DOCUMENTO DE ARQUITETURA DE SOFTWARE

SISTEMA DE GESTÃO DE TICKETS DE ATENDIMENTO PARA A EMPRESA PAPÉIS MIGUEL ESCOBAR

Objetivo do Sistema: O sistema de gestão de tickets de atendimento da Papéis Miguel Escobar tem como objetivo principal gerenciar solicitações de demandas de usuários. Ele permite que os usuários criem tickets para solicitar serviços, entrega de material, suporte técnico entre outras funcionalidades que podem ser adicionadas, e que os atendentes gerenciem e solucionem esses tickets de forma eficiente, em um ambiente coeso.

Principais Funcionalidades:

1. **Criação de Tickets:**
    - Usuários podem criar tickets definindo o setor de destino, a categorização dentre as opções possíveis e descrevendo a solicitação.
    - Cada ticket recebe um identificador único e é registrado no sistema.
2. **Visualização de Tickets:**
    - Usuários podem visualizar o status e detalhes dos seus próprios tickets.
    - Atendentes podem visualizar todos os tickets de seu setor para gerenciar e realizar o atendimento.
3. **Atualização e Resolução de Tickets:**
    - Atendentes podem atualizar o status dos tickets (do status Aberta para Em Atendimento quando atribui a si mesmo um ticket que não está sob posse de outro atendente, e de Em Atendimento para Finalizada quando soluciona a demanda).
    - Atendentes devem adicionar comentários nas soluções dos tickets.
4. **Notificações:**
    - Usuários recebem notificações sobre atualizações no status dos seus tickets.
    - Atendentes recebem notificações sobre novos tickets e atualizações importantes.
5. **Reabertura de Tickets:**
    - Usuários podem reabrir tickets finalizados, caso entendam que a solução oferecida não atende à solicitação.
    - Um novo ticket é criado com novo identificador único e uma referência ao ticket reaberto.
6. **Autenticação e Autorização:**
    - O sistema garante que apenas usuários autenticados possam acessar suas funcionalidades.
    - Diferentes níveis de acesso são garantidos para usuários comuns e atendentes.
7. **Transferência de Tickets:**
    - Atendentes podem transferir os tickets em sua posse para outros atendentes.
    - Atendentes podem atribuir a si mesmos tickets em posse de outros atendentes.
    - Cada transferência de ticket realizada é registrada no sistema com um identificador único e com os dados do atendente anterior, novo atendente e a direção (se foi autoatribuído ou se foi transferido para outro atendente).
8. **Emissão de Relatórios:**
    - Atendentes podem emitir relatórios sobre os tickets.
    - Atendentes podem escolher o tipo de relatório a ser emitido.
    - Atendentes podem filtrar opções para a emissão de relatórios.
    - Cada relatório recebe um identificador único e é registrado no sistema.

Tipos de Usuários:

1. **Usuário Comum:**
    - Pode criar tickets.
    - Pode visualizar e acompanhar o status dos seus próprios tickets.
    - Recebe notificações sobre atualizações nos seus tickets.
2. **Atendente:**
    - Pode visualizar todos os tickets.
    - Pode atualizar o status dos tickets.
    - Recebe notificações sobre novos tickets e atualizações importantes.
    - Pode emitir relatórios.

Fluxo de Trabalho (Tickets):

1. **Usuário Comum cria um ticket:**
    - O usuário acessa a interface do sistema e preenche um formulário com os detalhes da demanda.
    - O ticket é registrado no sistema e o usuário recebe uma confirmação.
2. **Atendente gerencia o ticket:**
    - O atendente visualiza o novo ticket na sua interface de gerenciamento.
    - O atendente atualiza o status do ticket conforme trabalha na resolução do problema.
    - O atendente pode adicionar comentários e soluções ao ticket.
3. **Usuário Comum acompanha o ticket:**
    - O usuário pode acessar a interface do sistema para visualizar o status do ticket e a solução provida em tickets finalizados.
    - O usuário recebe notificações sobre qualquer atualização no ticket.
4. **Resolução do ticket:**
    - Uma vez que o problema é resolvido, o atendente marca o ticket como resolvido.
    - O usuário recebe uma notificação informando que o ticket foi resolvido.
5. **Avaliação do atendimento (opcional):**
    - O usuário acessa a interface do sistema e visualiza os seus tickets que foram solucionados.
    - O usuário seleciona o ticket que pretende avaliar.
    - O usuário avalia com uma nota de 1 a 5 o atendimento prestado e a solução dada.
    - Opcionalmente, o usuário adiciona algum comentário à avaliação.

