PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Arquitetura de Software Distribuído

Vânio Vieira
Vanilson Donizete Vieira

PROJETO DE TRANSFORMAÇÃO DIGITAL

Belo Horizonte
2021

Vânio Vieira
Vanilson Donizete Vieira

PROJETO DE TRANSFORMAÇÃO DIGITAL

Trabalho de Conclusão de Curso de Especialização em Arquitetura de Software Distribuído como requisito parcial à obtenção do título de especialista.

Orientador: Prof. Dr. Pedro A. Oliveira

Belo Horizonte
2021

À família pela compreensão nas horas de ausência e pelos constantes incentivos durante toda a jornada.

AGRADECIMENTOS

Aos nossos pais, que nunca mediram esforços para nos proporcionar acesso ao ambiente escolar e ao conhecimento.

RESUMO

Este documento descreve uma proposta de solução de software distribuído baseada em uma arquitetura de microsserviços para uma empresa do ramo têxtil. O projeto de software tem como objetivo promover a transformação digital definida pela mudança da estratégia organizacional. Esta mudança estratégica demandou investir em uma evolução disruptiva como resposta ao movimento provocado pela onda da indústria 4.0, que levou ao aumento da competitividade dos mercados local e global.

A solução incorpora requisitos arquiteturais que definem sua natureza distribuída, podendo ser acessada de qualquer lugar através de dispositivos com navegadores de internet. Além disso, a arquitetura é baseada em microsserviços que habilitam o escalonamento horizontal de todos os componentes novos, permitindo lidar de forma flexível com diferentes cargas de trabalho, e possibilitando a evolução em escala conforme as necessidades da organização.

A execução do projeto habilitará a empresa para uma ampla utilização de avançados recursos tecnológicos e quantidades massivas de dados, possibilitando a aplicação efetiva da estratégia para alavancar as vendas, manter a competitividade e o ritmo de constante crescimento esperados.

Palavras-chave: microsserviços, arquitetura de software, software distribuído, transformação digital, projeto de software.

SUMÁRIO

1. Objetivos do trabalho

O objetivo geral deste trabalho é apresentar o projeto de arquitetura de software para o Sistema Integrado de Gestão e Operação (SIGO), utilizando tecnologias livres e modernas, e aplicando recursos tecnológicos avançados para proporcionar vantagem competitiva, face a mudança estratégica definida. A arquitetura da solução, detalhada neste documento, visa possibilitar a manutenção dos ativos organizacionais, provendo alto grau de integração entre os diversos sistemas utilizados pela organização, que incluem os sistemas legados preexistentes, bem como as novas soluções que serão desenvolvidas e/ou adquiridas de mercado.

Os objetivos específicos são:

- Construir o módulo de autenticação e autorização de usuários, um gerenciador de usuários e credenciais de acesso para os módulos do sistema.
- Construir o módulo de Gestão de Normas, um repositório centralizado para as normas técnicas e regulamentações relacionadas ao processo industrial que também deverá acessar bases de dados externas para verificar as atualizações de normas.
- Construir o módulo de Assessorias e Consultorias, um gerenciador de contratos assessorias e consultorias e das ações resultantes.
- Construir o módulo de Gestão do Processo Industrial, um centro de operações que proverá meios de integração com os outros módulos, com os sistemas existentes e sistemas externos.

2. Descrição geral da solução

A solução SIGO fornece um conjunto de capacidades para que a organização possa utilizar amplamente seus recursos tecnológicos, possibilitando que seus sistemas legados estejam interconectados com os novos componentes e suportando sua reformulação de gestão, fornecendo acesso para participação dos colaboradores nos processos.

2.1. Apresentação do problema

O crescimento da digitalização teve um grande impacto no cenário empresarial global. A Indústria 4.0 é um movimento em direção aos métodos de produção de próxima geração que continuam a levar a uma maior globalização. A “quarta revolução industrial” reflete um desenvolvimento geral na fabricação e define a próxima fase de digitalização, impulsionada

pelo aumento dos volumes de dados, do poder computacional e conectividade, do surgimento de recursos analíticos e de inteligência de negócios, de novas formas de interação homem-máquina e melhorias na transferência de instruções digitais para o mundo físico, como a robótica avançada e a impressão em 3D.

A Indústria Têxtil do Brasil Ltda. (IndTexBr) é uma organização de abrangência nacional, que fabrica e comercializa tecidos no atacado. Dificuldades financeiras recentes, originadas com a retração do mercado local e a entrada de novos concorrentes chineses em sua área de negócio, levaram a empresa a investir em uma estratégia de evolução disruptiva. A IndTexBr pretende tornar-se uma exportadora do ramo têxtil, e por isso está realizando um grande projeto de mudança estratégica, que busca através da redução da dependência de fornecedores internos, da implantação de tecnologias mais avançadas de produção, de automatização, de reformulação no modelo de gestão, da capacitação de colaboradores e de investimentos em inovação, aumentar seu lucro em 5% e economizar de 10 a 20% de seu custo de produção.

Neste contexto, torna-se necessária uma ação planejada de transformação digital, fortemente baseada na utilização de recursos tecnológicos e de dados massivos, na integração entre os diversos sistemas de informação utilizados, nas pessoas e nos processos de negócio, que permitirá alavancar suas vendas e melhorar as perspectivas a médio e longo prazos.

2.2. Descrição geral do software (Escopo)

O projeto SIGO tem como objetivo estabelecer uma plataforma de sistemas baseada numa arquitetura de microsserviços, compatível com aplicações distribuídas, que possibilite a integração entre os sistemas existentes atualmente na empresa, os sistemas e módulos que serão construídos, e os sistemas e módulos que poderão ser adquiridos de mercado para o pleno atendimento aos requisitos de negócio demandados pela mudança estratégica.

O sistema a ser desenvolvido terá suas funcionalidades divididas em módulos independentes e integrados, que poderão ser implantados de acordo com a prioridade e necessidade da empresa. Por motivos econômicos e de agilidade, os sistemas já existentes deverão ser mantidos e reaproveitados, e os dados produzidos deverão ser disponibilizados na plataforma por meio de integrações. Por sua característica modular e sua compatibilidade com

aplicações distribuídas, o sistema poderá ser implantado em nuvem híbrida, e permitirá que os sistemas legados preexistentes sejam mantidos *on Premise*.

O SIGO deverá suportar ambientes *web* e *mobile*, ser de fácil utilização, seguro, ter alta disponibilidade e disponibilizar meios de integração para os diferentes sistemas, tanto de terceiros quanto desenvolvidos internamente.

Resumidamente, trata-se de uma interface de usuário (*frontend web)* que se comunica com uma camada de microsserviços que será responsável pelos dados e pelas regras de negócio de cada área atendida especificamente. A arquitetura da aplicação também prevê a integração com serviços externos. A integração com o serviço de normas é realizada através do consumo da API fornecida no formato SaaS, e a conexão é feita através do microsserviço de Gestão de Normas.

