AtendeFácil: Estendendo uma plataforma de videoconferências para funcionar também como um Serviço de Atendimento ao Consumidor

Matheus de Araújo Correia Lima Melo

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

João Pessoa, PB

2024

Matheus de Araújo Correia Lima Melo

AtendeFácil: Estendendo uma plataforma de videoconferências para funcionar também como um Serviço de Atendimento ao Consumidor

Relatório Técnico apresentado ao curso Engenharia da Computação
do Centro de Informática, da Universidade Federal da Paraíba,
como requisito para a obtenção do grau de Bacharel em Engenharia da Computação

Orientadora: Thaís Gaudencio do Rêgo

Novembro de 2024

Catalogação na publicação
Seção de Catalogação e Classificação

M528a Melo, Matheus de Araújo Correia Lima.
AtendeFácil: estendendo uma plataforma de videoconferências para funcionar também como um serviço de atendimento ao consumidor / Matheus de Araújo Correia Lima Melo. - João Pessoa, 2024.
70 f. : il.

Orientação: Thaís Gaudencio do Rêgo.
TCC (Graduação) - UFPB/CI.

1. Backend. 2. API REST. 3. Websocket. 4. SAC. 5. Videoconferência. I. Rêgo, Thaís Gaudencio do. II. Título.

UFPB/CI CDU 004

Elaborado por Michelle de Kássia Fonseca Barbosa - CRB-738

CENTRO DE INFORMÁTICA
UNIVERSIDADE FEDERAL DA PARAÍBA

Trabalho de Conclusão de Curso de Engenharia da Computação intitulado ***AtendeFácil: Estendendo uma plataforma de videoconferências para funcionar também como um Serviço de Atendimento ao Consumidor*** de autoria de Matheus de Araújo Correia Lima Melo, aprovada pela banca examinadora constituída pelos seguintes professores:

Prof. Dra. Thaís Gaudencio do Rêgo
Universidade Federal da Paraíba

Prof. Dr. Yuri de Almeida Malheiros Barbosa
Universidade Federal da Paraíba

Me. José Leoberto Soares Filho

João Pessoa, 11 de novembro de 2024

Centro de Informática, Universidade Federal da Paraíba
Rua dos Escoteiros, Mangabeira VII, João Pessoa, Paraíba, Brasil CEP: 58058-600
Fone: +55 (83) 3216 7093 / Fax: +55 (83) 3216 7117

AGRADECIMENTOS

A conclusão dessa etapa da minha vida não teria sido possível sem o apoio e a presença de pessoas queridas que estiveram ao meu lado em diversos momentos da trajetória. Primeiramente, agradeço profundamente aos meus familiares que, mesmo nos momentos mais desafiadores, me acolheram e ajudaram a superar as dificuldades emocionais e de saúde. Sua força e apoio foram fundamentais para que eu pudesse continuar e concluir esta etapa.

Aos amigos, minha gratidão eterna por estarem ao meu lado, não apenas como companhia, mas como um lembrete constante de que há mais na vida além dos problemas. Em especial, agradeço a Pedro e Venâncio, com quem tive a alegria de dividir a universidade e um lar, e amigos como Catarina, Rebeca e Gabriel que, desde o colégio, somam positivamente em minha vida e me apoiaram nessa caminhada. Meu sincero agradecimento a Pardal, cujo incentivo mútuo foi essencial nesta reta final.

Aos colegas de turma e curso, que compartilharam desafios, risadas, perrengues e conquistas ao longo da trajetória acadêmica. Sua companhia e apoio foram fundamentais e inspiradores em diversos momentos.

Agradeço à empresa Wisecare, que disponibilizou o tema deste trabalho, apoiando diretamente minha formação. Sou grato ao meu chefe, Leoberto, cuja orientação e exemplo profissional muito me ensinaram. Foi uma honra contar com sua presença na banca de avaliação deste trabalho.

Por fim, agradeço aos professores do curso, especialmente ao Professor Yuri Malheiros, co-orientador deste trabalho, cujas orientações e dicas valiosas contribuíram diretamente para o sucesso desse trabalho, e à Professora Thaís Gaudencio, uma inspiração constante e apoio fundamental durante toda a graduação. Sua dedicação e empatia com os alunos foram luz e motivação para a conclusão deste sonho.

A todos que, direta ou indiretamente, contribuíram para minha jornada, deixo meu mais sincero muito obrigado.

RESUMO

Este relatório técnico descreve o processo de projeto, implementação e validação da funcionalidade AtendeFácil, integrada ao *backend* da plataforma de videoconferência PBMeet. O objetivo principal foi adicionar uma solução de Serviço de Atendimento ao Consumidor (SAC), que possibilitasse o atendimento por videoconferência, melhorando o escopo de uso da plataforma. A implementação incluiu uma API REST documentada, suportada por um sistema de comunicação em tempo real via *websocket*. A funcionalidade foi testada manualmente, com foco nas operações de criação, acompanhamento e finalização de solicitações de atendimento. Os resultados confirmam o sucesso da integração do AtendeFácil com o PBMeet, garantindo uma solução eficiente e bem estruturada para o gerenciamento de atendimentos. Sugestões para trabalhos futuros incluem a adição de novas funcionalidades, como um sistema de métricas e um *dashboard* de histórico de atendimentos.

Palavras-chave: *backend*, API REST, *websocket*, SAC, videoconferência

ABSTRACT

This technical report describes the design, implementation, and validation process of the AtendeFácil feature, integrated into the *backend* of the PBMeet videoconferencing platform. The main objective was to add a Customer Service (SAC) solution that allows customer support via videoconference, enhancing the platform's range of uses. The implementation included a well-documented REST API, supported by a real-time communication system using *websocket*. The functionality was manually tested, focusing on the operations of creating, tracking, and completing customer service requests. The results confirm the successful integration of AtendeFácil into PBMeet, ensuring an efficient and well-structured solution for managing customer service. Suggestions for future work include adding new features such as a metrics system and a dashboard for service history tracking.

Key-words: *backend*, REST API, *websocket*, SAC, *videoconferencing*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACL – *Access Control List*

BD – *Banco de Dados*

CSS – *Cascading Style Sheets*

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Transfer Protocol*

JS – *JavaScript*

JSON – *JavaScript Object Notation*

MVC – *Model-View-Controller*

SAC – *Serviço de Atendimento ao Consumidor*

SGBD – *Sistema gerenciador de banco de dados*

SQL – *Structured Query Language*

TS – *TypeScript*

UUID - *Universally Unique IDentifier*

Sumário

1 INTRODUÇÃO

Muitos são os casos em que, tanto em serviços públicos, quanto em particulares, acontece a necessidade de atendimento especializado para um tema ou problema específico. Isso pode acontecer somente para responder dúvidas sobre um produto ou serviço que se esteja oferecendo, ou para de fato resolver diretamente uma demanda de seu cliente.

Existem inúmeras empresas especializadas em atendimento ao consumidor, oferecendo esse serviços para grandes ou pequenas empresas. Esses serviços costumam se compor de uma enorme infraestrutura com funcionários que passam por um treinamento específico para atender solicitações de um determinado tema ou empresa em específico.

No Brasil, o Serviço de Atendimento ao Consumidor (SAC) é regulamentado principalmente pelo Decreto nº 6.523/2008, também conhecido como *"Lei do SAC"*[1], e pela sua atualização, o Decreto nº 11.034/2022, a *"Nova Lei do SAC"*[2]. Estas normas estabelecem as regras para o funcionamento do SAC e os direitos dos consumidores em relação a esse serviço, estipulando tanto parâmetros de qualidade de serviço, como tempo de espera, quanto os setores que são obrigados a fornecer o SAC, como serviços de telefonia, bancos, planos de saúde, companhias aéreas, entre outros serviços regulados.

Dentro desse contexto, surge a necessidade de soluções que possam modernizar e humanizar o atendimento, oferecendo alternativas ao modelo tradicional de *call center*. É nesse cenário que se insere o AtendeFácil, um serviço de *call center* por webconferência, que faz parte da funcionalidade do WiseAgenda/PBMeet.

O AtendeFácil proporciona uma experiência mais próxima e eficaz ao permitir, além do *chat*, o uso de vídeo para interações com os clientes. Essa ferramenta pode ser integrada diretamente ao site ou página inicial da empresa por meio de banners ou ícones, facilitando o acesso dos usuários que precisam de atendimento especializado. Um exemplo prático de sua aplicação é o "AtendeFácil DETRAN", onde os usuários podem esclarecer dúvidas sobre procedimentos como emplacamento de veículos, solicitação de documentos, pagamento de multas, entre outros.

Com o aumento da demanda por atendimentos mais ágeis e personalizados, o AtendeFácil surge como uma solução inovadora que alia tecnologia à eficiência no atendimento ao cliente.

[1]Disponível em: https://www.jusbrasil.com.br/legislacao/93373/lei-do-call-center-decreto-6523-08/. Acesso em: 07 de agosto de 2024

[2]Disponível em: https://www.jusbrasil.com.br/legislacao/1449962544/decreto-11034-22/. Acesso em: 07 de agosto de 2024

1.1 Objetivo geral

O objetivo geral do projeto é adicionar uma funcionalidade de SAC a uma plataforma de videconferências já existente, denominada PBMeet. Essa funcionalidade, denominada AtendeFácil, busca promover uma experiencia moderna, eficaz e humana na resolução de problemas envolvendo atendimento ao consumidor . Particularmente, esse relatório tem o foco no projeto *backend*, dessa funcionalidade, uma vez que que a interface foi desenvolvida por outra equipe.

1.2 Objetivos específicos

- Fazer o levantamento dos requisitos de modelo de dados e regras de negócio da aplicação
- A partir da interface, estipular e documentar as rotas da API REST *Node.JS*
- Extender o modelo de dados do PBMeet com as tabelas relacionadas ao AtendeFácil
- Implementar as rotas da aplicação específicas ao AtendeFácil
- Disponibilizar uma documentação viva do tipo Swagger para facilitar o desenvolvimento da interface
- Testar o funcionamento e as regras de negócio

1.3 Estrutura do relatório técnico

O presente relatório está dividido em 5 capítulos. No primeiro, é realizada a introdução ao tema e a definição do problema e dos objetivos do trabalho. No segundo, são expostos os principais conceitos e tecnologias utilizados para a construção da solução. No terceiro, são desenvolvidos os requisitos da aplicação proposta e a sua arquitetura. No quarto, são apresentados o plano de testes, um comparativo com plataformas semelhantes e uma análise sobre os resultados. No quinto e último capítulo são disponibilizadas as conclusões obtidas sobre o trabalho, bem como uma lista de sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados o PBMeet e os principais conceitos de tecnologia utilizados para construir a aplicação, com o foco no *backend*, além de especificar as ferramentas escolhidas.

2.1 O PBMeet

O PBMeet (Figuras 1-5) é um sistema web de criação e gerenciamento de reuniões e videoconferências desenvolvido pela Companhia de Processamento de Dados da Paraíba (CODATA), em parceria com a Universidade Federal da Paraíba (UFPB) e a Rede Nacional de Ensino e Pesquisa (RNP). Ele é utilizado para agendar e realizar reuniões virtuais, com funcionalidades como notificações por e-mail para os participantes. É uma solução voltada principalmente para órgãos e entidades do governo da Paraíba, oferecendo um ambiente seguro e customizado para videoconferências. Atualmente, é adotada como ferramenta padrão oficial para subsidiar as reuniões e encontros virtuais do Governo do Estado. [3].

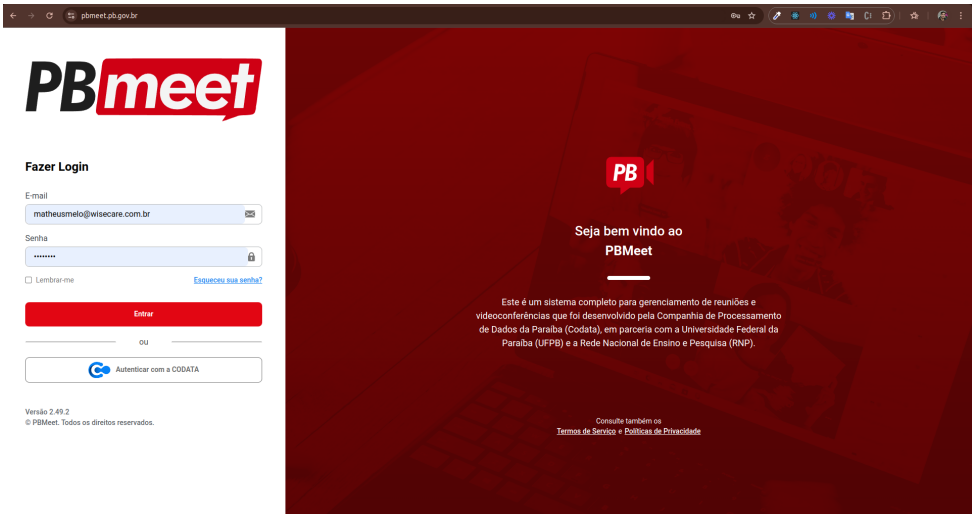

Figura 1: Tela de acesso ao PBMeet.

[3]Disponível em: https://paraiba.pb.gov.br/noticias/governo-digital-codata-lanca-plataforma-pbmeet-para-reunioes-remotas-dos-orgaos-estaduais

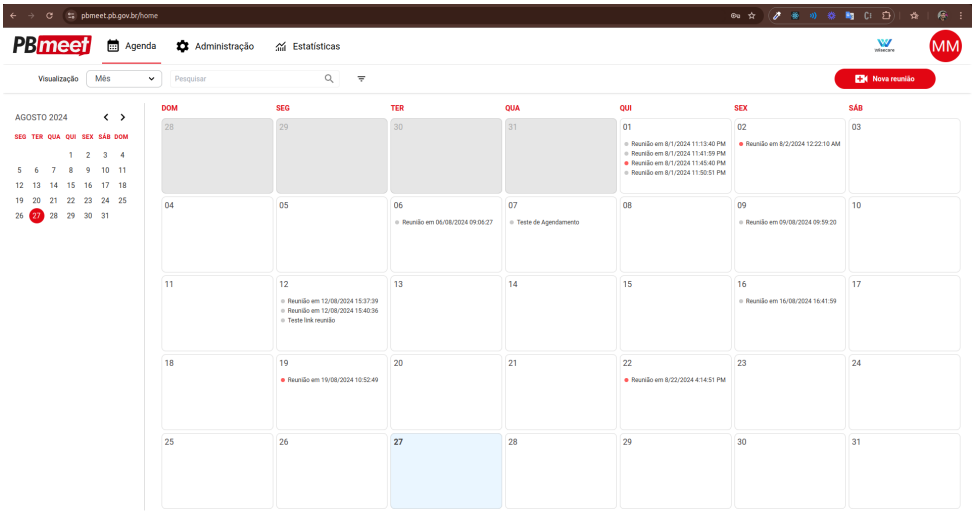

Figura 2: Tela Inicial do PBMeet.