Fluxo de Trabalho (Emissão de relatórios):

1. **Atendente emite relatório:**
    - O atendente acessa a seção de relatórios do sistema.
    - O atendente escolhe as datas inicial e final que o relatório deve abranger.
    - O atendente seleciona o tipo de relatório desejado
    - O atendente seleciona entre os diversos filtros disponíveis para o relatório
    - O atendente emite o relatório.

Tecnologias e Ferramentas Utilizadas:

- **Frontend:** Nextjs/React para a interface do usuário.

- **Backend:** Laravel para a lógica de negócios e API. Node.js com Express como camada de comunicação com o Banco de Dados MongoDB, aceitando somente requisições do Backend Laravel.
- **Database:** MongoDB somente para armazenar os relatórios gerados. MariaDB para o cerne do sistema.
- **Autenticação:** JWT (JSON Web Tokens) para autenticação e autorização.
- **Notificações:** Serviços de notificação no próprio sistema, com possíveis melhorias envolvendo push notifications e e-mails.

Para melhor visualizar as integrações definidas, foram desenvolvidos diagramas utilizando as especificações do modelo C4, conforme seguem abaixo:

Nível 1: Diagrama de Contexto

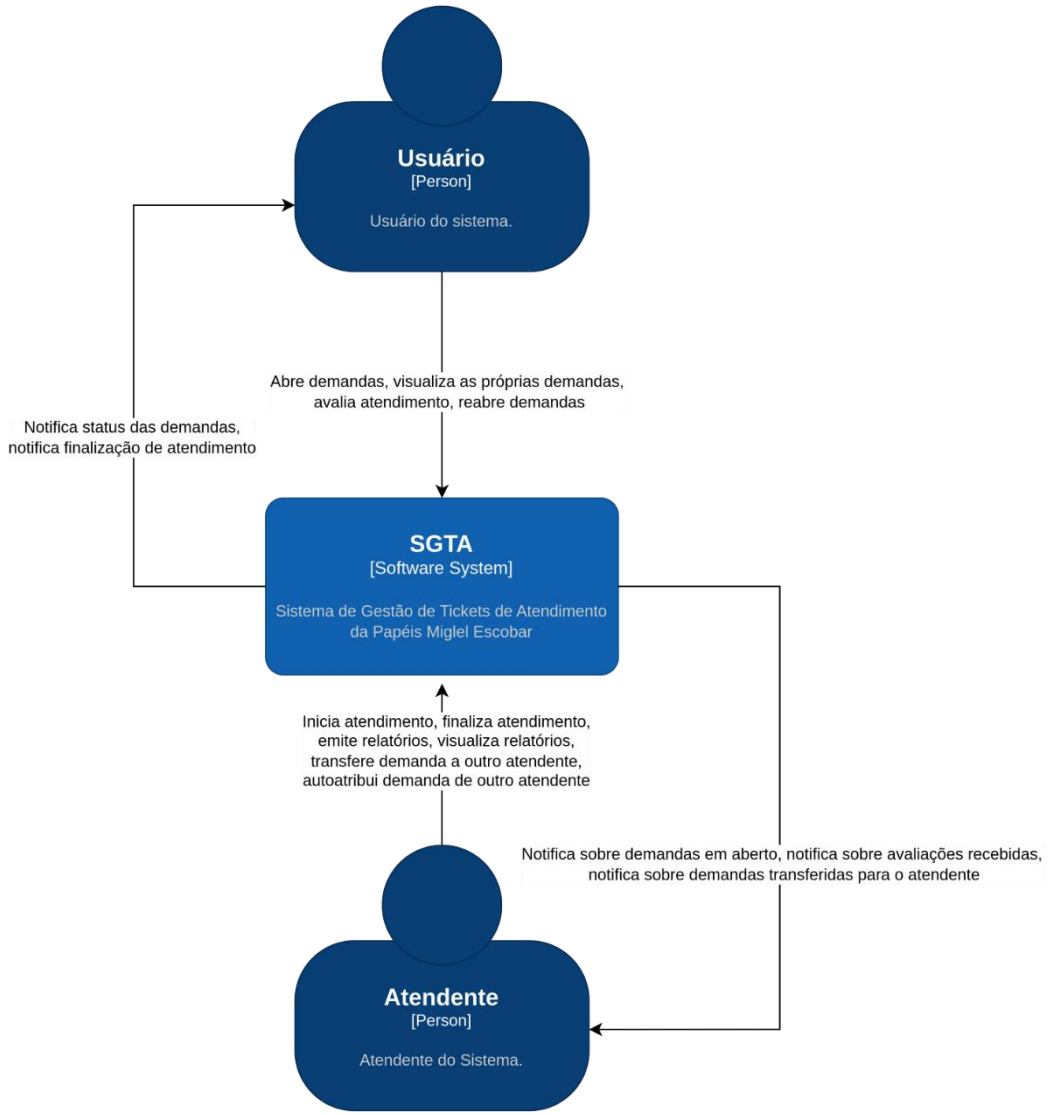

Figura. Diagrama de Contexto.