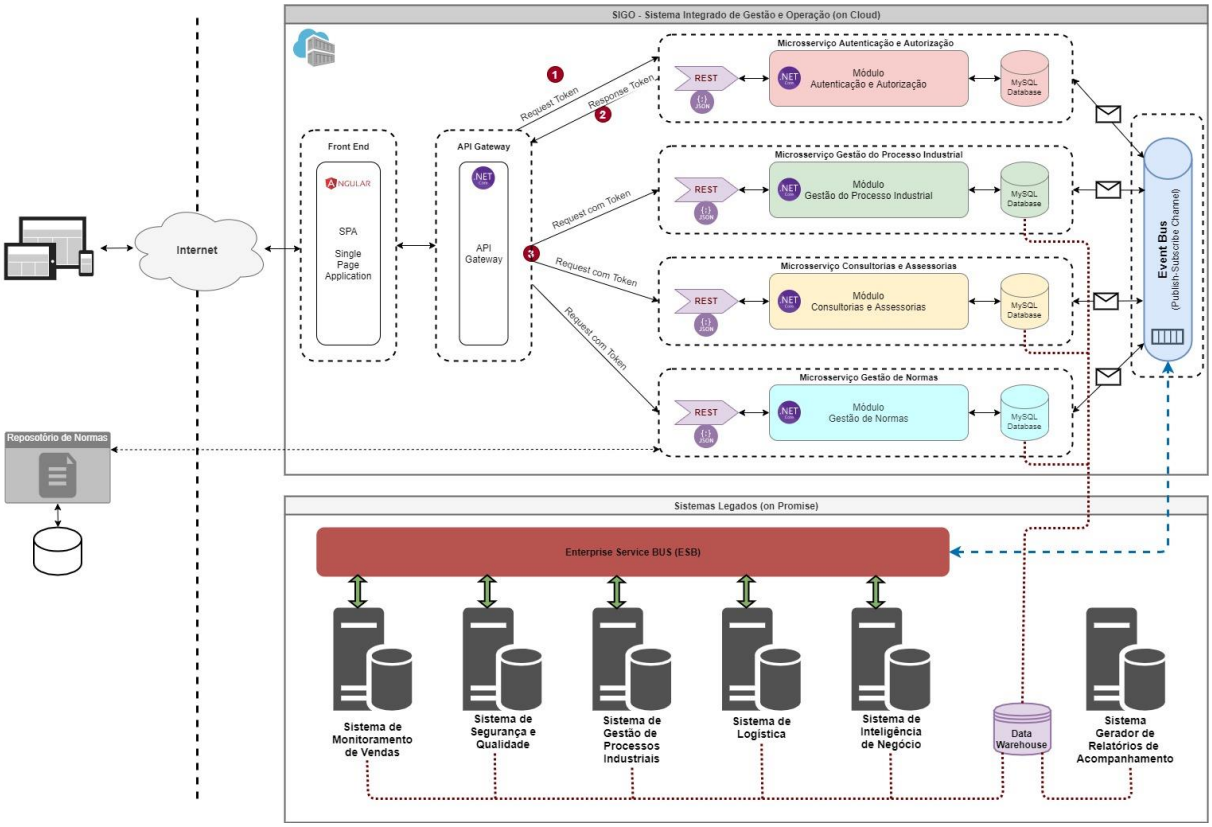

Figura 1 - Diagrama Informal da Arquitetura.

Adicionalmente, outros componentes arquiteturais poderão estar presentes para sanar dificuldades impostas pela arquitetura de microsserviços.

3. Definição conceitual da solução

Esta seção apresenta uma especificação do sistema por meio dos requisitos funcionais e não funcionais, das restrições e dos mecanismos arquiteturais considerados.

3.1. Requisitos Funcionais

Constituem módulos funcionais básicos da arquitetura a ser projetada para o SIGO:

1. Módulo Autenticação:
    - O módulo deve permitir o cadastro de novos usuários.
    - O módulo deve realizar a autenticação de um usuário que efetue login.
    - O módulo deve permitir que um usuário efetue o logout.
2. Módulo Gestão de Normas:
    - O módulo deve disponibilizar um repositório de normas técnicas (documentos em formato digital).
    - O módulo deve disponibilizar o repositório de normas em tecnologias de nuvem.
    - O módulo deve se conectar a um *webservice* que fará acesso às bases de dados externas de normas buscando identificar e notificar sobre possíveis mudanças e evoluções.
    - O módulo deve possibilitar o planejamento de ações que envolvam a aplicação das normas.
3. Módulos Consultorias e Assessorias:
    - O módulo deve possibilitar a gestão contratual de empresas de assessoria e consultoria.
    - O módulo deve ser integrado aos demais módulos do sistema.
    - O módulo deve possibilitar o planejamento e controle de atividades industriais.
4. Módulo Gestão do Processo Industrial:
    - O módulo deve realizar integração através de troca de mensagens com os demais módulos do sistema e com a solução ERP implantada.

Conforme especifica o item 4, além dos módulos funcionais básicos, o SIGO deve possibilitar a integração com os sistemas já existentes na organização. Estes sistemas têm como característica principal o uso de plataformas legadas, com destaque para o protocolo SOAP, as

linguagens Java e ABAP, o SGBD Oracle e middlewares Orientados a Mensagem por meio de comunicações síncronas ou assíncronas.

Nesse contexto destacam-se as seguintes aplicações legadas:

1. Sistema de Logística:
    - Possibilita gerenciar os recursos logísticos nas operações de compra e venda.
    - Possibilita gerenciar a utilização dos insumos pela atividade.
    - Está totalmente integrado aos sistemas de Compras e de Vendas.
2. Sistema de Gestão de Processos Industriais:
    - Possibilita gerenciar os processos envolvidos com a atividade têxtil, de forma coordenada, eficiente e ambientalmente responsável.
    - Possibilita controlar todo o workflow de fabricação, tanto no nível operacional como tático.
    - Possibilita realizar o registro diário e por turno de trabalho das paradas e problemas na produção.
    - Possibilita gerar uma lista de atividades diárias programadas para cada setor industrial e enviar diretamente aos setores ligados à produção, visando fabricar e separar a produção de tecidos, para posterior destinação por meio do Sistema de Logística.
    - Utiliza uma ferramenta de modelagem de processos (BPM).
3. Sistema de Segurança e Qualidade:
    - Subsistema responsável por garantir a segurança dos processos industriais, bem como a qualidade tanto dos produtos parciais quanto do produto acabado (final).
    - Atualmente não possui acesso via integração ao sistema de Gestão de Normas, de forma que não é possível assegurar que a norma mais atual e adequada está sendo seguida.
    - Sua integração e utilização são complexas e inadequadas, além de não possuir uma padronização com o restante dos subsistemas implantados.
4. Sistema de Monitoramento de Vendas:
    - Possibilita monitorar as vendas da empresa.
    - Acessado pelos gestores e outros colaboradores envolvidos no processo de vendas.
    - Sistema web/mobile multiplataforma acessível a smartphones nas plataformas Android e iOS, desenvolvido internamente com o uso de *React* e *React Native*.

5. Sistema de Inteligência do Negócio:
    - Solução de *Business Intelligence* (BI) adquirida da Oracle, mas ainda não foi implantada devido a incompatibilidades com a plataforma da SAP.
    - Possibilita gerar informações sobre insumos, produção, eventos anormais etc., e repassá-los para o sistema de monitoramento.
    - Utiliza como insumo as informações dos outros subsistemas/módulos, realizando tratamento dos dados de *big data* e de BI a fim de permitir tomadas de decisão mais assertivas.
6. Sistema Gerador de Relatórios de Acompanhamento:
    - Ferramenta externa (*Crystal Reports)* que gera relatórios e consultas sob demanda.
    - Possibilita acompanhar a real situação das atividades da empresa em tempo real, nas diversas áreas envolvidas.
    - Permite aos usuários gerarem as saídas que desejam, de acordo com seu perfil e necessidades de informação.

3.2 Requisitos Não Funcionais

A seguir são apresentados os requisitos não funcionais (RNF) do sistema:

- **Acessibilidade** – O sistema deve suportar ambientes web e móveis.

Estímulo	Usuário visualizando a listagem de normas técnicas cadastradas.
Fonte do Estímulo	Usuário acessando a funcionalidade do sistema através de um dispositivo móvel.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão de Normas.
Resposta	A interface se adapta a tela do dispositivo: a camada de apresentação se adapta as resoluções e tamanho das telas, mudando os componentes de posição de forma a ficar melhor a navegação do usuário, possibilitando a utilização em microcomputadores e dispositivos móveis através de navegador de internet.
Medida da Resposta	O usuário consegue visualizar as informações com experiência completa, onde a identidade visual é mantida nas diferentes resoluções de tela.

- **Usabilidade** – O sistema deve prover boa usabilidade.

Estímulo	Usuário visualizando o cadastro de uma norma técnica.
Fonte do Estímulo	Usuário acessando a funcionalidade e visualiza uma norma a partir da listagem de normas.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão de Normas.
Resposta	A interface de usuário apresenta facilidade de navegação, simplicidade e objetividade.
Medida da resposta	O usuário consegue realizar a operação sem a necessidade de auxílio de outra pessoa.

- **Desempenho** – O sistema deve possuir bom desempenho.