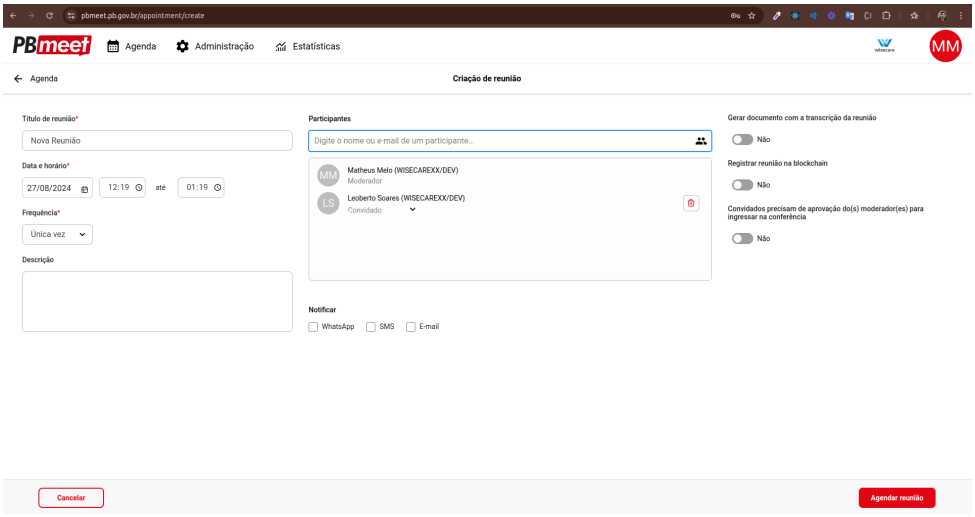

Figura 3: Tela de agendamento de uma videoconferência no PBMeet.

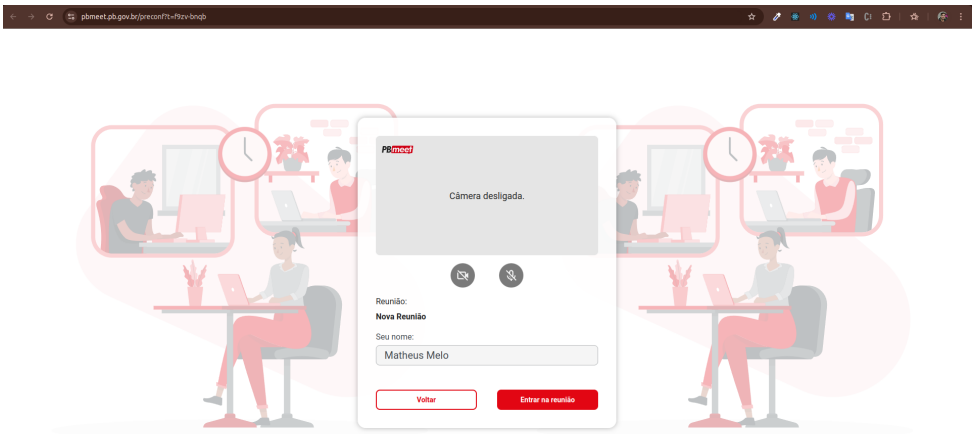

Figura 4: Tela de acesso a uma videoconferência no PBMeet.

Além das funcionalidades básicas de toda videoconferência, o PBMeet fornece funcionalidades extras, como acessibilidade dentro da reunião, transcrição do áudio e registro na *blockchain* do manifesto da videoconferência.

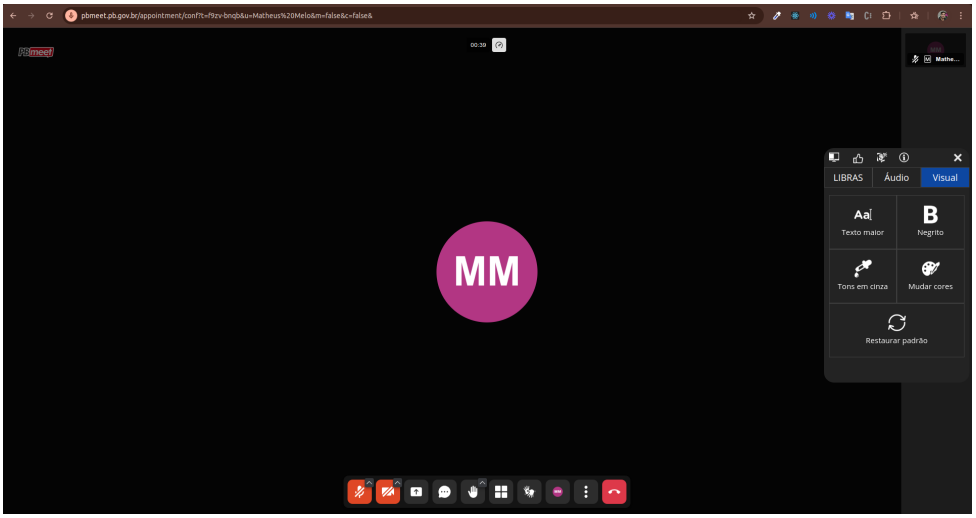

Figura 5: Uma videoconferência do PBMeet, com as funções de acessibilidade em evidencia.

2.2 Conceitos Gerais

Nesta seção, estão especificados os principais conceitos em que se fundamenta a solução proposta por nós.

2.2.1 *Frontend*

O ***frontend*** se refere à parte da aplicação com a qual os usuários interagem diretamente, i.e., a interface e seus elementos visuais [1]. Atua juntamente com o *backend*, a partir da realização de requisições de dados para os exibir em sua interface. É normalmente desenvolvido com o uso de: *Hypertext Markup Language* (HTML) para a estruturação, *Cascading Style Sheets* (CSS) para a estilização e *JavaScript* (JS) para a interatividade.

2.2.2 *Backend*

O ***backend*** se refere à parte da aplicação que é responsável por fazer o processamento, armazenamento e fornecimento dos dados, dados estes que são frequentemente gerados pelos usuários e são servidos ao *frontend*, comumente através de mensagens escritas em *JavaScript Object Notation* (JSON) . Geralmente é composto por três partes: o servidor, que é o hardware em si; a aplicação, que lida com as requisições do *frontend*, recebendo, processando e enviando dados; e o banco de dados [1].

2.2.3 Banco de Dados

O **banco de dados** trata essencialmente da associação entre uma coleção de dados, também chamada de base de dados, e um sistema gerenciador de banco de dados (SGBD), que permite o armazenamento, consulta, modificação e exclusão de dados de forma eficiente. Os bancos de dados podem ser relacionais, utilizando SQL como linguagem de consulta, ou não relacionais (NoSQL), oferecendo maior flexibilidade para tipos variados de dados [2].

2.2.4 API REST

A **API REST** (*Representational State Transfer)* é um conjunto de convenções para a criação de serviços web, que permite a comunicação entre diferentes sistemas através de requisições HTTP. As APIs RESTful são construídas em torno de recursos, que podem ser acessados e manipulados utilizando métodos HTTP como GET, POST, PUT, DELETE, entre outros [3].

2.2.5 DTOs

DTOs *(Data Transfer Objects)* são objetos que carregam dados entre diferentes camadas de uma aplicação. Eles são utilizados para encapsular os dados de uma forma que seja eficiente para a transmissão, sem expor a lógica de negócios, ou a estrutura interna dos objetos persistidos [4].

2.2.6 Autenticação, Autorização e ACL

A **autenticação** é o processo de verificação da identidade de um usuário ou sistema. A **autorização** refere-se à concessão de permissões específicas para acessar recursos ou executar ações dentro do sistema. Já as **Listas de Controle de Acesso (ACL)** são estruturas que definem as permissões de acesso de usuários, ou grupos, a determinados recursos ou funções do sistema [5].

2.2.7 *Controllers*

Os ***controllers*** são componentes da arquitetura de software que atuam como intermediários entre o *frontend* e o *backend*, recebendo as requisições HTTP, processando-as (muitas vezes utilizando serviços ou modelos) e retornando a resposta adequada ao cliente. Eles desempenham um papel crucial no padrão arquitetural *Model-View-Controller* (MVC) [6].

2.2.8 Swagger

Swagger é um conjunto de ferramentas para documentar, desenvolver e consumir APIs REST. Ele permite que desenvolvedores criem uma documentação interativa, que pode ser usada para testar *endpoints* e gerar clientes automaticamente para diferentes linguagens de programação [7].

2.2.9 ORM

O **ORM** (*Object-Relational Mapping)* é uma técnica que permite mapear classes de um sistema orientado a objetos para tabelas de um banco de dados relacional, facilitando as operações de persistência de dados. Com o ORM, desenvolvedores podem interagir com o banco de dados utilizando o paradigma orientado a objetos, abstraindo a complexidade das consultas SQL [8].

2.2.10 Repository

O **repository** é um padrão de design que separa a lógica de acesso a dados do resto da aplicação, fornecendo uma interface para realizar operações de leitura, escrita, atualização e exclusão de dados de forma abstrata. Ele permite que a aplicação mude a forma como os dados são persistidos, sem alterar a lógica de negócios [9].

2.2.11 Migration

Migrations são scripts que gerenciam alterações na estrutura do banco de dados de forma programática, garantindo que as mudanças de *schema* sejam aplicadas de maneira consistente e controlada, ao longo do ciclo de vida da aplicação. Eles são fundamentais em ambientes onde o banco de dados evolui juntamente com o código [10].

2.2.12 Inversão de Controle

A **inversão de controle (IoC)** é um princípio de design que desacopla a execução de tarefas específicas das classes que as realizam, transferindo a responsabilidade de controlar o fluxo da aplicação para um *container* ou *framework*. Isso facilita a testabilidade e a manutenção do código [11].

2.2.13 Arquitetura de Microserviços

A **arquitetura de microserviços** é um estilo arquitetural onde a aplicação é dividida em pequenos serviços independentes, que são implementados, testados e implan-

tados de maneira isolada. Cada microserviço possui seu próprio escopo, banco de dados e se comunica com outros serviços através de APIs [12].

2.2.14 Clean Architecture

A **Clean Architecture** é uma abordagem para a organização de código que enfatiza a separação de responsabilidades, promovendo a independência das regras de negócios em relação a *frameworks*, bancos de dados e interfaces de usuário. Ela se baseia na ideia de que o código deve ser estruturado em camadas, onde a camada central contém a lógica de negócios mais pura [13].

2.3 Tecnologias utilizadas

Nesta seção, estão especificadas as tecnologias utilizadas para o desenvolvimento de nossa solução, separadas de acordo com a área conceitual em que foi utilizada.

2.3.1 Gerais

- ***JavaScript***[4]: O JS é uma linguagem de programação interpretada de tipagem fraca. Utilizada amplamente no cenário de desenvolvimento Web, principalmente na construção de páginas interativas, também é usada para construir aplicações fora do navegador, como citado no tópico do *Node.js*. Faremos uso do JS para implementar a lógica de funcionamento da interface;

- ***TypeScript***[5]: O *TypeScript* é uma linguagem de programação de código aberto desenvolvida pela Microsoft. Esta linguagem é um superconjunto do JS, i.e., adiciona funcionalidades às já existentes do JS. Como indica seu nome, é de tipagem forte, o que possibilita uma maior previsibilidade sobre o funcionamento do código, de modo a facilitar a detecção de possíveis erros;

- ***Git***[6]: É uma ferramenta de controle de versão distribuído que permite o gerenciamento e rastreamento de alterações em arquivos de código-fonte ao longo do tempo. Utilizada pra manter todo o código desenvolvido, é fundamental para garantir segurança e flexibilidade na gestão do código.

[4]Documentação disponível em: https://developer.mozilla.org/en-US/docs/Web/JavaScript. Acesso em 19 de agosto de 2024.

[5]Website oficial disponível em: https://typescriptlang.org/. Acesso em: 1 de abril de 2024

[6]Documentação disponível em: https://git-scm.com/. Acesso em 19 de agosto de 2024

2.3.2 *Frontend*

- ***React***[7]: O *React* é uma biblioteca *frontend* para JS, de código aberto, desenvolvida pela *Meta*, então *Facebook*. Com o *React* é possível construir interfaces de usuário, a partir de partes chamadas "componentes", que podem ser reutilizadas várias vezes pela aplicação, a fim de facilitar a codificação da aplicação. O *React* também introduz uma nova extensão de arquivo, o *JavaScript XML* (JSX), que permite a combinação de HTML e JS em um único arquivo;

2.3.3 *Backend*

- ***Runtime* e *Framework***
  - ***Node.js***[8]: O *Node.js* é um ambiente de execução (*runtime*) de JS, no lado do servidor. Com ele, é possível executar JS fora de um navegador, e é justamente com ele que executamos nossa aplicação web.
  - ***Express***[9]: É um *framework* web minimalista para *Node.js* que facilita o desenvolvimento de APIs. O *Express* oferece um conjunto de recursos para lidar com requisições HTTP, gerenciamento de rotas e *middlewares*, possibilitando um desenvolvimento rápido e escalável de servidores web e uma melhor estruturação de aplicações de *backend*.
- **Banco de dados**
  - ***PostgreSQL***[10]: O *PostgreSQL* é um SGBD de código aberto para bancos de dados relacionais. O utilizamos para criar e manter todo o *schema* de banco de dados da aplicação.
- **ORM (*Object-Relational Mapping*)**
  - ***TypeOrm***[11]: É uma biblioteca de mapeamento objeto-relacional (ORM) para TypeScript. É uma ferramenta que garante uma interação mais eficiente com o Banco de Dados(BD), ao permitir que se defina entidades e relacionamentos do BD utilizando classes TypeScript, além de fornecer uma API para executar consultas, inserções, atualizações e exclusões no BD indiretamente, de forma simplificada, ou diretamente, com SQL puro, como o usuário preferir.
- ***Websocket***

[7]Website oficial disponível em: https://react.dev/. Acesso em: 19 de agosto de 2024
[8]Website oficial disponível em: https://nodejs.org/. Acesso em: 19 de agosto de 2024
[9]Website oficial disponível em: https://expressjs.com/. Acesso em: 19 de agosto de 2024
[10]Website oficial disponível em: https://postgresql.org/. Acesso em: 19 de agosto de 2024
[11]Website oficial disponível em: https://typeorm.io/. Acesso em: 19 de agosto de 2024

  - ***Socket.io***[12]: É uma biblioteca para comunicação em tempo real entre clientes e servidores via *websockets*. Foi a ferramenta escolhida para implementar a conexão *websocket* em tempo real entre os atendentes e os clientes da aplicação.

