
4. linguagens de programação: cada linguagem de programação possui suas singularidades, seja no tratamento de vetores, registros e variáveis. É necessário tratar essas diferenças para que não haja erros de informações e haja uma comunicação efetiva;
5. implementação de diferentes desenvolvedores: é necessário que diferentes desenvolvedores utilizem de padrões comuns no desenvolvimento para que haja comunicação entre os diferentes sistemas.

O sistema Deu Pet é uma solução distribuída. A aplicação *back-end* é implementada utilizando a linguagem de programação Java. O *back-end* e também o *front-end* da aplicação, serão executadas por meio do thymeleaf.

Também é utilizada neste sistema a arquitetura *Representational State Transfer* (REST), para acesso aos serviços providos pela aplicação servidora, linguagem Java e estilo de programação SOA/REST. O sistema utiliza o protocolo *Hypertext Transfer Protocol Secure* (HTTPS) para comunicação de dados entre as aplicações com segurança.

6.5.1.2 Escalabilidade

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

Para possuir escalabilidade, o sistema Deu Pet, utiliza boas práticas de programação, com algoritmos otimizados. Em relação ao banco de dados, existem rotinas de manutenção e criação de índices. Outra formar possível para aprimorar a escalabilidade é a utilização de serviços *Cloud*, entretanto, essa aplicação ainda não definiu se irá utiliza-lo.

6.5.1.3 Abertura

De acordo com Tanenbaum e Steen (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 simples, são os aplicativos que utilizam as APIs do Google Maps, conseguindo utilizar dados e informações presentes na API, sem precisar ter contato com os desenvolvedores do Google.

A aplicação Deu Pet, mesmo se utilizando de uma documentação padronizada, não está em seu escopo a abertura de sua API para terceiros.

6.5.1.3 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 considerável importância. As seguranças de recursos de informação têm três componentes: confidencialidade (proteção contra exposição para pessoas não autorizadas), integridade (proteção contra alteração ou dano) e disponibilidade (proteção contra interferência com os meios de acesso aos recursos) (DOLLIMORE e KINDBERG 2007).

Além dos três citados, temos a autenticidade, que completa a confidencialidade, visa estabelecer a validade da transmissão, da mensagem e do seu remetente. O objetivo é que o destinatário possa comprovar a origem e autoria de um determinado documento.

Neste projeto, utiliza-se o JSON Web Token (JWT), que de acordo com Adriano (2017), é um sistema de transferência de dados que pode ser enviado via requisição POST ou em um cabeçalho HTTP (*header*) de maneira “segura”, essa informação é assinada digitalmente por um algoritmo HMAC, ou um par de chaves pública/privada, usando método de criptografia *Rivest – Shamir – Adleman* (RSA). Pode-se ver na Figura 1 um cenário onde será requisitado um *token* por meio do verbo HTTP POST, que irá devolver um *token* validado para que as próximas requisições que utilizem os verbos HTTP possam utilizá-lo.

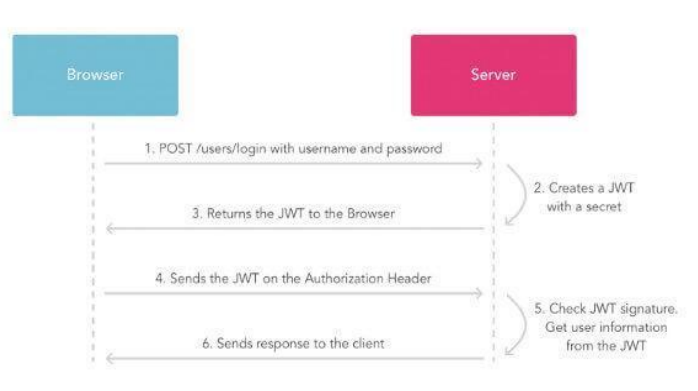


FIGURA 1: Diagrama de sequência usando JWT

FONTE: Adriano (2017)

O sistema Deu Pet também se utiliza do Spring Security, que é uma estrutura de autenticação e controle de acesso poderosa e altamente personalizável. É o padrão de fato para proteger aplicativos baseados em Spring. O Spring Security é uma estrutura que se concentra no fornecimento de autenticação e autorização para aplicativos escritos em Java. Como todos os projetos do Spring, o poder real do Spring Security é encontrado na facilidade com que pode ser estendido para atender aos requisitos personalizados.

6.6.1.5 Manuseio de falhas

Dollimore e Kindeberg (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, os 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.

Com base nisso, a aplicação Deu Pet ainda não tratou de forma total o desafio do manuseio de falhas. A princípio, será tratado a nível de aplicação. Em falha na API, o sistema irá retornar uma mensagem simples de falha ao usuário final para conhecimento.

6.5.1.6 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 (DOLLIMORE e KINDBERG 2007).

Os aplicativos do Deu Pet utilizam a mesma API, desenvolvida de modo concorrente, para que todas requisições serem atendidas. Os processos são tratados pelo sistema operacional, seguem a ordem de prioridade e tempo de uso dos recursos conforme as características do hardware e SO.

6.5.1.7 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. (DOLLIMORE e KINDBERG 2007).

Um sistema com o objetivo de atender às necessidades dos clientes pode realizar diversas operações relacionadas ao banco de dados, utilizar a API sem a necessidade de entendimento da distribuição, suas regras técnicas e de implementação, apenas fornecendo os dados necessários.

O Deu Pet utiliza métodos que podem ser executados concorrentemente e recursos compartilhados sem a interferência entre si, realizados via funções do *thymeleaf* e pelo SGBD *PostgreSQL*, que fornece interfaces genéricas para esses aspectos bem como implementações específicas de armazenamento de persistência.

### **6.5.2 Tecnologias e Arquiteturas de Distribuição**

A aplicação Deu Pet 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.

O servidor é responsável por validar as requisições do cliente, processar, persistir os dados e retornar o resultado. O cliente (um navegador) é o canal de comunicação com o usuário, sendo por intermédio dele que ele terá acesso ao sistema, além de prover a interface gráfica de usuário necessária para a interação.

O diagrama de distribuição do sistema Deu Pet está representado na Figura 2. Ela explica como funciona o sistema Deu Pet da perspectiva de um Sistema Distribuído. Nela está presente os processos clientes, que são constituídos pelos processos:

1. P1 - processo SGBD PostgreSQL: é o processo responsável por fazer o gerenciamento de acessos ao banco de dados. O banco de dados é chamado de Deu Pet;
2. P2 - processo servidor de e-mail: é o processo responsável por receber e enviar e-mails;
3. P3 - processo Heroku: é uma plataforma de nuvem como serviço que suporta várias linguagens de programação, é onde a componente servidora será executada. O sistema Operacional sendo o Dynos.
4. P4 - processo do *browser* em um dispositivo móvel: fará o processamento da  
   aplicação *front-end*;
5. P5 – processo do *browser* Google Chrome: fará o processamento da aplicação *frontend*;

(LEMBRETE: REALIZAR CORREÇÃO DO DIAGRAMA COM BASE NOS APONTAMENTOS ENVIADOS POR EMAIL PELO PROFESSOR GAVIÃO NO DIA 12/04. ASSIM QUE ARRUMAR RETORNAR À CORREÇÃO PARA O PROFESSOR CONFERIR SE FICOU BOM)

Uma imagem contendo Texto

Descrição gerada automaticamente

FIGURA 2- Diagrama de distribuição

FONTE: Autoria Própria