Estímulo	Usuário visualizando a listagem de consultorias cadastradas.
Fonte do Estímulo	Usuário acessando a funcionalidade do sistema através de um dispositivo desktop.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão de Assessorias e Consultorias.
Resposta	O sistema exibe a lista de consultorias em poucos segundos.
Medida da resposta	O sistema exibe a listagem de consultorias em menos de 5 segundos.

- **Manutenibilidade** – O sistema deve ser de fácil manutenção.

Estímulo	Analista de Garantia da Qualidade encontra um defeito em um módulo do sistema.
Fonte do Estímulo	O Analista de Garantia da Qualidade reporta a ausência de um elemento de tela necessário no cadastro de Normas.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão de Normas.
Resposta	Após o envio do código de correção para o repositório, é gerada uma nova versão do módulo para verificação e validação.
Medida da resposta	A correção é aplicada e novos testes são executados, garantindo uma nova versão pronta para a implantação sem afetar o funcionamento de outros módulos.

- **Testabilidade** – O sistema deve ser passível de ser testado em todas as suas funcionalidades.

Estímulo	Analista de Garantia da Qualidade executando testes manuais.
Fonte do Estímulo	Analista de Garantia da Qualidade testando o módulo de Assessorias e Consultorias.
Ambiente	Homologação, carga normal.
Artefato	Módulo de Assessorias e Consultorias.
Resposta	Os testes unitários das funcionalidades são realizados com resposta visual dos resultados de cada teste.
Medida da resposta	O sistema possibilitou realizar os testes através scripts automatizados acionado por apenas um comando.

- **Confiabilidade** – O sistema deve ser confiável e robusto, se recuperando no caso da ocorrência de erro.

Estímulo	Falha em *container* de microsserviço.
Fonte do Estímulo	*Container* do Módulo de Normas apresenta falha e é desativado.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão de Normas.
Resposta	Outro *container* do Módulo de Normas com a mesma configuração é inicializado de forma automática.
Medida da resposta	O módulo de Gestão de Normas continua disponível.

- **Interoperabilidade** – O sistema deve se comunicar com sistemas externos via APIs *RESTful* de integração.

Estímulo	Consulta a um dos repositórios externos de normas.
Fonte do Estímulo	Usuário visualizando o cadastro de uma norma técnica, automaticamente o Módulo de Gestão de Normas faz requisição ao repositório externo para consultar a situação atual da norma.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão de Normas.

Resposta	O sistema externo responde a requisição com os dados da norma requisitada.
Medida da resposta	A comunicação com o sistema externo é realizada com sucesso.

- **Segurança** – O sistema deve apresentar segurança no acesso e manipulação de dados.

Estímulo	Tentativa de acesso a funcionalidade sem permissão.
Fonte do Estímulo	Usuário não autenticado tenta fazer acesso em qualquer um dos módulos do sistema.
Ambiente	Produção, carga normal.
Artefato	Qualquer dos módulos disponíveis.
Resposta	O usuário é redirecionado pelo sistema para a página de autenticação.
Medida da resposta	O sistema, tanto na interface de usuário quanto nas interfaces das APIs, não permite o acesso não autenticado e não autorizado.

- **Disponibilidade** – O sistema deve estar disponível 24 horas por dia, nos sete dias da semana, para as funcionalidades ligadas à produção.

Estímulo	Atualização de módulo.
Fonte do Estímulo	Uma atualização do código-fonte de módulo é gerada após liberação de nova versão.
Ambiente	Produção, carga normal.
Artefato	Módulo de Gestão do Processo Industrial.
Resposta	Todos os usuários continuam usando o sistema.
Medida da resposta	O acesso ao sistema as funcionalidades existentes continuam disponíveis. A funcionalidades atualizadas são aplicadas e estão disponíveis para uso.

- **CI/CD** – O sistema deve ser desenvolvido utilizando recursos de integração contínua.

Estímulo	Geração de versão.
Fonte do Estímulo	Uma alteração de código-fonte é aprovada.
Ambiente	Produção, carga normal.
Artefato	Qualquer dos módulos disponíveis.

Resposta	O código-fonte é atualizado.
Medida da resposta	O código-fonte é atualizado, as atualizações são incluídas no repositório e as alterações são aprovadas. Uma nova versão do sistema é gerada.

3.3. Restrições Arquiteturais

- O sistema deve possuir uma estrutura modular, devendo ser desenvolvido e ser implantável por módulos.
- O sistema deve persistir os dados por meio de tecnologia de SGBD relacional.
- O sistema deve ser baseado em tecnologias robustas, livres e gratuitas.
- O sistema deve utilizar componentes reutilizáveis.
- O sistema deve ser testado e validado utilizando estratégias de integração contínua.
- O sistema deve ser compilado e implantado utilizando estratégias de entrega contínua.
- O sistema deve apresentar características de aplicações distribuídas, tais como abertura, portabilidade e uso extensivo de recursos de rede.
- O sistema deve adotar uma arquitetura baseada em microsserviços;
- O sistema deve ser hospedado em nuvem híbrida, com parte dos componentes (sistemas legados) sendo mantidos *on Premise*.
- As interfaces de usuário devem apresentar adaptação visual de página a qualquer dispositivo em que sejam visualizadas, sem a necessidade do uso de versões específicas para os diversos tipos de dispositivos e tamanhos de telas.
- A interface de usuário (*frontend)* de acesso aos módulos do sistema deve ser desenvolvida no formato aplicação de página única (*Single Page Application* – SPA).

3.4. Mecanismos Arquiteturais

Mecanismo de Análise	Mecanismo de Design	Mecanismo de Implementação
Linguagem	Linguagem de programação	C#
Plataforma	Plataforma de desenvolvimento	.NET Core 3.1
Persistência	ORM	Entity Framework Core
Persistência	Banco de dados relacional	MySQL
Versionamento	Versionamento de código fonte	Git
Repositório	Repositório do código fonte	GitHub
Implantação	*Container*	Docker
CI/CD	*Pipeline* de integração contínua e entrega contínua	Jenkins Pipelines
Testes	Testes automatizados	xUnit
Dependências	Gerenciamento de Dependências	Nuget e NPM
Compilação	Geração de artefatos de *backend*	MSBuild
Transpilação	Geração de artefatos de *frontend*	Webpack
Segurança	Autenticação e Autorização	JSON Web Token baseado em perfis (*Roles*), afirmações (*Claims*) e políticas (*Policies*)
API Gateway	Integração	Ocelot para .NET Core
Descrição de API	Descrição, consumo e visualização de API's RESTful	Swagger
API	Microsserviços	REST/RESTful
Integração com outros módulos / sistemas	Interfaces utilizando JSON	API's REST
ESB	Barramento de Serviços	Mule ESB
Mensageria	Integração	RabbitMQ
Frontend	Interface homem máquina	Typescript Angular e CSS Bootstrap

4. Modelagem e projeto arquitetural

Nesta seção são apresentados os diagramas que permitem entender a arquitetura da aplicação, detalhando-a suficientemente para viabilizar sua implementação.

4.1 Modelo de casos de uso

Os diagramas de casos de uso oferecem uma visão global dos requisitos do sistema representados pelos casos de uso e pelos atores que dele participam. Para uma melhor análise arquitetural do projeto, os casos de uso foram agrupados por módulos de acordo com a descrição dos requisitos funcionais na seção 3.1.