- **Validação de Dados**

  - ***Zod***[13]: É uma biblioteca de validação e esquemas para TypeScript e JS, projetada para validar e transformar dados com segurança e simplicidade. Fornece um ótimo suporte e integração com TypeScript, permitindo não só a manutenção da consistência dos dados que entram na aplicação, como também auxilia na criação de tipos automáticos de TypeScript, que facilitam o desenvolvimento.

- **Documentação da API**

  - ***Swagger - OpenAPI V3***[14]: É uma biblioteca que permite que desenvolvedores definam a estrutura das APIs, incluindo *endpoints*, métodos HTTP, parâmetros e tipos de resposta, em um formato legível, tanto para humanos, quanto para máquinas. Essa definição permite que seja gerada uma documentação automática em HTML, que pode ser visualizada pelos desenvolvedores e clientes da API através de uma rota da API.

[12]Website oficial disponível em: https://socket.io/. Acesso em: 19 de agosto de 2024
[13]Website oficial disponível em: https://zod.dev/. Acesso em: 19 de agosto de 2024
[14]Website oficial disponível em: https://swagger.io/specification/v3/. Acesso em: 19 de agosto de 2024

3 METODOLOGIA

Esse capítulo enuncia os métodos e procedimentos adotados no desenvolvimento desse projeto, a funcionalidade AtendeFácil da plataforma PBMeet. Além disso, fornece uma visão geral do uso da funcionalidade, descrevendo, também, os requisitos funcionais da aplicação, a arquitetura utilizada, sua interface e os casos de uso derivados a partir dela.

3.1 Visão Geral

O projeto visa desenvolver um novo módulo na API do PBMeet para agregar uma nova funcionalidade de SAC, permitindo que usuários do sistema sejam designados como atendentes e recebam de forma síncrona clientes com queixas, dúvidas ou demandas sobre um determinado tópico, cadastrado por administradores da plataforma.

Assim, a solução busca fornecer aos administradores do sistema PBMeet uma forma de integrar a sites e/ou aplicativos externos, de sua propriedade, um meio para que seus clientes possam, através da plataforma PBMeet, conversar com atendentes para solucionar um problema, sobre um tópico específico. A Figura 6 ilustra uma simplificação do comportamento desejado para a funcionalidade, do ponto de vista da interface.

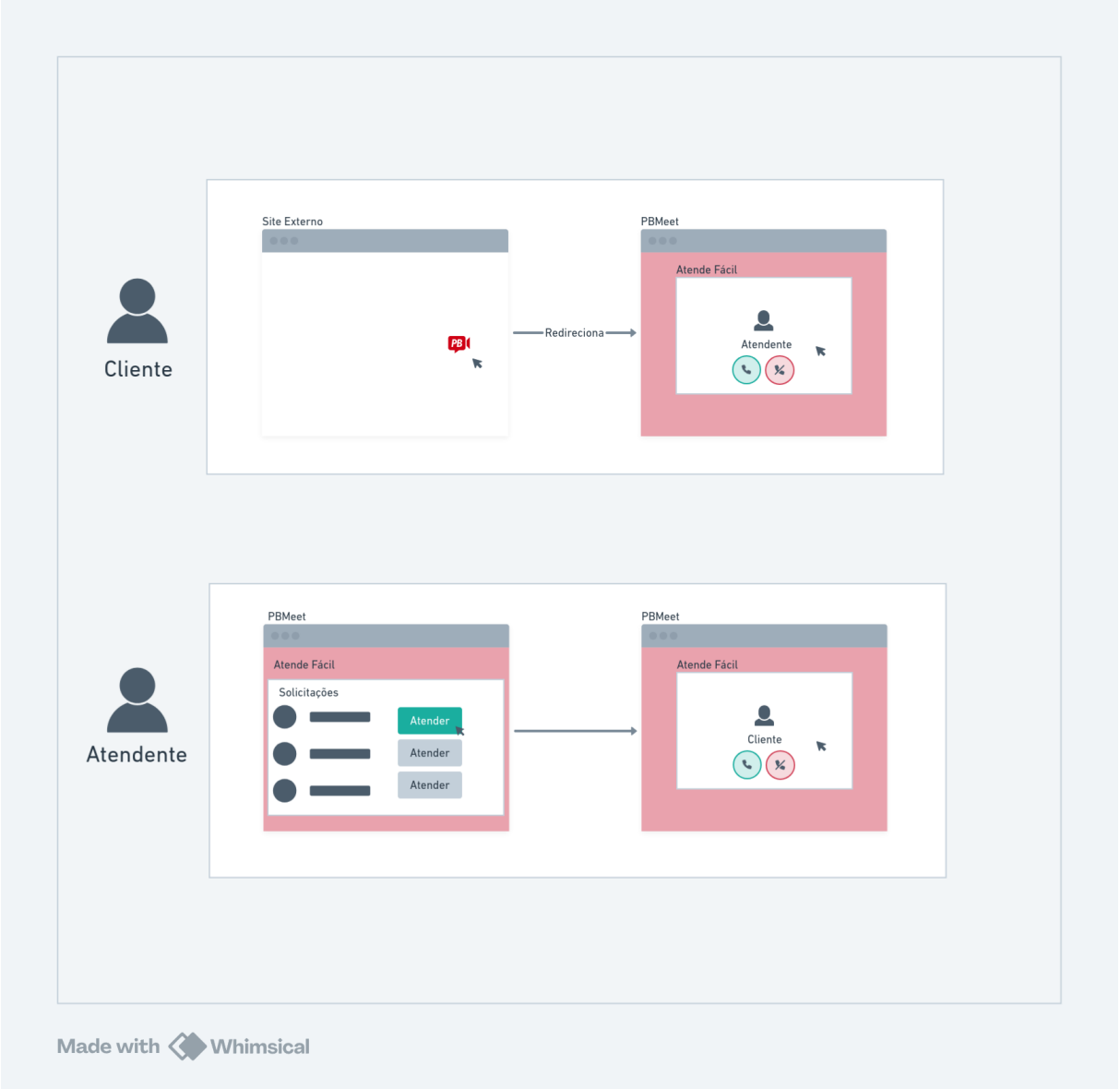

Figura 6: Simplificação do Fluxo de Interação de Atendente e Cliente.
Fonte: própria do autor

Para construir essa solução, o projeto foi desenvolvido com base nos objetivos e requisitos da funcionalidade, levando em consideração o código e a estrutura já existentes na API do PBMeet. As regras de negócio e o modelo de dados foram construídos de modo a estender as funcionalidades já presentes, minimizando o esforço de trabalho e garantindo uma boa integração com o código existente.

A organização do código da API do PBMeet, estruturada de acordo com os princípios da Clean Architecture, com módulos distintos e independentes, organizados em camadas conforme sua função e lógica de negócio, permitiu que a implementação da nova funcionalidade fosse tranquila e eficiente. A estrutura modular possibilitou a adição e modificação de componentes sem impactar outras partes do sistema, mantendo a coesão e facilitando a manutenção futura. Isso resultou em uma implementação ágil, com menor risco de erros

e maior facilidade para realização de testes.

Para elicitar os requisitos, primeiramente foram identificados os usuários que interagiriam com a plataforma, e os papéis que eles desempenhariam. Os papéis de Atendente, Cliente e Administrador foram definidos de acordo com os objetivos da funcionalidade. Em seguida, os recursos com os quais esses usuários interagiriam foram identificados, como os Tópicos e Solicitações. Finalmente, as ações de interação entre os usuários e os recursos foram definidas. Com base nisso, foi criado um modelo de dados e fluxo inicial para guiar o design da interface.

O design da interface da funcionalidade foi concebido considerando princípios de usabilidade e responsividade, e integrando-se à interface existente. A partir dela, as regras de negócio relacionadas ao módulo de SAC foram consolidadas e estabelecidas, definindo como os usuários iriam interagir com a plataforma e como a plataforma deveria responder a eles. A definição de rotas, entidades e mecanismos da API de SAC do PBMeet foi derivada desses requisitos, e, por fim, implementada.

Em fase final, os testes foram realizados diretamente na plataforma, verificando so resultados das interações entre os diferentes usuários e a nova funcionalidade implementada. Cada cenário de uso foi avaliado manualmente para assegurar que o sistema atendesse aos requisitos estabelecidos e que as interações ocorressem de forma correta e fluida. Esse processo permitiu identificar e corrigir problemas diretamente na interface, garantindo que a funcionalidade estivesse pronta para ser integrada ao ambiente de produção.

3.2 Usuários e Papéis

A intenção da funcionalidade é permitir que usuários de um serviço externo ao PBMeet sejam atendidos por usuários internos, em tópicos criados e gerenciados pelos administradores da plataforma. Assim foram definidos três papéis, ou tipos de usuários, que interagem com a plataforma: Cliente, Atendente e Administrador.

3.2.1 Cliente

O cliente é uma figura externa ao sistema, que deseja receber atendimento especializado quanto a um assunto específico da plataforma externa, em que a funcionalidade está sendo disponibilizada.

3.2.2 Atendente

Atendentes são usuários internos do PBMeet, que são designados para receber as solicitações dos clientes. Um usuário do PBMeet não necessariamente é um atendente,

ele precisa ser indicado pelo administrador da sua organização como atendente em um ou mais tópicos.

3.2.3 Administrador

Administradores são usuários internos do PBMeet com um nível de autorização e responsabilidade maior em uma determinada organização do sistema. São os administradores de uma organização que habilitam e configuram a funcionalidade de SAC para sua organização, criam tópicos de atendimento e designam usuários como atendentes dos mesmos.

Assim, em um cenário hipotético, um administrador da organização "DETRAN" do PBMeet habilita e configura a SAC para essa organização, indicando quais usuários serão atendentes de um tópico "Dúvidas sobre emplacamento". Com isso, no site externo do DETRAN, poderá ser incorporado um botão "AtendeFácil DETRAN", que redireciona os clientes para a fila de atendimento dentro do PBMeet.

3.3 Requisitos Funcionais

A seguir estão descritos os requisitos funcionais do sistema, que orientaram o desenvolvimento com a descrição detalhada das funcionalidades que o sistema deve oferecer e qual o comportamento do mesmo, conforme o usuário interage. Considerando os três diferentes fluxos da funcionalidade, um para cada tipo de usuário, os requisitos funcionais a seguir são descritos em função dos casos de uso da funcionalidade, que são diferentes para cada tipo de usuário.

3.3.1 Requisitos Funcionais para Usuário Administrador

3.3.1.1 [RFADM01] Configuração de SAC

O sistema deve permitir que um administrador realize o cadastro inicial de configurações da SAC. O administrador deverá fornecer informações de suporte para o SAC, como e-mail de contato, telefone e o horário de funcionamento da SAC. Depois de fornecer esses dados a SAC fica habilitada para ser utilizada.

3.3.1.2 [RFADM02] Botão de copiar código de incorporação

O sistema deve, após configurada a SAC de uma organização, fazer com que um botão de copiar código de incorporação fique disponível. Quando clicado, um código exemplo de incorporação do AtendeFácil numa interface externa será mostrado. Esse

código deverá conter um link externo com um *token* de autorização embutido, servindo de identificação para que sites externos integrem o AtendeFácil, com o redirecionamento para atendimento no PBMeet.

3.3.1.3 [RFADM03] Cadastro de tópicos (linha de atendimento)

O sistema deve permitir que um administrador cadastre e gerencie tópicos para uma organização específica. Cada tópico deve ter pelo menos um atendente vinculado e deve funcionar somente nos horários e dias designados para esse tópico.

3.3.1.4 [RFADM04] Cadastro de atendentes

O sistema deve permitir que um administrador designe ou remova usuários como atendentes de SAC para um tópico específico.

3.3.1.5 [RFADM05] Cadastro de horário de funcionamento

O sistema deve permitir que um administrador gerencie o horário de funcionamento da SAC, seja geral ou por tópico. Caso um tópico não tenha um horário de funcionamento específico, o da organização é utilizado.

3.3.1.6 [RFADM06] Desabilitar tópicos (linha de atendimento)

O sistema deve permitir que um administrador desabilite tópicos da SAC, para que eles não sejam mais listados como uma opção de atendimento. Esses tópicos ficam listados como desabilitados e não são permanentemente desabilitados, podendo ser reativados.

3.3.2 Requisitos Funcionais para Usuário Atendente

3.3.2.1 [RFATD01] Visualizar fila de atendimento

O sistema deve permitir que o usuário atendente visualize as solicitações de atendimento nos tópicos em que ele está habilitado como atendente. Nessa lista, atendente deverá escolher uma dessas solicitações por vez para atender.

3.3.2.2 [RFATD02] Visualizar lista de atendimentos finalizados

O sistema deve permitir que o usuário atendente visualize, separadamente, as solicitações de atendimento concluídas. Em cada solicitação aparecerá uma *tag* indicando o status final da solicitação, com as opções: transferido, resolvido, interrompido e não resolvido.