Em um contexto geral, o sistema se apoia nas interações entre 2 tipos de usuários principais: o usuário comum e o atendente. São estes que interagem diretamente com o sistema, enxergando o mesmo como uma caixa-preta que recebe as requisições e retorna resultados, oferecendo

uma interface unificada que permite que ambos os tipos de usuários se comuniquem por mensagens inseridas no sistema.

Nível 2: Diagrama de Containers

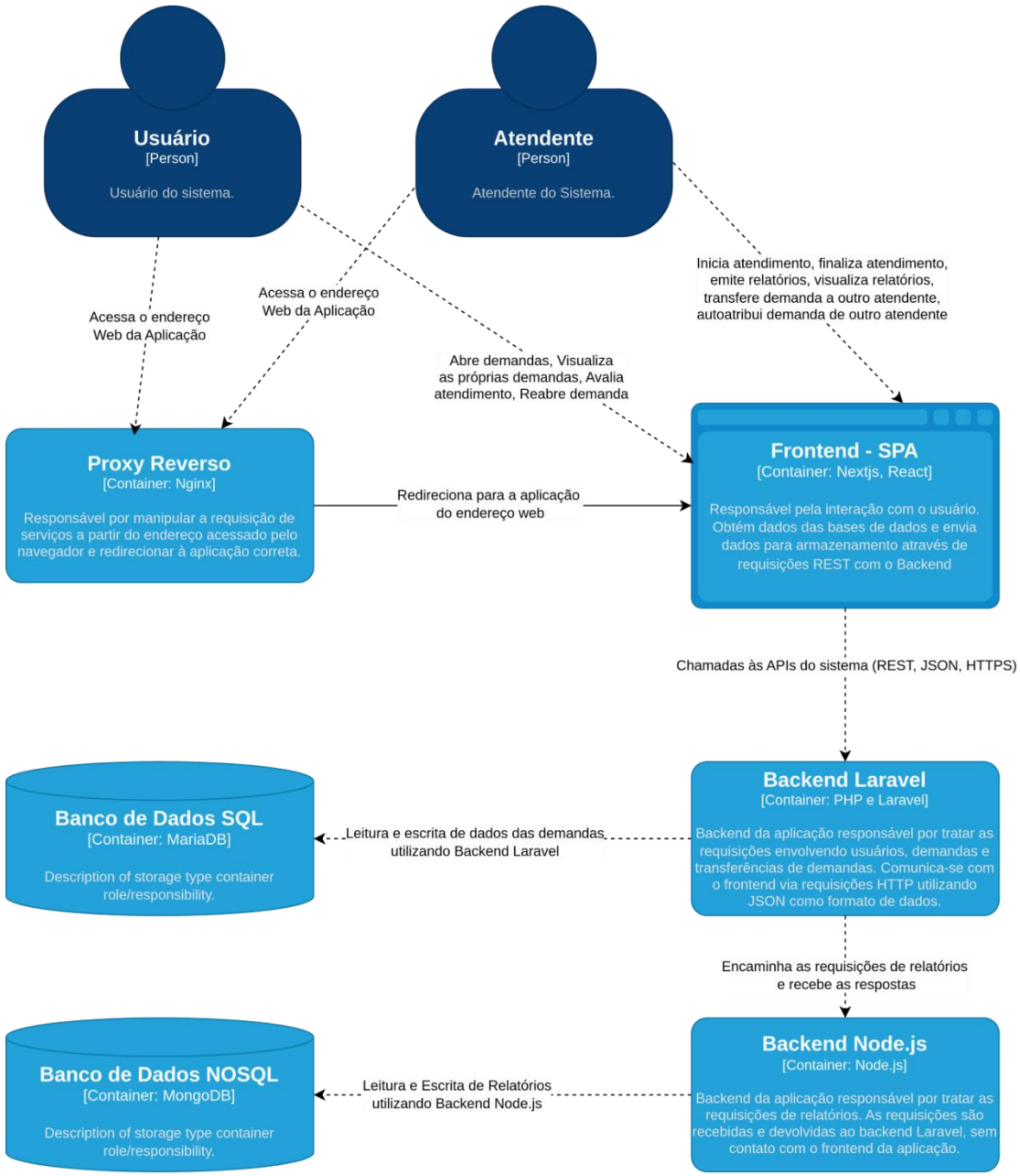

Figura. Diagrama de Contêineres.