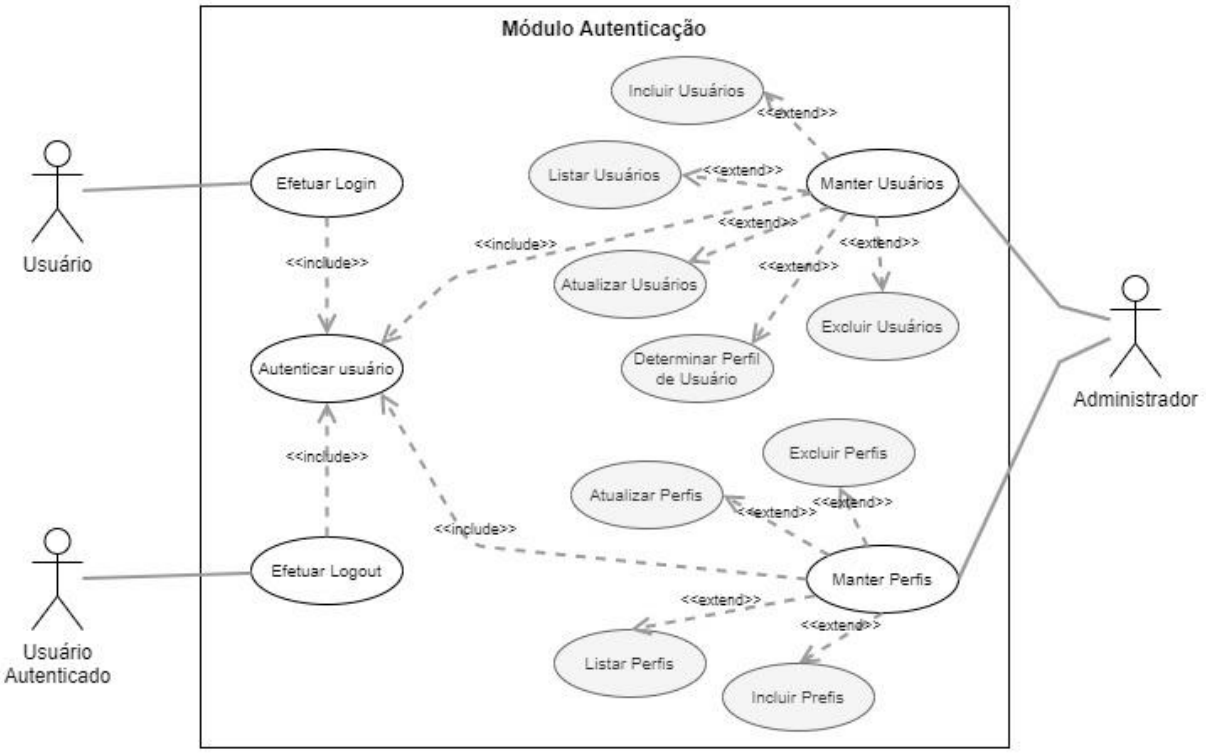

Figura 2 - Casos de Uso do Módulo Autenticação.

- Módulo Autenticação:
  - Caso de uso: **Efetuar login**

  Possibilitar que um usuário do sistema possa obter acesso ao sistema através de um token de autenticação após informar seu nome de usuário e senha.
  - Caso de uso: **Efetuar logout**

  Possibilitar que um usuário autenticado possa se desconectar do sistema.

- Caso de uso: **Manter usuários**

Possibilitar que os usuários administradores do sistema possam gerenciar os demais usuários através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

- Caso de uso: **Manter perfis de acesso**

Possibilitar que os usuários administradores do sistema possam gerenciar os perfis de acesso atribuídos aos usuários através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

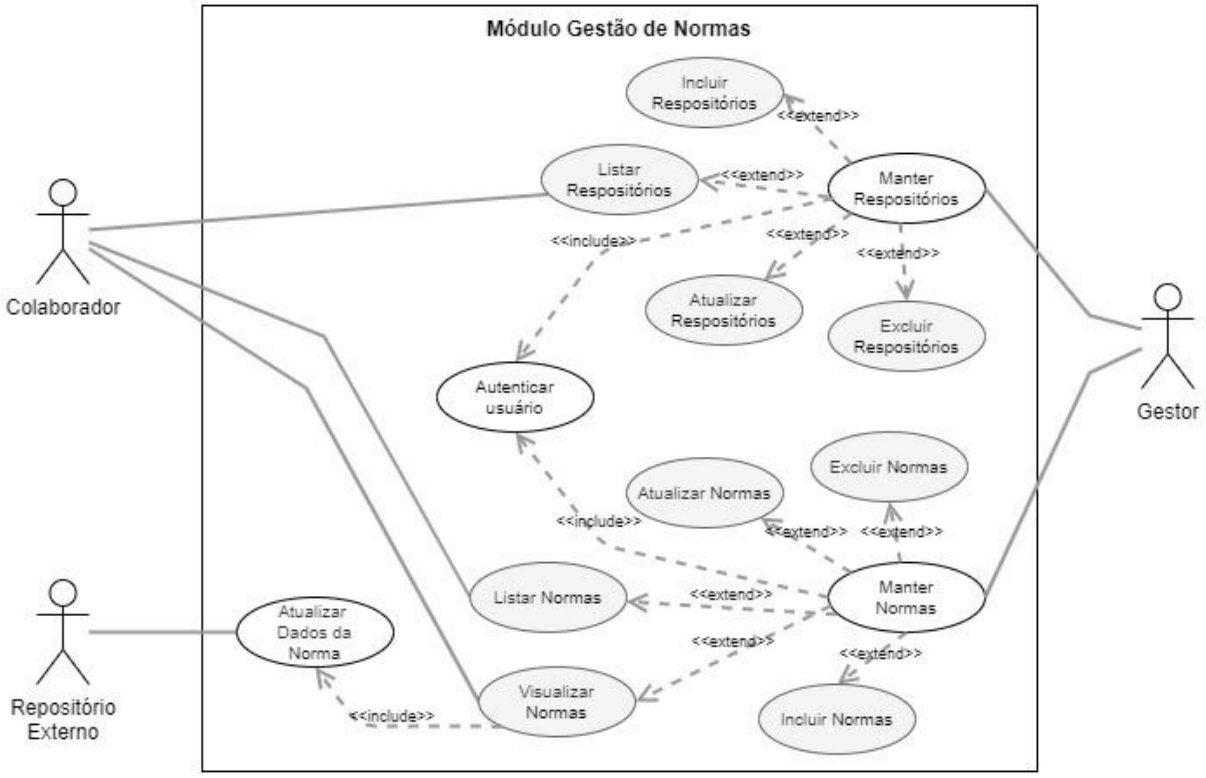

Figura 3 - Casos de Uso do Módulo Gestão de Normas.

- <u>Módulo Gestão de Normas</u>:
  - Caso de uso: **Atualizar dados da norma**

Possibilitar que o sistema consulte um sistema externo, detentor das informações sobre uma norma, para atualização dos dados da norma, incluindo uma versão atualizada dos documentos quando for o caso.

- Caso de uso: **Manter Repositórios**

Possibilitar que os usuários gestores possam gerenciar os cadastros de repositórios através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

Possibilitar aos usuários colaboradores acesso aos cadastros de repositórios para listar e visualizar os dados.

- Caso de uso: **Manter Normas**

Possibilitar que os usuários gestores possam gerenciar os cadastros de normas através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

Possibilitar aos usuários colaboradores acesso aos cadastros de normas para listar e visualizar os dados.