3.3.2.3 [RFATD03] Iniciar atendimento

O sistema deve permitir que o usuário atendente inicie um atendimento a partir da lista de solicitações. Depois de iniciado o atendimento, o atendente espera a chegada do solicitante dentro da videoconferência, ao mesmo tempo que o último é notificado que sua vez chegou.

3.3.2.4 [RFATD04] Informações dentro da conferência

O sistema deve permitir que o usuário atendente visualize, dentro da conferência, as informações relevantes para o atendimento da solicitação atual, como nome e identificação do cliente, o assunto (tópico) e a descrição da solicitação.

3.3.2.5 [RFATD05] Adicionar arquivos de apoio associados ao atendimento

O sistema deve permitir que o usuário atendente adicione à solicitação documentos e/ou outros arquivos que achar relevante para o caso. Esses arquivos poderão ser áudios, vídeos ou documentos e terão no máximo 500MB de tamanho. Esses arquivos só poderão ser adicionados ou removidos enquanto estiver acontecendo o atendimento, mas poderão ser recuperados depois que estiver finalizado.

3.3.2.6 [RFATD06] Adicionar anotações associadas ao atendimento

O sistema deve permitir que o usuário atendente adicione a solicitação uma anotação em texto que pode ser editada durante todo o atendimento e que persistirá após ele ser finalizado.

3.3.2.7 [RFATD07] Transferir atendimento

O sistema deve permitir que o usuário atendente transfira o atendimento para outro tópico, caso ache necessário. Ele deverá incluir uma breve descrição do motivo de transferência ao fazer isso. Depois de transferido o atendimento, tanto o atendente, como o cliente, são desconectados da videoconferência e o cliente volta para a fila de atendimento, dessa vez para a fila do tópico para a qual foi transferido.

3.3.2.8 [RFATD08] Visualizar histórico de anotações do atendimento

O sistema deve permitir que, caso a solicitação de atendimento tenha sido transferida, o próximo atendente a lidar com a solicitação possa visualizar uma lista com o histórico de anotações para entender melhor o problema. Nessa lista deve existir, além das anotações, os nomes dos responsáveis e as datas e horários em que foram inseridas.

3.3.2.9 [RFATD09] Finalizar o atendimento

O sistema deve permitir que o atendente finalize o atendimento da solicitação. Ele deverá escolher entre três opções : resolvido, não resolvido e interrompido (caso ele perca a conexão com o cliente). Para cada uma das opções, o atendente deverá incluir uma descrição para explicar sua decisão.

3.3.3 Requisitos Funcionais para Usuário Cliente

3.3.3.1 [RFCLT01] Ingressar em uma fila de atendimento

Estando dentro do horário de funcionamento da SAC e do tópico, clientes podem ingressar em uma fila para serem atendidos. A partir do site externo, o cliente deverá ser redirecionado para o PBMeet, onde selecionará o assunto (tópico) da sua solicitação e adicionará, em texto, uma descrição do seu problema ou dúvida. Além disso, o cliente também deverá se identificar, com nome, CPF, e-mail e telefone.

3.3.3.2 [RFCLT02] Aviso de que não há atendentes disponíveis

O sistema deve avisar ao usuário atendente que não há atendentes disponíveis, caso o mesmo tente solicitar atendimento fora do horário de atendimento do tópico e/ou da organização.

3.3.3.3 [RFCLT03] Sala de espera

O sistema deve direcionar o usuário cliente para uma "sala de espera", onde informações relevantes ao seu atendimento serão exibidas e onde ele será notificado quando sua vez chegar.

3.3.3.4 [RFCLT04] Cancelar solicitação

O sistema deve permitir que o usuário cliente, depois de entrar na sala de espera, cancele sua solicitação se assim desejar, saindo, então, da fila de atendimento.

3.3.3.5 [RFCLT05] Entrar no atendimento

O sistema deve permitir que o usuário cliente entre na videoconferência para receber atendimento. Isso deverá acontecer assim que algum atendente aceitar a solicitação. O cliente terá um intervalo de tempo, a partir da solicitação aceita, para entrar no atendimento, caso esse tempo se esgote, ele poderá escolher se volta mais uma vez para a fila de atendimento, ou se desiste da solicitação.

3.3.3.6 [RFCLT06] Informações dentro da conferência

O sistema deve permitir que o usuário cliente verifique, dentro da videoconferência, os detalhes de sua solicitação, incluindo os dados que ele mesmo inseriu, bem como um identificador do atendimento, que servirá de número de protocolo, caso o cliente deseje identificar posteriormente esse atendimento.

3.3.3.7 [RFCLT07] Adicionar arquivos de apoio associados ao atendimento

Assim como acontece para o atendente, o sistema deve permitir que o usuário cliente adicione à solicitação documentos e/ou outros arquivos que achar relevante para o seu caso. Esses arquivos poderão ser áudios, vídeos ou documentos e terão no máximo 500MB de tamanho.

3.4 Requisitos Não Funcionais

Os requisitos não funcionais descritos a seguir foram considerados essenciais para garantir a usabilidade e segurança da funcionalidade AtendeFácil.

3.4.0.1 [RNF01] Autenticação e Autorização

Todo acesso ao PBMeet, e consequentemente, ao AtendeFácil, deve ser controlado por mecanismos de autenticação e autorização para assegurar que apenas usuários autenticados possam acessar a plataforma e suas funcionalidades. O sistema deve verificar a identidade do usuário e aplicar uma Lista de Controle de Acesso (ACL), garantindo que as permissões estejam devidamente configuradas para cada tipo de usuário.

3.4.0.2 [RNF02] Documentação e Manutenibilidade

A API REST do AtendeFácil deve ser documentada de forma clara e precisa, utilizando o Swagger para fornecer uma documentação interativa. Isso facilita o desenvolvimento e a manutenção contínua da plataforma, permitindo que novos desenvolvedores compreendam a estrutura e os endpoints da API de forma mais ágil.

3.5 Arquitetura

Em sua arquitetura, o AtendeFácil segue a Figura 7. O usuário interage com interface da aplicação, o *frontend* desenvolvido em *React*, que, por sua vez, realiza requisições *HTTP* ao *backend*, sendo esse um servidor *Node.JS*. O servidor valida as credenciais do usuário, bem como sua ACL, e, caso esteja tudo de acordo, garante-se o sucesso dessas

requisições *HTTP*, que realizam inserções, atualizações e consultas ao banco de dados, obedecendo as regras de negócio que o sistema impõe, validando tudo que entra e sai do servidor e manipulando as informações necessárias.

Além disso, algumas das funcionalidades do AtendeFácil demandam uma conexão em tempo real dos atores envolvidos, disparando eventos e notificações aos usuários, à medida que alguma condição é atendida. Assim, para atingir esses objetivos, tanto o *frontend React* quanto o servidor *Node.JS* se conectam a um servidor *websocket*, que mantém e gerencia a conexão simultânea entre os usuários e direciona os eventos a quem precisa recebê-los.

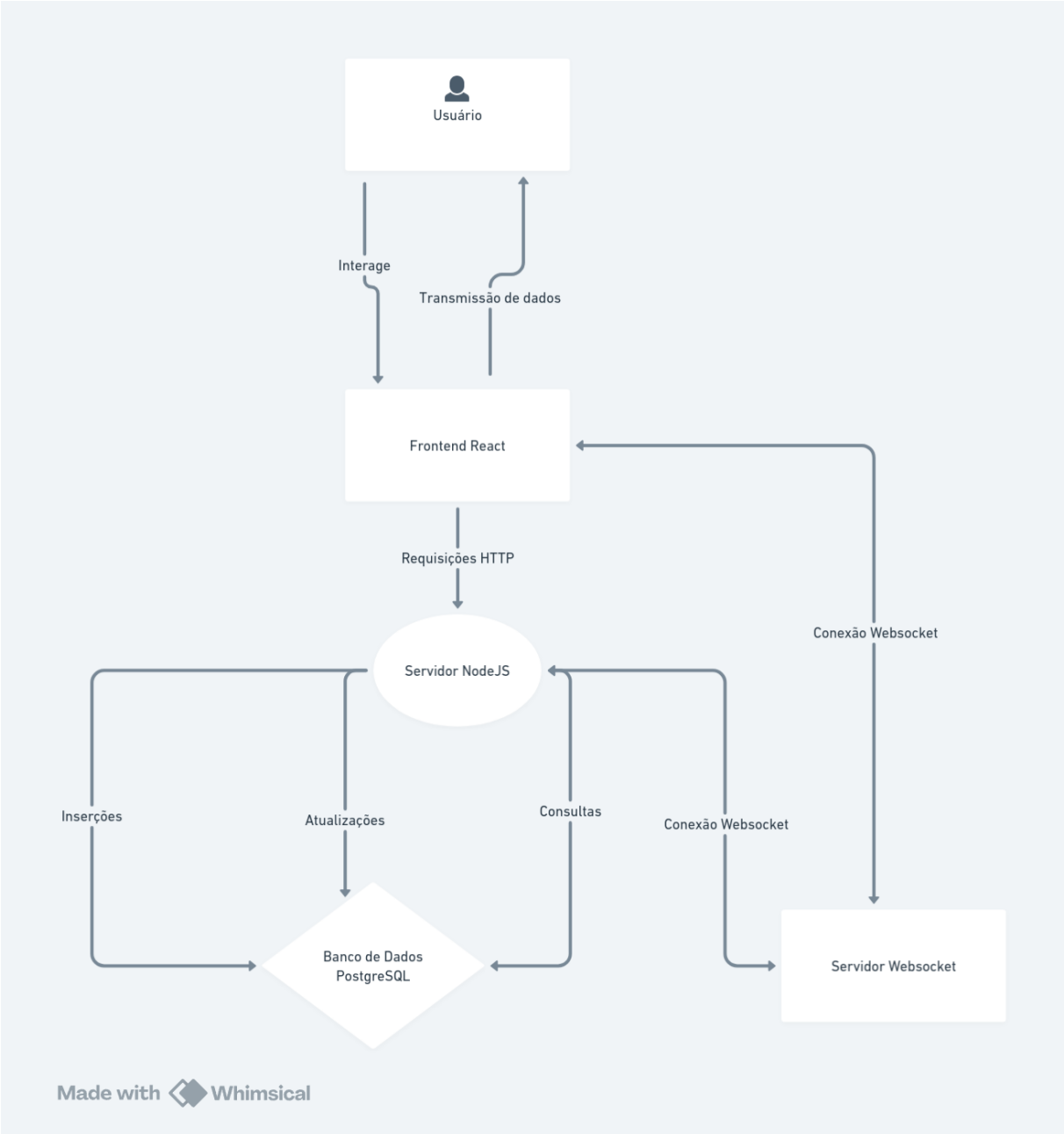

Figura 7: Arquitetura da aplicação.
Fonte: própria do autor

3.5.1 Banco de dados

O modelo de dados existente na plataforma foi estendido com novas tabelas, algumas delas se relacionando com outras já existentes no modelo de dados. As tabelas a seguir documentam as adicionados ao banco de dados e o diagrama na figura 8 mostra a relação entre essas entidades.

Para atender aos requisitos **RFATD05**, **RFCLT07**, relacionados aos arquivos de apoio, associados a uma solicitação de atendimento a Tabela **conferenceFile** foi inserida ao *schema* do banco de dados. Já a Tabela **conferenceNote** atende os requisitos **RFATD06** e **RFATD08** relacionados as anotações de um atendimento. Essas entidades poderiam ser associadas diretamente a um registro de solicitação de atendimento **sacRequest**, mas foram intencionalmente associadas a tabela já existente **conference**, para mantê-las genéricas e garantir que essas funcionalidades possam ser utilizadas em outras conferências do PBMeet, não apenas os do AtendeFácil.

A Tabela **sacRequest** é a principal, ela define uma solicitação de atendimento e todas as demais se associam a elas de alguma forma. A **sacSpecialist** define o novo tipo de usuário **Atendente** associando esse papel com a Tabela de Usuários do sistema, já a Tabela **sacTopic** define as *linhas de atendimento* ou *tópicos* de uma organização do sistema e permite que solicitações sejam direcionadas para categorias e atendentes específicos. Por fim, a Tabela **sacSchedule** é uma tabela auxiliar que define dias e horários de funcionamento da SAC para uma organização e/ou uma *linha de atendimento*.