Navegando de modo mais profundo no sistema, podemos visualizar os componentes macro do sistema (os contêineres), que neste caso são compostos por um Proxy Reverso, responsável por receber as requisições http/https e direcioná-las ao serviço correto. Temos uma aplicação SPA que é o que de fato os usuários e atendentes “enxergam”, a linha de frente com o cliente.

Em níveis mais afastados dos usuários em geral (usuários comuns e atendentes), chegamos aos contêineres de backend e de banco de dados.

Optou-se por dividir o backend em 2, sendo que o backend Node.js possui comunicação exclusiva a partir do backend Laravel - está isolado, nenhuma requisição que não seja originada do backend Laravel obterá qualquer resposta.

Essa decisão ocorreu por conta da facilidade e disponibilidade muito maior de bibliotecas que lidem com o Banco de Dados MongoDB disponíveis para Node.js junto com Express. Assim, divide-se a responsabilidade dos componentes do sistema, com o backend Laravel sendo responsável por lidar com requisições que envolvam consultas ou escrita na base de dados SQL MariaDB, e caso a requisição seja para uma rota que solicite dados da base de dados NOSQL MongoDB, esta é repassada para o backend Node.js, que se comunica com a base MongoDB, processa os dados solicitados e devolve via chamadas REST, em formato Json, a solicitação. Esta, então, é repassada para o frontend e se torna disponível ao usuário requisitante.

Nível 3: Diagrama de Componentes

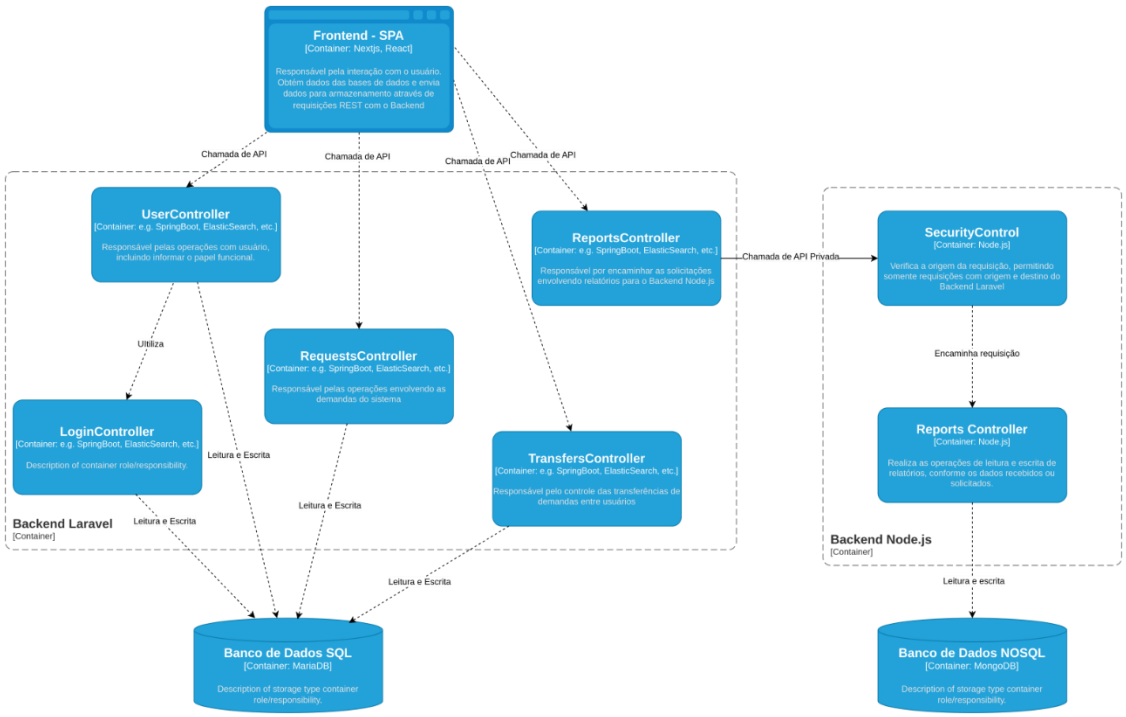

Figura. Diagrama de Componentes.

Dado que neste trabalho um MVP não será implementado, optou-se por elaborar o C4 Model somente até o terceiro nível. O nível de Código não será elaborado neste documento, sendo deixado para os próximos passos da construção do sistema, quando este se fizer imprescindível.