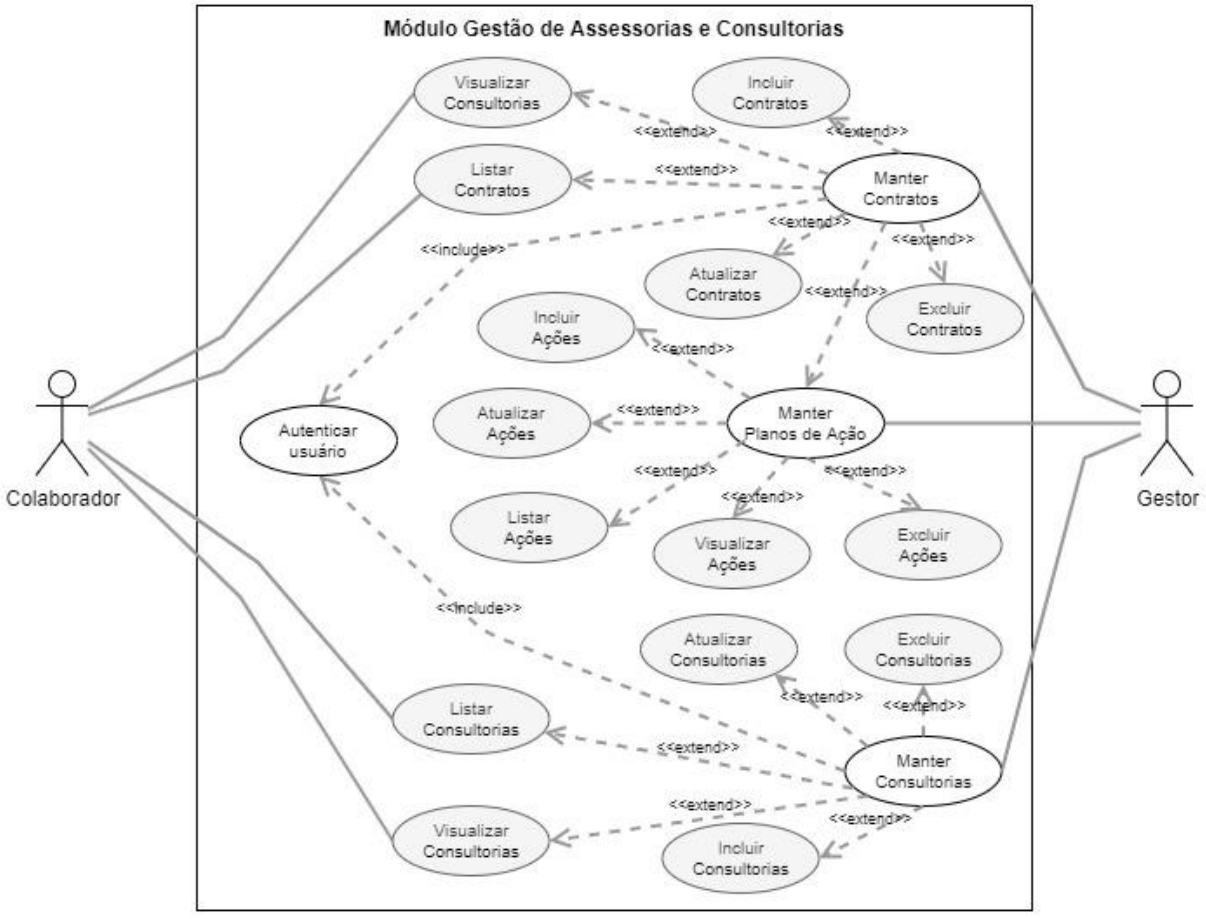

Figura 4 - Casos de Uso do Módulo Gestão Assessorias e Consultorias.

- <u>Módulos Consultorias e Assessorias</u>:
  - Caso de uso: **Manter Consultorias**

Possibilitar que um usuário gestor possa gerenciar os cadastros de consultorias através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

Possibilitar aos usuários colaboradores acesso aos cadastros de consultorias para listar e visualizar os dados.

- Caso de uso: **Manter Contratos**

Possibilitar que um usuário gestor possa gerenciar os contratos através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

Possibilitar aos usuários colaboradores acesso aos cadastros de contratos para listar e visualizar os dados.

- Caso de uso: **Manter Planos de Ação**

Possibilitar que os usuários gestores possam gerenciar planos de ação relacionados aos contratos através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

Possibilitar aos usuários colaboradores acesso aos planos de ação relacionados aos contratos para listar e visualizar os dados.

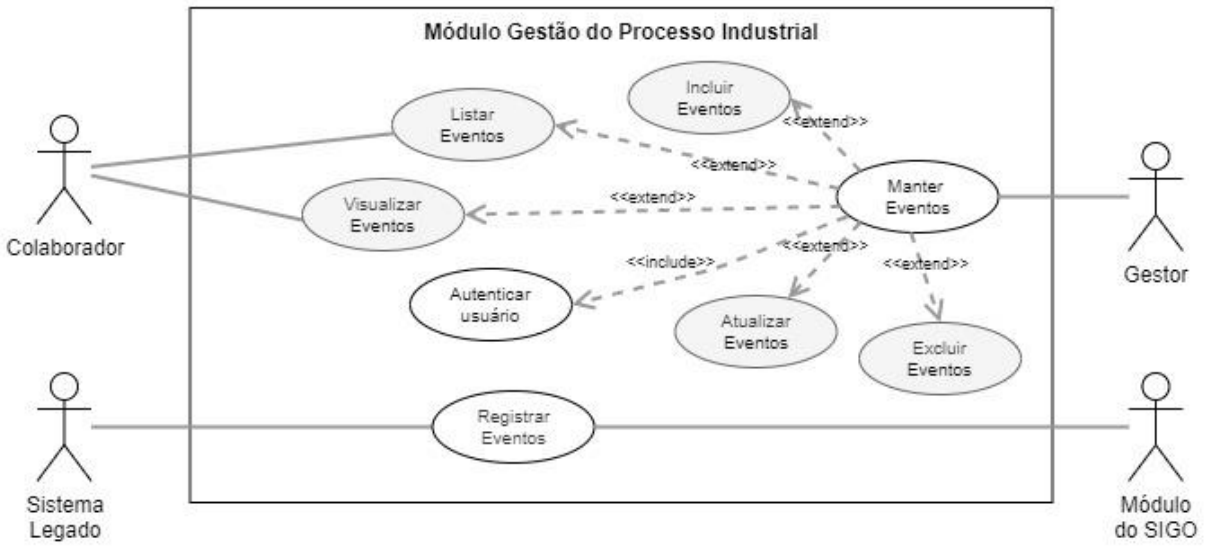

Figura 5 - Casos de Uso do Módulo Gestão do Processo Industrial.

- Módulo Gestão do Processo Industrial:

- Caso de uso: **Manter Eventos**

Possibilitar que os usuários gestores possam gerenciar os eventos relacionados aos diversos sistemas da solução através das funcionalidades: cadastrar, listar, visualizar, atualizar e excluir.

Possibilitar aos usuários colaboradores acesso aos eventos relacionados aos diversos sistemas da solução para listar e visualizar os dados.

- Caso de uso: **Registrar Eventos**

Possibilitar que os sistemas legados e os módulos do SIGO possam registrar.

4.3. Modelo de componentes

O modelo de componentes pretende demonstrar os componentes utilizados na solução, apresentando as tecnologias utilizadas e os mecanismos utilizados para comunicação entre os componentes.