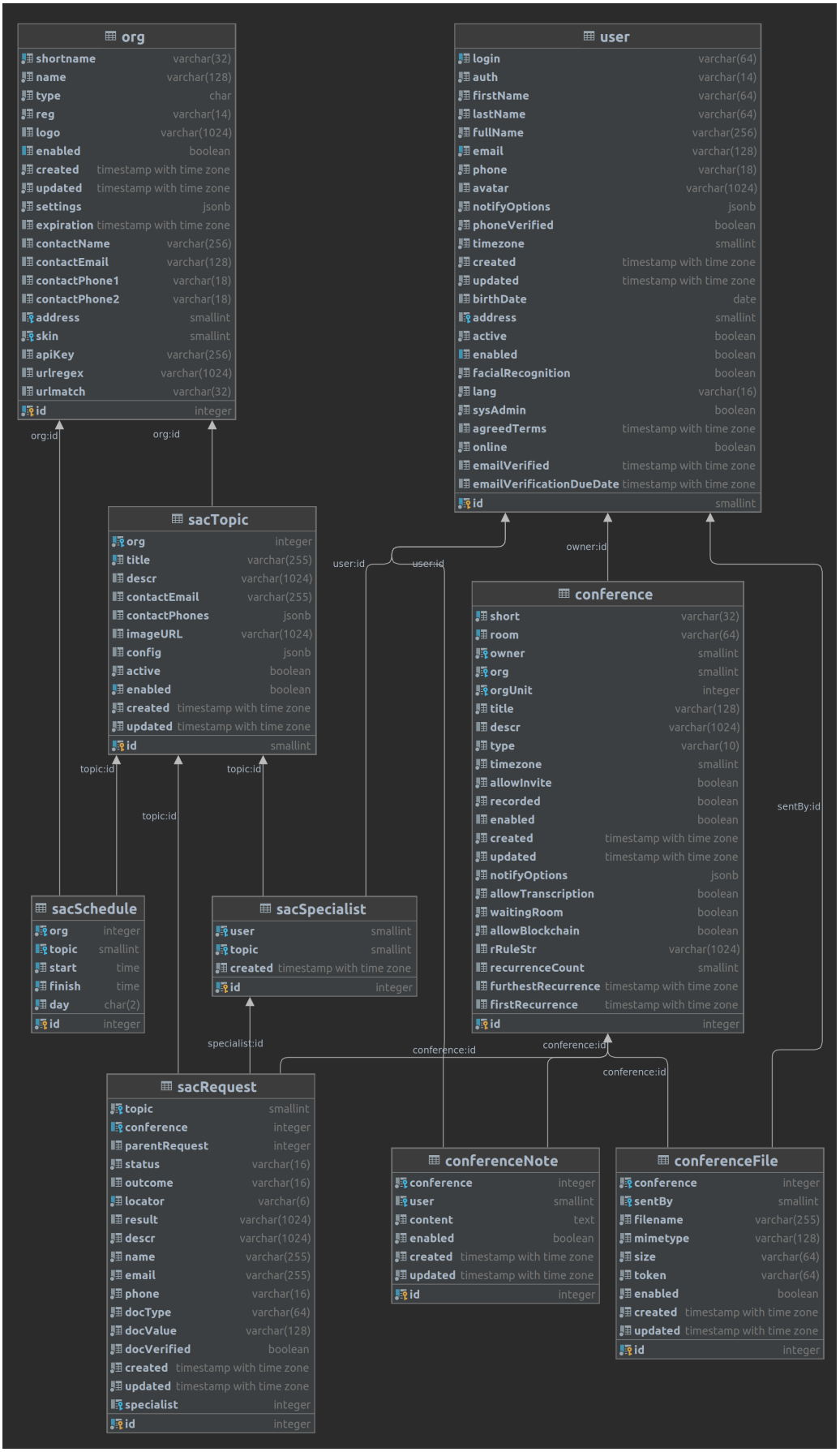

Figura 8: Diagrama entidade-relacionamento.
Fonte: própria do autor

- **Tabela: conferenceFile**
  Arquivos de uma conferência.

Atributo	Descrição	Tipo	Observações
id	Identificador único serial.	**serial**	PK
conference	Identificador da conferência.	**serial**	FK → conference(id)
sentBy	Identificador do usuário.	**serial**	FK → user(id), NULLABLE
filename	Nome do arquivo.	**varchar(255)**	
mimetype	Tipo do arquivo.	**varchar(128)**	
size	Tamanho do arquivo em bytes.	**varchar(64)**	
token	Token do arquivo no storage.	**varchar(64)**	
enabled	Booleano responsável pela deleção lógica.	**boolean**	Default: True
created	Timestamp da criação deste registro.	**timestamptz**	
updated	Timestamp da atualização deste registro.	**timestamptz**	

- **Tabela: conferenceNote**
  Anotações de uma conferência.

Atributo	Descrição	Tipo	Observações
id	Identificador único serial.	**serial**	PK
conference	Identificador da conferência.	**serial**	FK → conference(id)
user	Identificador do usuário.	**serial**	FK → user(id)
content	Conteúdo das anotações.	**text**	
enabled	Booleano responsável pela deleção lógica.	**boolean**	Default: True
created	Timestamp da criação deste registro.	**timestamptz**	
updated	Timestamp da atualização deste registro.	**timestamptz**	

- **Tabela: sacSchedule**
  Agendamento de disponibilidade de uma SAC ou Tópico.

Atributo	Descrição	Tipo	Observações
id	Identificador único serial.	**serial**	PK
org	Identificador serial da organização.	**serial**	FK → Org
topic	Identificador serial do sacTopic. NULL quando a configuração for para a organização inteira.	**serial**	FK → sacTopic, NULLABLE

Atributo	Descrição	Tipo	Observações
start	Hora e minuto que o agendamento fica disponível.	**time**	Formato hh:mm
finish	Hora e minuto que o agendamento fica indisponível.	**time**	Formato hh:mm
day	Dia da semana representado por um char(2). Deve ser único para uma mesma org ou tópico.	**char(2)**	MO, TU, WE, TH, FR, SA, SU

- **Tabela: sacTopic**

  Tópicos de discussão cadastrados para cada organização.

Atributo	Descrição	Tipo	Observações
id	Identificador único serial.	**serial**	PK
org	Identificador da organização que o tópico pertence.	**serial**	FK → Org(id)
title	Título que aparece para seleção.	**varchar(255)**	UNIQUE
descr	Descrição detalhada que aparece como hint.	**varchar(1024)**	NULLABLE
contactEmail	E-mail de contato.	**varchar(255)**	NULLABLE
contactPhones	Telefones de contato. Array JSON.	**jsonb**	NULLABLE
imageURL	URL da imagem que será mostrada na apresentação.	**varchar(1024)**	NULLABLE
config	JSON com atributos que indicam: – requer identificação na entrada (nome, CPF, ...) – se a transcrição está habilitada – se a gravação está habilitada – se deve verificar CPF – se a pesquisa de satisfação está habilitada	**jsonb**	NULLABLE
active	Booleano responsável por habilitar o tópico.	**boolean**	Default: true
enabled	Booleano responsável pela deleção lógica.	**boolean**	Default: true
created	Timestamp da criação deste registro.	**timestamptz**	
updated	Timestamp da atualização deste registro.	**timestamptz**	

- **Tabela: sacRequest**

Requisição para SAC.

Atributo	Descrição	Tipo	Observações
id	Identificador único serial.	**serial**	PK
topic	Identificador do tópico que o usuário quer atendimento.	**serial**	FK → sacTopic(id)
conference	Identificador da conferência.	**serial**	FK → conference(id), NULLABLE
specialist	Identificador do especialista de atendimento.	**serial**	FK → sacSpecialist(id), NULLABLE
parentRequest	Identificador da requisição PAI, ou seja, requisição de transferência.	**serial**	FK → sacRequest(id), NULLABLE
status	Status da solicitação: – WAITING: O usuário se encontra esperando atendimento – ATTENDING: O usuário se encontra em atendimento – LEFT: O usuário deixou o SAC – FINISHED: Atendimento finalizado	**varchar(16)**	
outcome	Status de finalização: – RESOLVED: Solicitação resolvida com sucesso – UNRESOLVED: Solicitação não resolvida – TRANSFERRED: Solicitação transferida de tópico	**varchar(16)**	NULLABLE
locator	Localizador gerado automaticamente ao entrar na fila, que servirá como protocolo. Formato: base58 com 6 caracteres.	**varchar(6)**	UNIQUE
result	Texto explicando os resultados do atendimento.	**varchar(1024)**	NULLABLE
descr	Descrição da situação.	**varchar(1024)**	
name	Nome do usuário.	**varchar(255)**	

Atributo	Descrição	Tipo	Observações
email	E-mail.	**varchar(255)**	
phone	Telefone.	**varchar(16)**	
docType	Tipo de documento fornecido ao entrar na fila: – CPF – RG – CNH – PASSPORTE	**varchar(64)**	
docValue	Valor do documento conforme o tipo, digitado pelo usuário.	**varchar(128)**	
docVerified	Indica se o documento foi verificado. Ex.: no caso do CPF, se foi consultado na base do ministério da fazenda.	**boolean**	Default: False
created	Timestamp da criação deste registro.	**timestamptz**	
updated	Timestamp da atualização deste registro.	**timestamptz**	

- **Tabela: sacSpecialist**
  Especialista do SAC.

Atributo	Descrição	Tipo	Observações
id	Identificador único serial.	**serial**	PK
user	Identificador do usuário.	**serial**	FK → user(id)
topic	Identificador do tópico.	**serial**	FK → sacTopic(id)
created	Timestamp da criação deste registro.	**timestamptz**	

3.5.2 *Endpoints*

A seguir estão descritos todos os *endpoints* implementados para satisfazer os requisitos da funcionalidade AtendeFácil.

3.5.2.1 Enviar arquivos de apoio da conferência

Esta requisição tem a finalidade de enviar arquivos de apoio durante uma conferência.

POST /api/v1/conferences/{short}/files

Authentication: APIKEY {apiKey}

Atributos de URL

- **short**: string, obrigatório, não nulo, não vazio — Identificador da conferência.

Atributos de entrada

- **files**: array (file), obrigatório, não nulo.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*conference*)

3.5.2.2 Listar arquivos de apoio da conferência

Esta requisição tem a finalidade de listar arquivos de apoio durante uma conferência.

POST /api/v1/conferences/{conferenceShort}/files/list

Authentication: APIKEY {apiKey}

Atributos de URL

- **short**: string, obrigatório, não nulo, não vazio — Identificador da conferência.

Atributos de entrada

- **dataControl**: objeto, opcional
  - *limit*: number, opcional
  - *offset*: number, opcional
  - *paging*: boolean, opcional
- **orderBy**: array(objeto), opcional
  - *order*: number, opcional
  - *attribute*: string, obrigatório, enum: {'ID', 'SENTBY', 'CREATED'}
  - *direction*: string, obrigatório, enum: {'ASC', 'DESC'}
- **filters**: objeto, opcional
  - *id*: number, opcional
  - *sentBy*: string, opcional

Status de resposta esperados

- **200 OK** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*conference*)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "metadata": {
    "offset": 0,
    "limit": 10,
    "total": 100
  },
  "records": [
    {
      "id": 1,
      "filename": "filename.pdf",
      "mimeType": "PDF",
      "size": "312312",
      "sentBy": {
        "id": 1,
        "fullname": "João Silva",
        "avatar": "https://path.to.image/1231"
      },
      "created": "2023-09-28T19:28:12.825Z",
      "updated": "2023-09-28T19:28:12.825Z"
    }
  ]
}
```

3.5.2.3 Remover arquivos de apoio da conferência

Esta requisição tem a finalidade de remover arquivos de apoio de uma conferência.

DELETE /api/v1/conferences/{short}/files/{file}

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CONFERENCE	UPDATE_ANY	Remover arquivos em conferências de qualquer organização
CONFERENCE	UPDATE_ORG	Remover arquivos em conferências de organização que administra
CONFERENCE	UPDATE_UNIT	Remover arquivos em conferências de setor que administra
CONFERENCE	UPDATE_OWN	Remover arquivos em conferências em que é proprietário

Tabela 7: Permissões ACL para remoção de arquivos da conferência

Atributos de URL

- **short**: string, obrigatório, não nulo, não vazio — Identificador da conferência.
- **file**: string, obrigatório, não nulo, não vazio — Identificador do arquivo.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*conference*, *file*)

3.5.2.4 Download arquivo de apoio da conferência

Esta requisição tem a finalidade de baixar os arquivos de apoio de uma conferência. Rota aberta.

GET /api/v1/conferences/{short}/files/{file}/DOWNLOAD

Authentication: APIKEY {apiKey}

Atributos de URL

- **short**: string, obrigatório, não nulo, não vazio — Identificador da conferência.
- **file**: string, obrigatório, não nulo, não vazio — Identificador do arquivo.

Status de resposta esperados

- **200 OK** – Sucesso (arquivo)
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (tópico)

3.5.2.5 Enviar anotação da conferência

Esta requisição tem a finalidade de enviar anotações durante uma conferência.

POST /api/v1/conferences/{short}/notes

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CONFERENCE	UPDATE_ANY	Adicionar anotações em conferências de qualquer organização
CONFERENCE	UPDATE_ORG	Adicionar anotações em conferências de organização que administra
CONFERENCE	UPDATE_UNIT	Adicionar anotações em conferências de setor que administra
CONFERENCE	UPDATE_OWN	Adicionar anotações em conferências em que é proprietário

Tabela 8: Permissões ACL para envio de anotação da conferência

Atributos de URL

- **short**: string, obrigatório, não nulo, não vazio — Identificador da conferência.

Atributos de entrada

- **note**: string, obrigatório, não nulo.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*conference*)

3.5.2.6 Listar anotações da conferência

Esta requisição tem a finalidade de listar anotações de uma conferência.

POST /api/v1/conferences/{short}/notes/list

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CONFERENCE	READ_ANY	Listar anotações em conferências de qualquer organização
CONFERENCE	READ_ORG	Listar anotações em conferências da organização que administra
CONFERENCE	READ_UNIT	Listar anotações em conferências de setor que administra
CONFERENCE	READ_OWN	Listar anotações em conferências que é proprietário
CONFERENCE	READ_ATTENDEE	Listar anotações em conferências que é participante

Tabela 9: Permissões ACL para listar anotações da conferência

Atributos de URL

- **short**: string, obrigatório, não nulo, não vazio — Identificador da conferência.

Atributos de entrada

- **dataControl**: objeto, opcional
  - *limit*: number, opcional
  - *offset*: number, opcional
  - *paging*: boolean, opcional
- **orderBy**: array (objeto), opcional
  - *order*: number, opcional

  - *attribute*: string, obrigatório, enum: {'ID', 'SENTBY', 'CREATED'}
  - *direction*: string, obrigatório, enum: {'ASC', 'DESC'}
- **filters**: objeto, opcional
  - *id*: number, opcional
  - *sentBy*: string, opcional

Status de resposta esperados

- **200 OK** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*conference*)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "metadata": {
    "offset": 0,
    "limit": 10,
    "total": 100
  },
  "records": [
    {
      "id": 1,
      "note": "Anotação do usuário",
      "sentBy": {
        "id": 1,
        "fullname": "João Silva",
        "avatar": "https://path.to.image/1231"
      },
      "created": "2023-09-28T19:28:12.825Z",
      "updated": "2023-09-28T19:28:12.825Z"
    }
  ]
}
```

3.5.2.7 Cadastrar tópico

Esta requisição tem a funcionalidade de cadastrar um tópico para uma organização. Vale salientar que o atributo *title* deve ser único, sendo necessária a verificação.

POST /api/v1/sac/topics

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	CREATE_TOPIC_ANY	Criar um tópico em qualquer organização
CUSTOMERSERVICE	CREATE_TOPIC_ORG	Criar um tópico na própria organização

Tabela 10: Permissões ACL para criar tópicos

Atributos de entrada

- **org**: inteiro, obrigatório, não nulo.
- **title**: string, obrigatório, não nulo, não vazio, máx.: 255.
- **descr**: string, opcional, máx.: 1024.
- **contactEmail**: string, opcional, máx.: 255.
- **contactPhones**: array(string), opcional.
- **imageURL**: string, opcional, máx.: 1024.
- **active**: boolean, opcional.
- **config**: objeto, opcional.
- **schedule**: array (objeto), obrigatório
  - *start*: string, obrigatório, formato: hh-mm.
  - *finish*: string, obrigatório, formato: hh-mm.
  - *day*: string, obrigatório, enum: {'MO', 'TU', 'WE', 'TH', 'FR', 'ST', 'SU'}.
- **attendants**: array (inteiros), obrigatório.

Status de resposta esperados

- **201 CREATED** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*org*)
- **409 CONFLICT** – Recurso com o mesmo identificador (*title*)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "id": 1
}
```

3.5.2.8 Listar tópicos

Esta requisição tem a finalidade de listar os tópicos do SAC, com a possibilidade de realizar filtragens, controle da quantidade de itens retornados, preparados para paginação e ordenação. Apenas tópicos aos quais o usuário tem permissão devem ser retornados, excluindo os logicamente deletados (*enabled* = *true*).

POST /api/v1/sac/topics/list

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	READ_TOPIC_ANY	Listar qualquer tópico
CUSTOMERSERVICE	READ_TOPIC_ORG	Listar tópicos apenas da organização que faz parte
CUSTOMERSERVICE	READ_TOPIC_OWN	Listar tópicos nos quais é atendente habilitado

Tabela 11: Permissões ACL para listar tópicos

Atributos de entrada

- **dataControl**: objeto, opcional
  - *limit*: number, opcional
  - *offset*: number, opcional
  - *paging*: boolean, opcional
- **orderBy**: array(objeto), opcional
  - *order*: number, opcional
  - *attribute*: string, obrigatório, enum: {'ID', 'TITLE', 'SCHEDULE_TIME', 'SCHEDULE_DAY', 'N_SPECIALISTS', 'SUPPORT_EMAIL'}
  - *direction*: string, obrigatório, enum: {'ASC', 'DESC'}
- **filters**: objeto, opcional
  - *id*: number, opcional
  - *timestamp*: objeto, opcional
    - *begin*: string, opcional
    - *end*: string, opcional
  - *active*: boolean, opcional

Status de resposta esperados

- **200 OK** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "metadata": {
    "offset": 0,
    "limit": 10,
    "total": 100
  },
```

```
    "records": [
      {
        "id": 1,
        "org": 1,
        "title": "TÍTULO",
        "descr": "DESCRIÇÃO",
        "contactEmail": "email@email.com",
        "contactPhones": ["1134121723612"],
        "imageURL": "https://picsum.photos/200/300",
        "config": {},
        "created": "2023-09-28T19:28:12.825Z",
        "updated": "2023-09-28T19:28:12.825Z",
        "specialistsCount": 10,
        "specialists": [
          {
            "id": 1,
            "fullname": "NOME",
            "avatar": "https://pathtoimage.com.br/sabsa.png"
          }
        ],
        "schedule": [
          {
            "day": "MO",
            "start": "07:00",
            "finish": "12:00"
          }
        ]
      }
    ],
    "overview": {
      "active": 10,
      "inactive": 5
    }
  }
```


3.5.2.9 Listar tópicos públicos

Esta requisição tem a finalidade de listar os tópicos do SAC, com a possibilidade de realizar filtragens, controle da quantidade de itens retornados, preparados para paginação e ordenação. Apenas tópicos que o usuário tem permissão devem ser retornados, assim como a exclusão dos deletados logicamente (*enabled* = *true*).

GET /api/v1/sac/public/topics

Authentication: APIKEY {apiKey}

Status de resposta esperados

- **200 OK** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
[
  {
    "title": "TÍTULO",
    "descr": "DESCRIÇÃO",
    "contactEmail": "email@email.com",
    "contactPhones": ["1134121723612"],
    "imageURL": "https://picsum.photos/200/300",
    "created": "2023-09-28T19:28:12.825Z",
    "updated": "2023-09-28T19:28:12.825Z",
    "schedule": [
      {
        "day": "MO",
        "start": "07:00",
        "finish": "12:00"
      }
    ]
  }
]
```

3.5.2.10 Recuperar tópico

Esta requisição tem a finalidade de recuperar em detalhes um tópico de forma individual. Apenas tópicos ativos devem ser recuperados (*enabled* = *true*).

GET /api/v1/sac/topics/{topic}

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	READ_TOPIC_ANY	Recuperar qualquer tópico
CUSTOMERSERVICE	READ_TOPIC_ORG	Recuperar tópicos apenas da organização que faz parte
CUSTOMERSERVICE	READ_TOPIC_OWN	Recuperar tópicos nos quais é atendente habilitado

Tabela 12: Permissões ACL para recuperação de tópicos

Atributos de URL

- **topic**: string, obrigatório, não nulo, não vazio — Identificador do tópico.

Status de resposta esperados

- **200 OK** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*topic*)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "id": 1,
  "org": 1,
  "title": "TÍTULO",
  "descr": "DESCRIÇÃO",
  "contactEmail": "email@email.com",
  "contactPhones": ["11341217236l2"],
  "imageURL": "https://picsum.photos/200/300",
  "config": {},
  "specialists": [
    {
      "id": 1,
      "fullname": "NOME",
      "avatar": "https://pathtoimage.com.br/sabsa.png"
    }
  ],
  "schedule": [
    {
      "day": "MO",
      "start": "07:00",
      "finish": "12:00"
    }
  ],
  "created": "2023-09-28T19:28:12.825Z",
  "updated": "2023-09-28T19:28:12.825Z"
}
```

3.5.2.11 Atualizar tópico

Esta requisição tem a finalidade de atualizar os dados de um tópico. Apenas tópicos ativos podem ser atualizados (*enabled* = *true*). Para atualizar o *title*, deve ser verificado se já existe outro tópico com o mesmo título.

PATCH /api/v1/sac/topics/{topic}

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	UPDATE_TOPIC_ANY	Atualizar qualquer tópico
CUSTOMERSERVICE	UPDATE_TOPIC_ORG	Atualizar apenas tópicos da organização que faz parte

Tabela 13: Permissões ACL para atualização de tópicos

Atributos de URL

- **topic**: string, obrigatório, não nulo, não vazio — Identificador serial do tópico.

Atributos de entrada

- **title**: string, opcional, não nulo, não vazio, máx.: 255.
- **descr**: string, opcional, máx.: 1024.
- **contactEmail**: string, opcional, máx.: 255.
- **contactPhones**: array (string), opcional.
- **imageURL**: string, opcional, máx.: 1024.
- **active**: boolean, opcional.
- **config**: objeto, opcional.
- **schedule**: array(objeto), opcional
  - *start*: string, obrigatório, formato: hh-mm.
  - *finish*: string, obrigatório, formato: hh-mm.
  - *day*: string, obrigatório, enum: {'MO', 'TU', 'WE', 'TH', 'FR', 'ST', 'SU'}.
- **attendants**: array (integers), opcional.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*topic*)
- **409 CONFLICT** – Recurso com o mesmo identificador (*title*)

3.5.2.12 Deletar tópico

Esta requisição tem a finalidade de deletar um tópico do sistema. A deleção será lógica, ou seja, o atributo *enabled* deverá assumir o valor *false*.

DELETE /api/v1/sac/topics/{topic}

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	DELETE_TOPIC_ANY	Remover tópicos de qualquer organização
CUSTOMERSERVICE	DELETE_TOPIC_ORG	Remover tópicos da própria organização

Tabela 14: Permissões ACL para remoção de tópicos

Atributos de URL

- **topic**: string, obrigatório, não nulo, não vazio — Identificador serial do tópico.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (*topic*)

3.5.2.13 Requisitar atendimento

Esta requisição tem a funcionalidade de requisitar um atendimento. Esta rota é aberta ao público, já que o usuário requisitante não terá credenciais.

POST /api/v1/sac/requests

Authentication: APIKEY {apiKey}

Atributos de entrada

- **topic**: integer, obrigatório, não nulo.
- **parentRequest**: integer, opcional, nullable.
- **name**: string, obrigatório, não nulo, não vazio, máx.: 255.
- **email**: string, obrigatório, não nulo, não vazio, máx.: 255.
- **phone**: string, obrigatório, não nulo, não vazio, máx.: 16.
- **docType**: string, opcional, não nulo, não vazio, enum: {'CPF', 'RG', 'PASSPORT', 'CNH'}, *default*: 'CPF'.
- **docValue**: string, obrigatório, não nulo, não vazio, máx.: 16.
- **descr**: string, obrigatório, não nulo, não vazio, máx.: 1024.

Status de resposta esperados

- **201 CREATED** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (organização, serviço)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "id": 1,
  "locator": "ABJS21BD",
  "websocketToken": "SAJYHBDJSAHBUYBUYBSAUBD=="
}
```

3.5.2.14 Listar solicitações de atendimento

Esta rota da aplicação tem a funcionalidade de listar as solicitações de atendimento.

POST /api/v1/sac/requests/list

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	READ_REQUEST_ANY	Listar qualquer requisição
CUSTOMERSERVICE	READ_REQUEST_ORG	Listar requisições apenas da organização que faz parte
CUSTOMERSERVICE	READ_REQUEST_OWN	Listar requisições nas quais é atendente habilitado

Tabela 15: Permissões ACL para listar solicitações

Atributos de entrada

- **dataControl**: objeto, opcional
  - *limit*: number, opcional
  - *offset*: number, opcional
  - *paging*: boolean, opcional
- **orderBy**: array(objeto), opcional
  - *order*: number, opcional
  - *attribute*: string, obrigatório, enum: {'ID', 'LOCATOR', 'CREATED', 'USER_NAME', 'USER_EMAIL', 'TOPIC_NAME', 'STATUS'}
  - *direction*: string, obrigatório, enum: {'ASC', 'DESC'}
- **filters**: objeto, opcional
  - *id*: number, opcional
  - *topic*: number, opcional
  - *status*: string[], opcional
  - *name*: string, opcional
  - *docType*: string, opcional
  - *docValue*: string, opcional
  - *timestamp*: objeto, opcional

    - *begin*: string, opcional
    - *end*: string, opcional

Status de resposta esperados

- **200 OK** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "metadata": {
    "offset": 0,
    "limit": 10,
    "total": 100
  },
  "records": [
    {
      "id": 1,
      "locator": "SANL275",
      "status": "ARRIVED",
      "desc": "DESCRIPTION",
      "name": "NAME",
      "email": "name@email.com",
      "phone": "38999803456",
      "docType": "CPF",
      "docValue": "978453764212",
      "docVerified": false,
      "created": "2023-09-28T19:28:12.835Z",
      "updated": "2023-09-28T19:28:12.835Z",
      "topicId": 1,
      "topic": {
        "id": 1,
        "title": "TITLE",
        "desc": "DESCRIPTION",
        "contactEmail": "rmail@email.com",
        "contactPhone": "(113412732612)",
        "imageURL": "https://picsum.photos/200/300",
        "config": {
          "schedule": {
            "day": "MO",
            "start": "07:00",
            "finish": "12:00"
          }
        }
```

```
        },
        "conference": {
          "id": 1,
          "short": "siaah28h"
        },
        "specialist": {
          "id": 1,
          "fullname": "NAME",
          "email": "name@email.com",
          "phone": "38999873451",
          "avatar": "https://picsum.photos/200/300"
        }
      }
    ]
  }
```


3.5.2.15 Recuperar requisição

Esta requisição tem a finalidade de recuperar em detalhes uma requisição de forma individual.

GET /api/v1/sac/requests/{request}

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	READ_REQUEST_ANY	Recuperar qualquer requisição
CUSTOMERSERVICE	READ_REQUEST_ORG	Recuperar requisições apenas da organização que faz parte
CUSTOMERSERVICE	READ_REQUEST_OWN	Recuperar requisições nas quais é atendente habilitado

Tabela 16: Permissões ACL para recuperar requisições

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição.

Status de resposta esperados

- **200 OK** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "id": 1,
  "locator": "SAN127S",
  "status": "ARRIVED",
  "descr": "DESCRIPTION",
  "name": "NAME",
  "email": "name@email.com",
  "phone": "83988904356",
  "docType": "CPF",
  "docValue": "701873674212",
  "docVerified": false,
  "created": "2023-09-28T19:28:12.825Z",
  "updated": "2023-09-28T19:28:12.825Z",
  "topic": {
    "id": 1,
    "title": "TITLE",
    "descr": "DESCRIPTION",
    "contactEmail": "email@email.com",
    "contactPhone": "1134121723612",
    "imageURL": "https://picsum.photos/200/300",
    "schedule": [
      {
        "day": "MO",
        "start": "07:00",
        "finish": "12:00"
      }
    ],
    "created": "2023-09-28T19:28:12.825Z",
    "updated": "2023-09-28T19:28:12.825Z"
  },
  "conference": {
    "id": 1,
    "short": "siah218h"
  },
  "specialist": {
    "id": 1,
    "fullname": "NAME",
    "email": "name@email.com",
    "phone": "83990873451",
    "avatar": "https://picsum.photos/200/300"
  }
}
```

3.5.2.16 Atendente aceita requisição

Esta requisição tem a finalidade de aceitar uma requisição de atendimento. Deve ser verificado se o usuário logado tem permissão para executar o atendimento, se ele é um especialista daquele tópico e se a solicitação já não foi finalizada (*outcome* $\neq$ *null*), ou atribuída a outro especialista. O atendente se atribui como especialista, muda o status para *ATTENDING*, cria a conferência (se ainda não houver) e entra nela.

POST /api/v1/sac/requests/{request}/ACCEPT

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	JOIN_REQUEST_ORG	Entrar como atendente em qualquer tópico da organização
CUSTOMERSERVICE	JOIN_REQUEST_OWN	Entrar como atendente em tópicos para os quais é habilitado na organização

Tabela 17: Permissões ACL para entrar como atendente

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição.

Status de resposta esperados

- **200 OK** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "roomName": "string",
  "sessionName": "string",
  "sessionToken": "string"
}
```

3.5.2.17 Atendente finaliza atendimento

Esta requisição tem a finalidade de finalizar uma requisição de atendimento. Deve ser verificado se o usuário logado tem permissão para realizar o atendimento, ou seja, se ele é o especialista responsável pelo tópico.