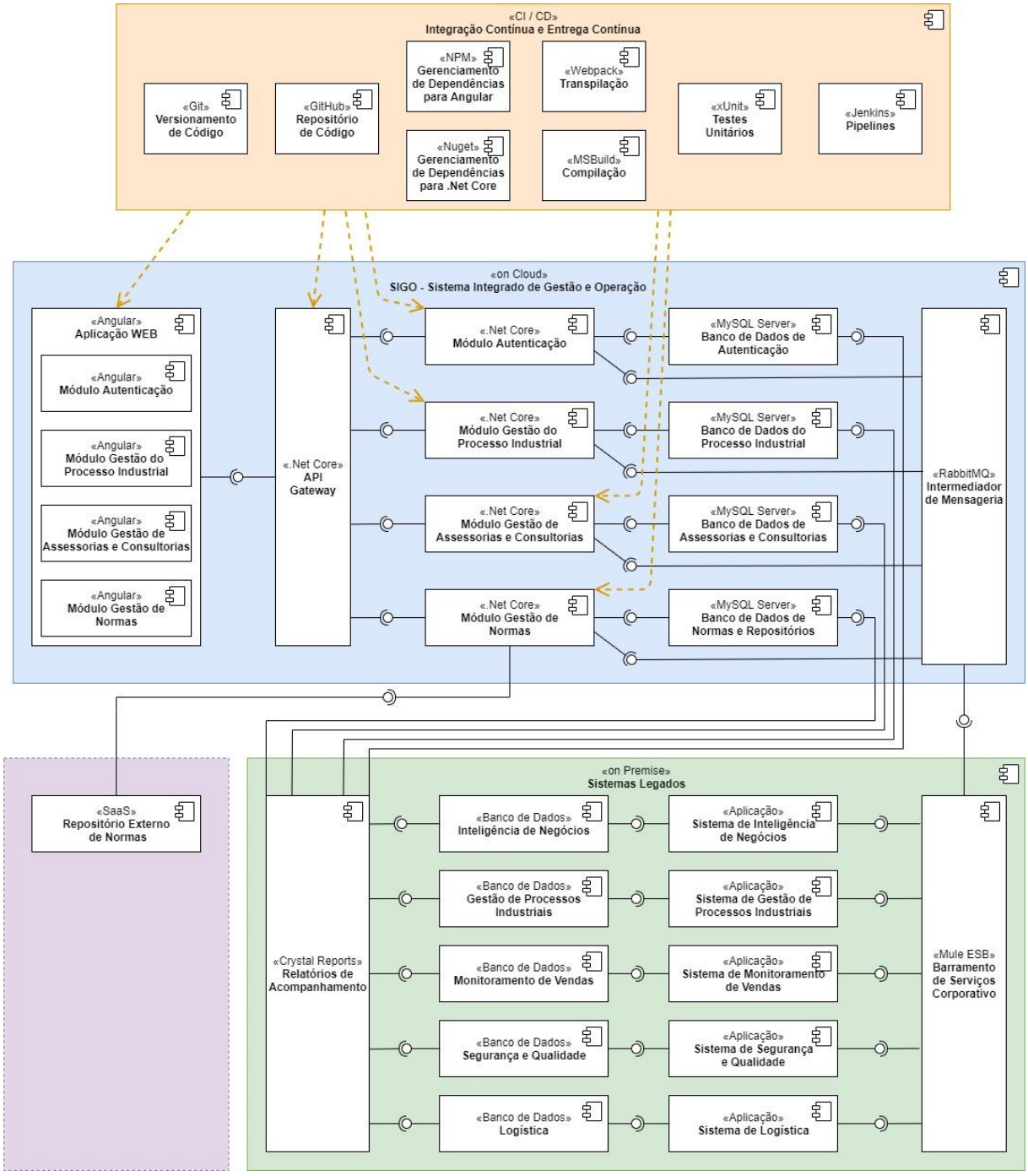

Figura 6 - Diagrama de Componentes.

O projeto buscou uma arquitetura modular, baseada em microsserviços e que possibilite desenvolvimento, testes e implantação de forma independente. Apesar disso, os módulos foram projetados de forma a prover integração entre os diversos ativos tecnológicos existentes, tanto dos novos componentes a serem desenvolvidos, quanto dos sistemas legados e já em operação na empresa.

Cada módulo é composto por um microsserviço desenvolvido utilizando a tecnologia .Net Core sobre a linguagem C# e realizando a persistência dos dados por meio de um banco de dados relacional provido pelo MySQL. A interface do usuário para o acesso ao módulo está inclusa em uma aplicação SPA, o que possibilita maior velocidade no acesso e na renderização das interfaces. Esta aplicação é gerada por meio de códigos produzidos utilizando os frameworks Angular, com a linguagem Typescript, e Bootstrap CSS. Atendendo assim as restrições arquiteturais de adotar microsserviços, modularidade, uso de tecnologias livres e robustas, persistência por meio de SGBD, interface de usuário em formato SPA

A aplicação de interface do usuário se comunica com os microsserviços passando obrigatoriamente por um API Gateway que vai intermediar as requisições e rotear a comunicação com os pontos de acesso dos microsserviços. Nenhum microsserviço dos módulos será exposto diretamente na web e somente poderá ser descoberto e acessado por meio do API Gateway.

Todas as comunicações entre interface do usuário e os microsserviços deverão utilizar um token JWT. Este token é gerado quando da autenticação do usuário e contém informações referentes a autorização do usuário para realizar operações no sistema, bem como uma assinatura digital que possibilita sua validação quanto a modificação do conteúdo. Requisições sem a presença do token serão automaticamente recusadas e o usuário direcionado para página de autenticação. Desta forma, é possível recuperar informações de autorização sem a necessidade de manter estado em algum serviço e sem comprometer a segurança.

A comunicação entre os módulos novos e os sistemas legados será realizada por meio do sistema de mensageria RabbitMQ, que também é uma tecnologia livre e robusta, que suporta uma série de protocolos de mensageria, incluindo ainda o enfileiramento, roteamento e confirmação de entrega de mensagens, além do monitoramento de recursos e a escalabilidade horizontal. Os novos módulos publicarão ou estarão subscritos em filas diretamente no sistema RabbitMQ. Os ativos legados que utilizam protocolo SOAP e *middlewares* Orientados a

Mensagem estarão hospedados *on Premise*, e serão integrados por meio de um sistema Enterprise Service BUS. Este fará então a entrega e o consumo de mensagens com os módulos do SIGO através do RabbitMQ, provendo assim integração total entre os sistemas. Atendendo desta forma as restrições arquiteturais de o sistema apresentar características de aplicações distribuídas com uso extensivo de redes, além da possibilidade de hospedar o sistema em nuvem híbrida, mantendo os ativos legados *on Premise*.

Todo o sistema foi projetado para utilização ampla das tecnologias de containers, buscando maior eficiência no desenvolvimento e implantação de código, bem como fazendo uso de recursos de infraestrutura como código para maior isolamento e portabilidade do sistema.

Para versionamento foi utilizado o GIT, em conjunto com o serviço de repositório do Github e com uso do GitFlow. Para Integração Contínua, espera-se utilizar a ferramenta Jenkins, responsável por executar os testes unitários, de estresse e de carga na aplicação, além de criar uma imagem do container para o módulo alterado (realizar o *deploy* automático a partir da definição dos pipelines). Atendendo assim as restrições arquiteturais de que o sistema deva ser compilado e implantado utilizando recursos de integração e entrega contínuas, bem como os requisitos não funcionais de testabilidade e manutenibilidade.

4.4. Modelo de implantação

Para uma melhor representação da alocação física dos componentes da solução, um diagrama de implantação apresenta o ambiente de implantação híbrido projetado, demonstrando os componentes de hardware alocados para implantação dos diversos componentes, bem como a comunicação entre eles.

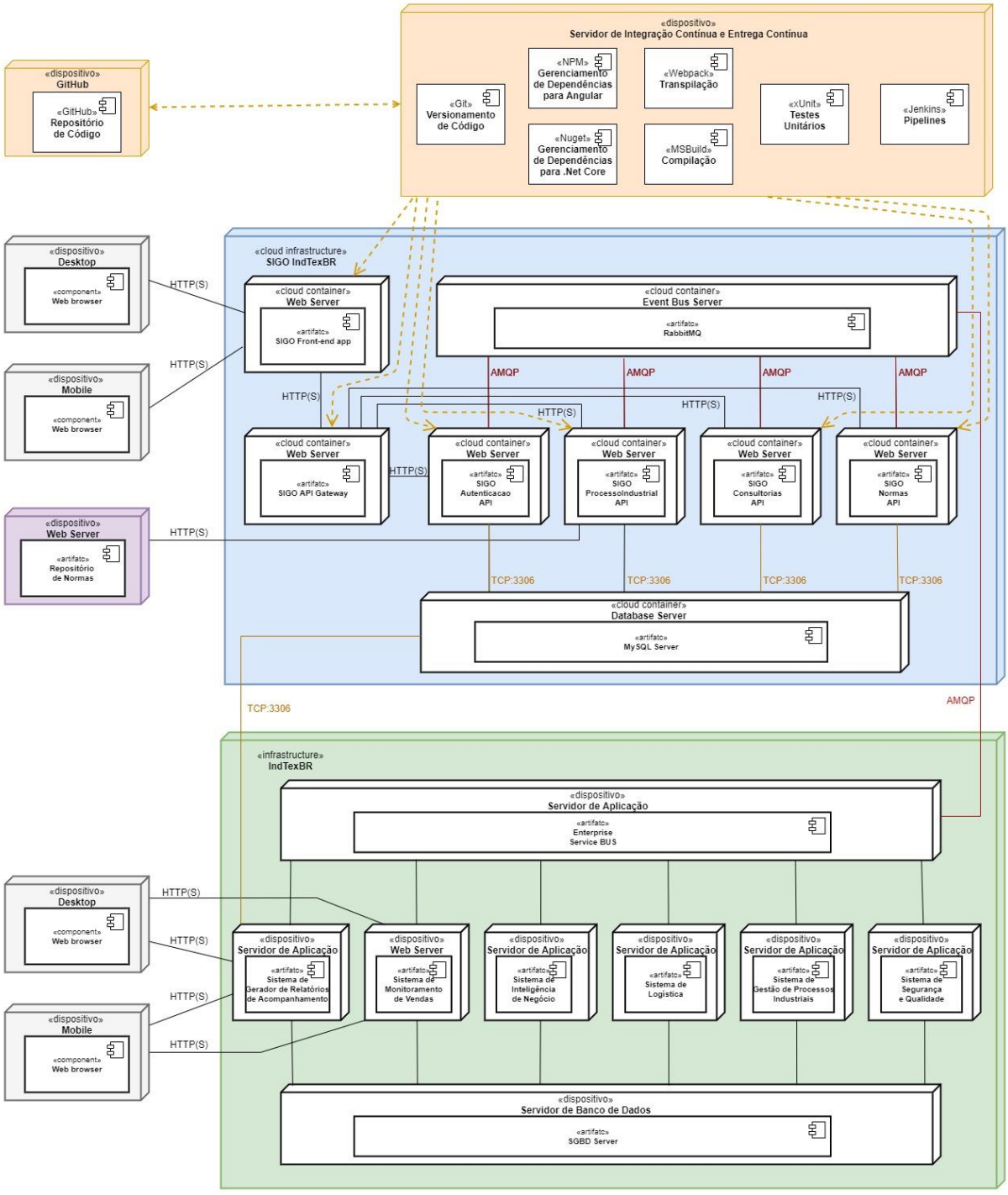

Figura 7 - Diagrama de Implantação.

4.5. Modelo de dados

Modelos de dados baseados nas classes componentes de cada módulo do sistema para a implementação da prova de conceito.

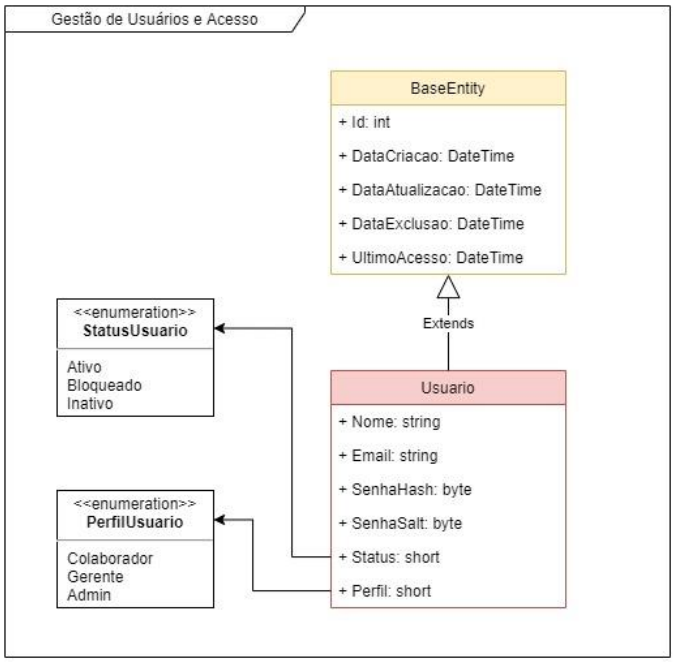

Figura 8 - Módulo Gestão de Usuários e Acesso.

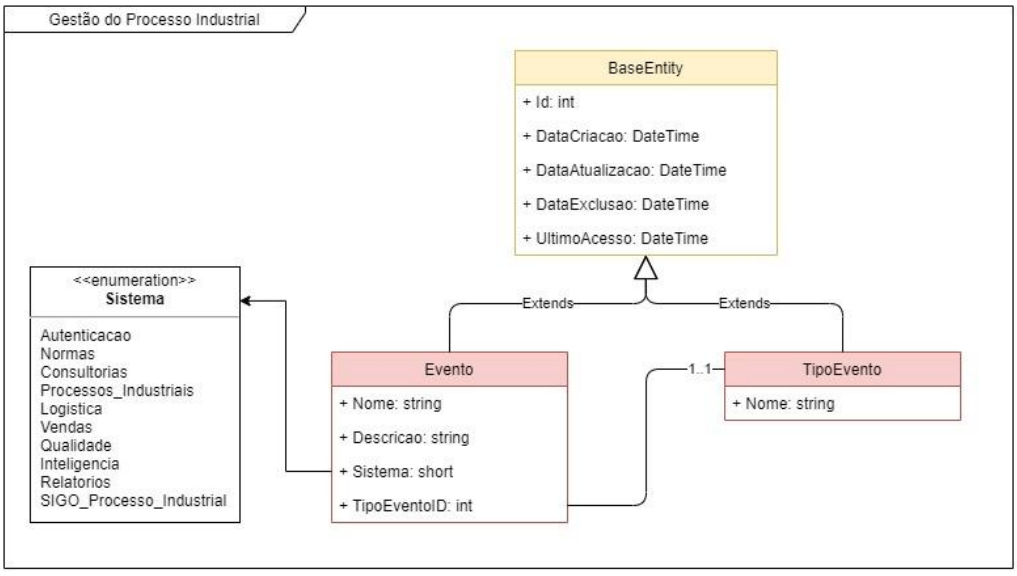

Figura 9 - Módulo Gestão do Processo Industrial.

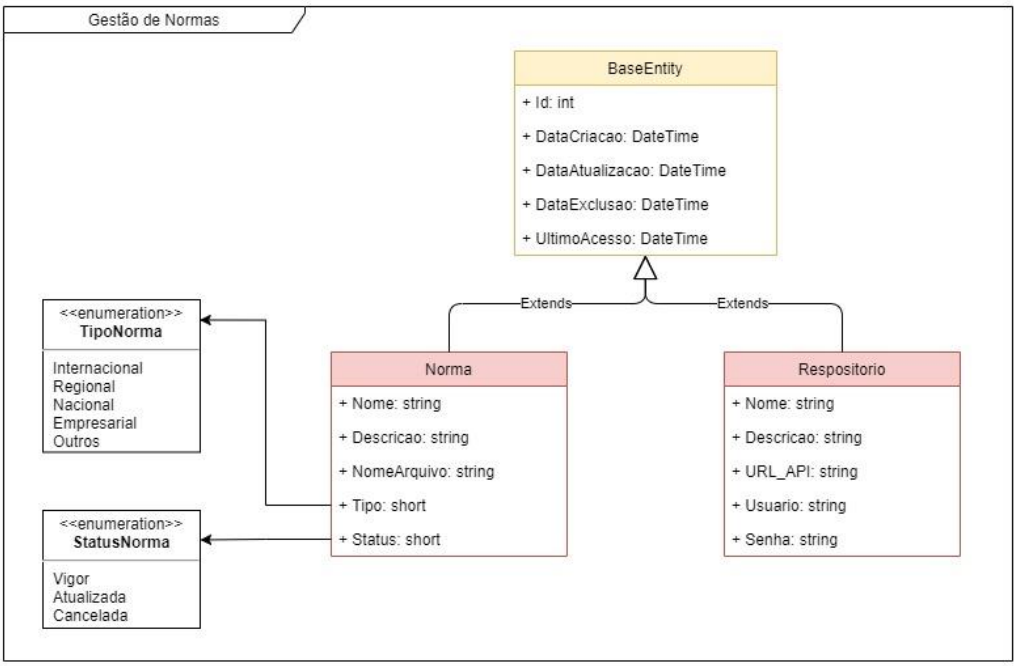

Figura 10 - Módulo Gestão de Normas.

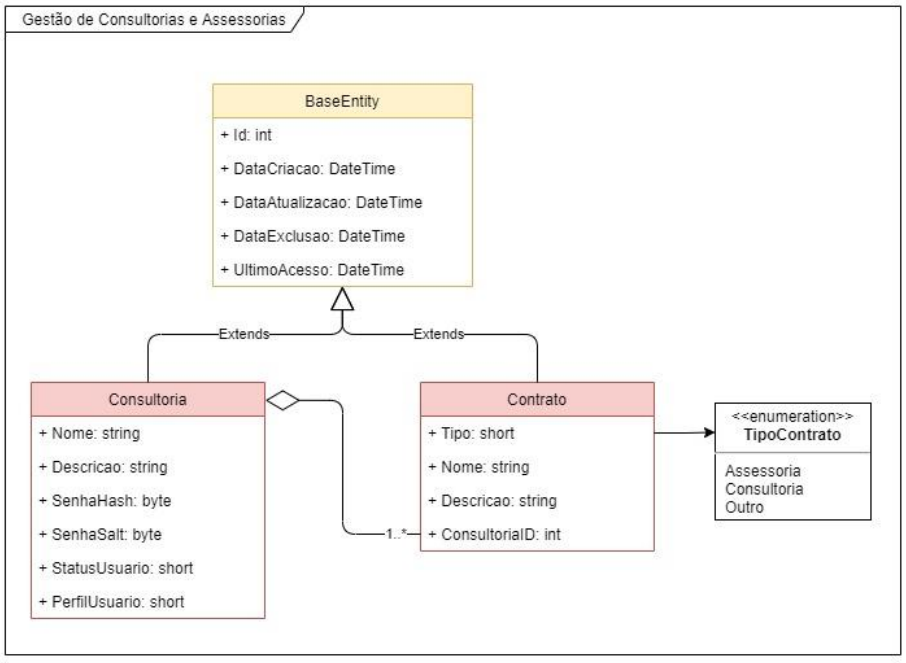

Figura 11 - Módulo Gestão de Consultorias e Assessorias.