POST /api/v1/sac/requests/{request}/CLOSE

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	CLOSE_REQUEST_ORG	Finalizar como atendente em qualquer tópico da organização
CUSTOMERSERVICE	CLOSE_REQUEST_OWN	Finalizar como atendente em tópicos para os quais é habilitado na organização

Tabela 18: Permissões ACL para finalizar atendimentos

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição.

Atributos de entrada

- **outcome**: string, obrigatório, enum: {'RESOLVED', 'UNRESOLVED', 'TRANSFERRED', 'INTERRUPTED'}.
- **result**: string, opcional.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

3.5.2.18 Solicitante entra na conferência

Esta requisição tem a finalidade de permitir que o solicitante entre na conferência após ser aceito pelo atendente. Verifica-se se o status é *ATTENDING* e se o atendimento não foi finalizado. Rota aberta.

POST /api/v1/sac/requests/{request}/JOIN

Authentication: APIKEY {apiKey}

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição (*locator*).

Status de resposta esperados

- **200 OK** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "roomName": "string",
  "sessionName": "string",
  "sessionToken": "string"
}
```

3.5.2.19 Solicitante sai da sala de espera

Esta requisição tem a finalidade de permitir que o solicitante saia da sala de espera. Rota aberta.

POST /api/v1/sac/requests/{request}/LEFT

Authentication: APIKEY {apiKey}

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição (*locator*).

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

3.5.2.20 Solicitante volta para fila

Esta requisição tem a finalidade de recolocar o solicitante novamente na fila de espera. Rota aberta.

POST /api/v1/sac/requests/{request}/QUEUE

Authentication: APIKEY {apiKey}

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição (*locator*).

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

3.5.2.21 Atendente transfere solicitação para outro tópico

Esta requisição tem a finalidade de um atendente transferir uma solicitação para outro tópico. Uma nova solicitação é criada, tendo como *parentRequest* a solicitação atual.

POST /api/v1/sac/requests/{request}/TRANSFER

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	CLOSE_REQUEST_ORG	Finalizar como atendente em qualquer tópico da organização
CUSTOMERSERVICE	CLOSE_REQUEST_OWN	Finalizar como atendente em tópicos para os quais é habilitado na organização

Tabela 19: Permissões ACL para transferir(e finalizar) atendimentos

Atributos de URL

- **request**: string, obrigatório, não nulo, não vazio — Identificador da requisição.

Atributos de entrada

- **topic**: integer, obrigatório.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

3.5.2.22 Criar ou atualizar configurações para SAC

Esta requisição tem a finalidade de definir as configurações de uma SAC. Após serem definidas, o botão de incorporar ficará habilitado no *frontend*. Nesse momento, deve ser criado um código (*sacCode*) do tipo UUID(*Universally Unique IDentifier*), de forma automática, que será utilizado como *APIKey*.

POST /api/v1/sac/{org}/CONFIG

Authentication: Bearer ACCESSTOKEN

Atributos de entrada

- **phones**: array¡string¿, obrigatório.
- **supportEmail**: string, obrigatório.
- **schedule**: array¡object¿, obrigatório

Recurso	Ação	Detalhes
CUSTOMERSERVICE	CREATE_CONFIG_ANY	Criar configurações de SAC em qualquer organização
CUSTOMERSERVICE	CREATE_CONFIG_ORG	Criar configurações de SAC na própria organização

Tabela 20: Permissões ACL para criação de configurações de SAC

- *start*: string, obrigatório, formato: hh-mm.
- *finish*: string, obrigatório, formato: hh-mm.
- *day*: string, obrigatório, enum: {'MO', 'TU', 'WE', 'TH', 'FR', 'ST', 'SU'}.

Status de resposta esperados

- **204 NO CONTENT** – Sucesso
- **400 BAD REQUEST** – Corpo da requisição incorreto
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

3.5.2.23 Recuperar configurações de SAC

Esta requisição tem a finalidade de recuperar as configurações de uma SAC. Caso não existam configurações para a organização, o botão de incorporar não deve ser habilitado no *frontend*.

GET /api/v1/sac/{org}/CONFIG

Authentication: Bearer ACCESSTOKEN

Recurso	Ação	Detalhes
CUSTOMERSERVICE	READ_CONFIG_ANY	Recuperar configurações de SAC de qualquer organização
CUSTOMERSERVICE	READ_CONFIG_ORG	Recuperar configurações de SAC apenas de uma organização que faz parte

Tabela 21: Permissões ACL para leitura de configurações de SAC

Atributos de URL

- **org**: string, obrigatório, não nulo, não vazio — Identificador da organização.

Status de resposta esperados

- **200 OK** – Sucesso
- **401 NOT AUTHORIZED** – Credenciais não fornecidas
- **403 FORBIDDEN** – Sem permissão
- **404 NOT FOUND** – Recurso não encontrado (requisição)

Formato da resposta esperado (em caso de sucesso)

```
Content-Type: application/json; charset=utf-8
{
  "enabled": "boolean",
  "config": {
    "apiKey": "string",
    "supportEmail": "string",
    "phones": "array",
    "schedule": [
      {
        "day": "MO",
        "start": "07:00",
        "finish": "12:00"
      }
    ]
  }
}
```

3.6 *Websocket*

Algumas das funcionalidades relacionadas ao atendimento de uma SAC impõem que exista um sincronismo entre quem solicita atendimento e quem recebe a solicitação, e, por esse motivo, um sistema de conexão *websocket* foi implementado no sistema.

A conexão *websocket* ocorre entre os usuários clientes e administradores, que, logados no sistema, também estão conectados ao servidor *websocket*. Determinadas interações dos usuários na plataforma disparam eventos que são distribuídos entre os atores relevantes a solicitação e desencadeam *feedbacks* visuais na interface e tratamento da informação no *backend*.

Esses eventos foram definidos e estão descritos na Tabela 22.

Tabela 22: Eventos emitidos durante o atendimento

Mnemônico	Descrição	Destinatário	Meio
SAC_REQUESTER_WAITING	Emitida quando um requisitante entra na sala de espera.	Atendentes do tópico	SOCKET
SAC_REQUESTER_LEFT	Emitida quando um requisitante sai manualmente da sala de espera ou do atendimento.	Atendentes do tópico	SOCKET

Mnemônico	Descrição	Destinatário	Meio
SAC_REQUESTER_DROP	Emitida quando um requisitante sai abruptamente.	Atendentes do tópico	SOCKET
SAC_REQUESTER_JOIN	Emitida quando um requisitante entra na conferência.	Atendentes do tópico	SOCKET
SAC_SPECIALIST_JOIN	Emitida quando o especialista aceita e entra num atendimento.	Requisitante	SOCKET
SAC_SPECIALIST_LEFT	Emitida quando um especialista sai manualmente do atendimento.	Requisitante	SOCKET
SAC_SPECIALIST_DROP	Emitida quando um especialista sai abruptamente.	Requisitante	SOCKET

3.6.1 *Diagrama de sequência*

Os endpoints e eventos WebSocket documentados anteriormente interagem de forma coordenada para atender aos requisitos da funcionalidade AtendeFácil. A figura 9 a seguir ilustra, por meio de um diagrama de sequência, o fluxo típico de uma solicitação de atendimento SAC, desde a criação até a finalização. Esse exemplo oferece uma visão mais clara das etapas envolvidas e da troca de informações entre o cliente, o atendente e o sistema, demonstrando como as diferentes partes da API colaboram para garantir o sucesso do atendimento.

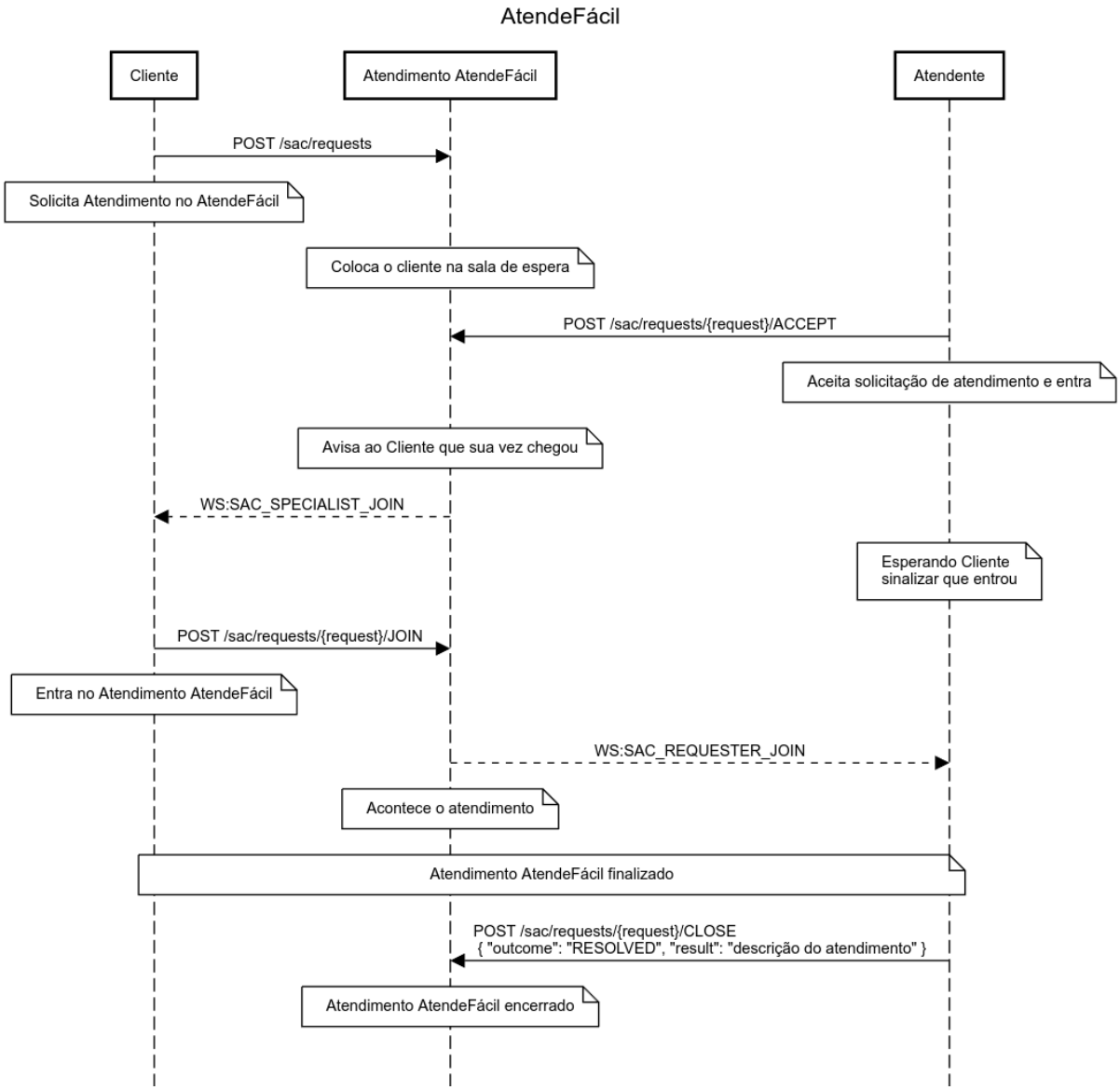

Figura 9: Diagrama de sequência de uma solicitação de atendimento SAC.
Fonte: própria do autor

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Este capítulo apresenta os resultados obtidos na implementação do *backend* da funcionalidade AtendeFácil na plataforma PBMeet, com foco nas principais funcionalidades, testadas manualmente. As validações foram feitas localmente, com o uso da documentação viva gerada pelo Swagger(Figura 10), que contém exemplos de requisições e respostas esperadas para cada caso. Esse processo foi fundamental para guiar o desenvolvimento e a integração com o *frontend*.

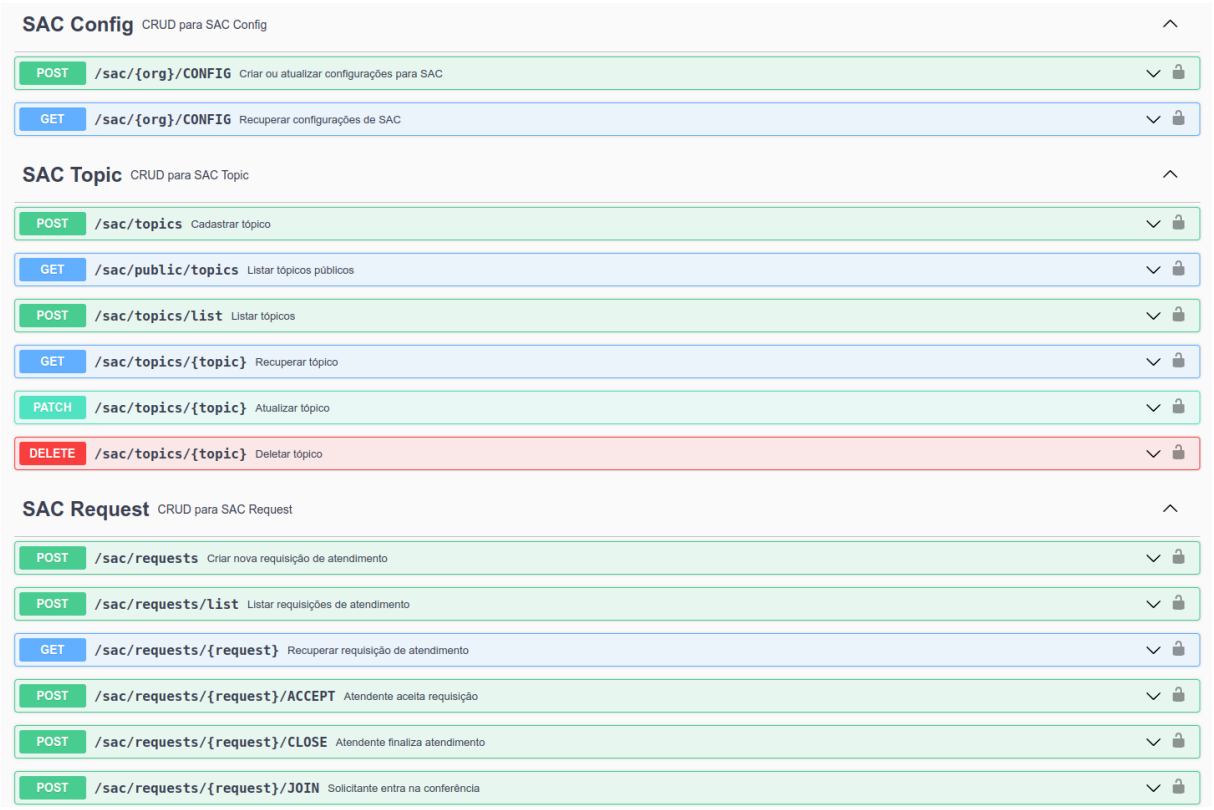

Figura 10: Swagger do AtendeFácil.

4.1 Plano de Testes Manuais

Os testes manuais focaram em verificar o comportamento esperado das funcionalidades desenvolvidas no *backend*, com ênfase em:

- **Funcionalidade de SAC:** Cobrir todos os aspectos do fluxo de atendimento ao cliente, desde a criação da solicitação, até a finalização do atendimento.
- **Persistência e Consistência de Dados:** Garantir que as operações de criação, atualização e exclusão de registros no banco de dados ocorressem conforme esperado.
- ***Websocket*:** Testar a comunicação em tempo real entre cliente e atendente, verificando o envio e recebimento de eventos durante o atendimento.

Todos os testes foram realizados diretamente via a interface gerada pelo Swagger, utilizando a API documentada para enviar requisições POST, PATCH, GET e DELETE, além da análise manual do comportamento das conexões *websocket*.

4.2 Casos de Teste Manuais

Nas Tabelas 23-29 são apresentados (Título, Descrição, Passos do Teste, Resultado Esperado e Resultado Obtido) os casos de teste manuais que foram realizados, categorizados pelos principais requisitos funcionais da aplicação.

4.2.1 Casos de Teste para as Rotas HTTP

Título	[TM01] Criar solicitação de atendimento
Descrição	O teste verifica a criação de uma solicitação de atendimento, inserindo os dados do cliente e do tópico de atendimento.
Passos do Teste	Requisição POST para a rota '/sac/request' com nome, e-mail, telefone e descrição do problema do cliente.
Resultado Esperado	A solicitação é salva no banco de dados com sucesso e o cliente recebe um identificador de solicitação (*locator*) para acompanhamento.
Resultado Obtido	Sucesso.

Tabela 23: Caso de Teste [TM01].

Título	[TM02] Listar solicitações de atendimento
Descrição	Verifica se a listagem de solicitações funciona corretamente, exibindo as solicitações pendentes de atendimento.
Passos do Teste	Requisição GET para a rota '/sac/request/list', que retorna todas as solicitações de atendimento pendentes para um determinado atendente.
Resultado Esperado	A resposta exibe uma lista de solicitações com dados corretos, como nome do cliente, descrição e status de espera.
Resultado Obtido	Sucesso.

Tabela 24: Caso de Teste [TM02].

Título	[TM03] Finalizar atendimento
Descrição	Testa a finalização de um atendimento em andamento, verificando se o status da solicitação é atualizado corretamente.
Passos do Teste	Requisição POST para a rota '/sac/request/id/finish' com o status do atendimento (RESOLVED ou UNRESOLVED) e o motivo da finalização.
Resultado Esperado	A solicitação é marcada como finalizada no banco de dados, e o status é atualizado para resolvido ou não resolvido.
Resultado Obtido	Sucesso.

Tabela 25: Caso de Teste [TM03].

Título	[TM04] Transferir atendimento
Descrição	Testa a transferência de uma solicitação para outro tópico ou atendente.
Passos do Teste	O atendente transfere a solicitação via POST para a rota '/sac/request/id/transfer' indicando o novo tópico.
Resultado Esperado	A solicitação é associada ao novo tópico, e o cliente é notificado da transferência.
Resultado Obtido	Sucesso.

Tabela 26: Caso de Teste [TM04].

4.2.2 Casos de Teste para *websocket*

Os testes de *websocket* foram realizados para verificar o comportamento da comunicação em tempo real entre cliente e atendente. O *websocket* é responsável por notificar os atendentes e clientes sobre eventos, como o início de um atendimento, ou a chegada de uma nova solicitação.

Título	[TM05] Notificação de nova solicitação
Descrição	Verifica se, ao ser criada uma nova solicitação de atendimento, os atendentes conectados recebem uma notificação em tempo real.
Passos do Teste	Criar uma solicitação de atendimento via POST e observar a notificação enviada via *websocket* para os atendentes conectados.
Resultado Esperado	A notificação de nova solicitação é recebida instantaneamente pelos atendentes.
Resultado Obtido	Sucesso.

Tabela 27: Caso de Teste [TM05].

Título	[TM06] Notificação de cliente entrando na conferência
Descrição	Testa a notificação em tempo real para o atendente quando o cliente entra na videoconferência.
Passos do Teste	Após a criação de uma solicitação e aceitação pelo atendente, o cliente entra na videoconferência e o *websocket* dispara a notificação.
Resultado Esperado	O atendente recebe uma notificação em tempo real de que o cliente entrou na conferência.
Resultado Obtido	Sucesso.

Tabela 28: Caso de Teste [TM06].

4.2.3 Casos de Teste para Gerenciamento de Arquivos

Título	[TM07] Enviar arquivo durante atendimento
Descrição	Verifica se é possível enviar um arquivo de apoio durante a videoconferência do atendimento.
Passos do Teste	O atendente usa a rota POST '/conference/id/files' para enviar um arquivo PDF de apoio durante o atendimento.
Resultado Esperado	O arquivo é armazenado no banco de dados e fica disponível para o cliente.
Resultado Obtido	Sucesso.

Tabela 29: Caso de Teste [TM07].

4.3 Análise dos Resultados

Os resultados dos testes manuais indicaram que todas as funcionalidades principais do AtendeFácil no *backend* funcionaram corretamente. As rotas de criação, listagem e finalização de solicitações, bem como as operações com arquivos de apoio, foram validadas com sucesso. Além disso, os testes de comunicação via *websocket* confirmaram o funcionamento das notificações em tempo real entre cliente e atendente.

A utilização da documentação Swagger foi fundamental para guiar os testes, oferecendo exemplos claros de requisições e respostas esperadas, o que facilitou a validação das funcionalidades. Embora a abordagem manual tenha suas limitações, em termos de cobertura, foi possível identificar e corrigir problemas durante os testes, garantindo que o sistema estivesse pronto para produção.

5 CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento da funcionalidade AtendeFácil no *backend* da plataforma PBMeet foi bem-sucedido ao agregar uma solução de SAC, via videoconferência. A implementação seguiu boas práticas de engenharia de software, resultando em uma API REST bem estruturada e documentada, além de um sistema de comunicação em tempo real eficiente com *websocket*. Isso possibilitou a criação de um fluxo de atendimento direto e prático entre clientes e atendentes, melhorando a eficiência do serviço oferecido.

A funcionalidade AtendeFácil trouxe as seguintes contribuições ao PBMeet:

- Expansão do escopo da plataforma, permitindo seu uso não apenas para reuniões e conferências, mas também como uma solução eficaz de SAC por videoconferência.
- Implementação de rotas REST que garantem o gerenciamento eficiente das solicitações de atendimento, com operações claras de criação, acompanhamento e finalização de atendimentos.
- Uso de *websocket* para garantir comunicação em tempo real entre clientes e atendentes, permitindo notificações rápidas e imediatas, o que melhora a experiência do usuário.
- Extensão do modelo de dados do PBMeet para gerenciar as novas entidades relacionadas ao atendimento, como solicitações, atendentes e tópicos de suporte, garantindo uma base sólida para as operações do SAC.

A qualidade da solução *backend* é refletida na organização do código, no uso de *controllers* para gerenciar as requisições e na validação de dados para garantir a integridade das operações. Essas boas práticas foram fundamentais para assegurar o bom funcionamento do sistema, entregando uma funcionalidade que atende às expectativas e oferece um serviço consistente.

5.1 Limitações

Embora o AtendeFácil tenha atendido aos principais objetivos de desenvolvimento, algumas limitações foram observadas no que diz respeito à funcionalidade e implementação da solução:

- **Cobertura de Testes de Exceção e Falta de Testes Automáticos**: A maior parte dos testes realizados cobre o fluxo principal da aplicação, focando no "caminho feliz". A inclusão de testes automáticos é necessária para assegurar uma verificação contínua da integridade do sistema. Além disso, testes de exceção são importantes para verificar a resposta do sistema a cenários de falha e condições extremas, como entradas incorretas ou situações inesperadas nos endpoints.

- **Interface para Monitoramento e Análise**: Atualmente, o AtendeFácil não oferece uma interface dedicada para monitoramento de métricas de atendimento, como tempo médio de resolução, número de atendimentos por atendente ou índice de satisfação. Isso limita a capacidade de análise detalhada e o acompanhamento da eficiência operacional, impactando o planejamento de melhorias contínuas.

- **Feedback do Atendimento**: Embora o sistema permita a finalização e categorização dos atendimentos, ele não oferece uma funcionalidade de feedback direto dos clientes. A ausência de uma avaliação sobre o atendimento limita a identificação de oportunidades para aprimoramento e a análise qualitativa do desempenho dos atendentes.

5.2 Trabalhos Futuros

Embora o *backend* da funcionalidade AtendeFácil tenha atingido seus principais objetivos, há oportunidades de expandir as funcionalidades e aprimorar a qualidade do serviço. Algumas sugestões para trabalhos futuros incluem:

- Adicionar **testes automáticos** aos *endpoints* do AtendeFácil, garantindo a verificação contínua da funcionalidade e da integridade do sistema com testes de unidade e integração. Esses testes ajudariam a identificar problemas antes de novas versões serem lançadas, aumentando a confiabilidade do sistema.

- Realizar e documentar **testes de exceção**, que verificam cenários de falha e tratam possíveis erros além do "caminho feliz". Esse tipo de teste ajudaria a robustecer o sistema ao antecipar comportamentos inesperados, contribuindo para a estabilidade em produção.

- Desenvolver um **sistema de métricas** para acompanhar o desempenho dos atendimentos, como o tempo médio de resolução e taxa de problemas solucionados, visando a melhoria contínua da qualidade do atendimento.

- Implementar um ***dashboard* de histórico de atendimentos**, oferecendo uma visão detalhada de métricas como tempo de resolução, número de atendimentos finalizados e qualidade do serviço. Esse painel ajudaria a gerar relatórios gerenciais para acompanhar a eficiência do SAC.

- Criar um sistema de **avaliação do atendimento**, permitindo que os clientes forneçam *feedback* sobre o atendimento recebido. Esse sistema incluiria critérios como cordialidade, tempo de resposta e resolução do problema, ajudando a identificar oportunidades de melhoria no serviço.

REFERÊNCIAS

[1] AMAZON WEB SERVICES. **The Difference Between Frontend and Backend**. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-frontend-and-backend/>. Acesso em: 23 out. 2024.

[2] ORACLE. **What is a Database?**. Disponível em: <https://www.oracle.com/br/database/what-is-database/>. Acesso em: 23 out. 2024.

[3] RED HAT. **What is a REST API?**. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>. Acesso em: 23 out. 2024.

[4] DEVNIT. **Desmistificando a Utilização de Mappers e DTOs em Cada Camada da Aplicação**. Disponível em: https://devnit.medium.com/desmistificando-a-utiliza%C3%A7%C3%A3o-de-mappers-e-dtos-em-cada-camada-da-aplica%C3%A7%C3%A3o-2aa7331fffc6. Acesso em: 23 out. 2024.

[5] NIZZOLA, M. **Entendendo Autenticação e Autorização**. Disponível em: https://marcionizzola.medium.com/entendendo-autentica%C3%A7%C3%A3o-e-autoriza%C3%A7%C3%A3o-76744ec71bf. Acesso em: 23 out. 2024.

[6] LE WAGON. **O Que é Padrão MVC?**. Disponível em: https://blog.lewagon.com/pt-br/skills/o-que-e-padrao-mvc/. Acesso em: 23 out. 2024.

[7] SWAGGER. **OpenAPI Specification**. Disponível em: https://swagger.io/specification/. Acesso em: 23 out. 2024.

[8] DEVMEDIA. **ORM - Object Relational Mapper**. Disponível em: https://www.devmedia.com.br/orm-object-relational-mapper/19056. Acesso em: 23 out. 2024.

[9] W2WEBSITES. **O Que é Repository Pattern?**. Disponível em: https://w2websites.com/glossario/o-que-e-repository-pattern/. Acesso em: 23 out. 2024.

[10] CASA DO DESENVOLVEDOR. **O Que São Migrations e Como Usá-las em Seu Projeto**. Disponível em: https://blog.casadodesenvolvedor.com.br/migrations/. Acesso em: 23 out. 2024.

[11] WIKIPEDIA. **Inversão de Controle**. Disponível em: https://pt.wikipedia.org/wiki/Invers%C3%A3o_de_controle. Acesso em: 23 out. 2024.

[12] ATLASSIAN. **Microservices Architecture**. Disponível em: https://www.atlassian.com/br/microservices/microservices-architecture. Acesso em: 23 out. 2024.

[13] ZUP. **Clean Architecture: Entenda a Arquitetura Limpa e Seus Benefícios**. Disponível em: https://zup.com.br/blog/clean-architecture-arquitetura-limpa. Acesso em: 23 out. 2024.