5. Prova de Conceito (POC) / Protótipo Arquitetural

A prova de conceito (*Proof of Concep*t – POC) foi construída objetivando validar o entendimento dos requisitos funcionais e não funcionais, e realizar a implementação dos requisitos e restrições arquiteturais especificados neste documento.

A seção 5.1. Implementação e Implantação descreve a implementação da prova de conceito da arquitetura (protótipo arquitetural) da aplicação enunciando as tecnologias utilizadas, os casos de uso priorizados, os requisitos não funcionais avaliados e o código fonte.

A seção 5.2 Interfaces / APIs descreve as interfaces dos microsserviços disponíveis.

5.1. Implementação e Implantação

5.1.1 Tecnologias utilizadas

A prova de conceito foi construída utilizando as tecnologias especificadas na seção 3.4 Mecanismos Arquiteturais.

Para o sistema *Enterprise Service Bus* (ESB) foi utilizada uma abordagem de envio direto de requisição para o sistema de Mensageria utilizando o Postman como interface. Esta abordagem visa simular as requisições dos sistemas legados por meio do ESB pelo qual serão integrados ao SIGO.

5.1.2 Casos de uso

Para a criação da POC alguns casos de uso foram priorizados na implementação. Os casos de uso contemplados são apresentados na tabela abaixo:

Módulo	Caso de Uso	Requisito não funcional
Autenticação	Autenticar Usuário	Segurança
Normas	Visualizar dados de uma Norma cadastrada	Interoperabilidade
		Usabilidade
		Acessibilidade
Gestão do Processo Industrial	Consultar Eventos	Interoperabilidade

5.1.3 Requisitos não funcionais

Conforme pode ser observado, os casos de uso priorizados para a realização da prova de conceito visam validar aspectos importantes da arquitetura tendo como base os requisitos não funcionais:

- **Segurança**

  Priorização de requisito não funcional motivada pela necessidade sempre atual de manter dados sensíveis, recursos e usuários protegidos de acessos não autorizados.

  Os critérios de aceite são:

  - O sistema deve permitir que usuários sem permissão possam acessar recursos não privados.
  - O sistema não deve permitir que usuários sem permissão possam acessar recursos privados.
  - O sistema deve redirecionar o usuário para tela de autenticação ao identificar que um acesso a recurso privado está sendo feito sem autenticação.

- **Interoperabilidade**

  Priorização de requisito não funcional motivada pela necessidade de validar a comunicação entre os ativos de *software* (novos e existentes).

  Os critérios de aceite são:

  - Os módulos a serem desenvolvidos devem ser capazes de se comunicar entre si.
  - Os módulos a serem desenvolvidos devem ser capazes de se comunicar com os sistemas legados.
  - O sistema deve se comunicar com recursos externos.

- **Usabilidade**

  Priorização de requisito não funcional motivada pela necessidade constante de prover uma melhor experiência aos usuários, sem perder o foco nos processos.

  Os critérios de aceite são:

- As telas do sistema devem apresentar funcionalidades simples e objetivas.
- O sistema deve apresentar interfaces de fácil navegação.

- **Acessibilidade**

  Priorização de requisito não funcional motivada pela necessidade de validar a adaptabilidade das interfaces de usuário aos diferentes dispositivos e tamanhos de tela.

  Os critérios de aceite são:

  - O sistema deve oferecer as mesmas funcionalidades em todos os dispositivos.
  - O sistema deve manter cores e estilos de elementos consistentes entre as páginas.
  - O sistema deve funcionar de forma consistente em dois ou mais navegadores.

- **Escalabilidade**

  Priorização de requisito não funcional motivada pela necessidade de os sistemas atuais serem mantidos em constante operação, 24 horas por dia, nos sete dias da semana.

  Os critérios de aceite são:

  - O sistema deve ser capaz de escalar horizontalmente seus componentes para atender a um aumento no número de requisições.
  - O sistema deve ser capaz de liberar os recursos tão logo o volume de requisições retorne ao volume normal definido para produção.

Certamente, para a construção final do sistema, há também a necessidade de adequação aos requisitos não funcionais levantados e descritos na seção 3.1 Requisitos não funcionais. Entretanto, como se trata de uma prova de conceito, a implementação se limitará a exercitar os requisitos acima especificados, sem prejuízo de futura implementação dos demais.

5.1.4 Código

O código-fonte da prova de conceito da solução proposta foi escrito observando aos princípios SOLID, apresentados por Robert C. Martin em um artigo publicado no ano 2000. Estes princípios objetivam a criação de estruturas de software sejam fáceis de entender, possam

ser modificados, e possam ser reutilizados. SOLID é um acrônimo que representa 5 princípios que norteiam a busca destes objetivos. A tabela a seguir apresenta um breve resumo do SOLID:

Princípio		**Definição**	
S	*Single Responsibility Principle*	Princípio da responsabilidade única	A melhor estrutura para um sistema deve ser influenciada pela estrutura social que a organização utiliza, de forma que cada módulo de software tenha somente uma razão para mudar.
O	*Open-Closed Principle*	Princípio Aberto-Fechado	Sistemas de software devem ser fáceis de mudar, permitindo que possam ser modificados pela adição de código ao invés da modificação do código existente.
L	*Liskov Substitution Principle*	Princípio da substituição de Liskov	Sistemas de software devem ser criados a partir de partes intercambiáveis, que aderem a um contrato específico, o que permite sua substituição quando necessário.
I	*Interface Segregation Principle*	Princípio da Segregação da Interface	Uma classe não deve ser forçada a implementar interfaces e métodos que não irão utilizar. Evitando assim que o projeto de software evite depender de coisas que não serão utilizadas.
D	*Dependency Inversion Principle*	Princípio da Inversão de Dependência	O código deve depender de abstrações e não das implementações. Definindo uma política em que código que implementa detalhes de alto nível não dependa de código que implementa detalhes de níveis mais baixos.

Também foram adotados os princípios do Design Orientado por Domínio (*Domain Driven Design* – DDD) que direcionam a modelagem e codificação do sistema com base na realidade dos negócios, falando sobre problemas como domínios e descrevendo as áreas de problema independentes como contextos limitados.

Os princípios DDD têm como objetivo facilitar a implementação de regras e processos complexos realizando a divisão de responsabilidades através de camadas conforme a descrição a seguir:

- Camada de aplicação: é o código do projeto principal onde são desenvolvidos os controladores e serviços da API. Sua função é receber as requisições e direcionar para o serviço específico que irá executar a determinada ação. Possui referências para as camadas Service e Domain.
- Camada de domínio: é o código de implementação de classes/modelos que serão mapeadas para o banco de dados, as declarações de interfaces, constantes, DTOs (*Data Transfer Object*) e *enums.*
- Camada de serviço: é o código de operação do projeto onde são definidas as regras de negócio e realizadas as validações dos dados antes da persistência. Possui referências para as camadas *Domain* e *Infrastructure*.
- Camada de infraestrutura: é dividida em duas subcamadas
    - *Data*: responsável pela persistência dos dados, utilizando, ou não, algum ORM.
    - *Cross-Cutting*: contém funcionalidades que podem ser utilizadas em qualquer parte do código, como por exemplo validação de dados ou consumo de API externa. Possui referências para a camada Domain.

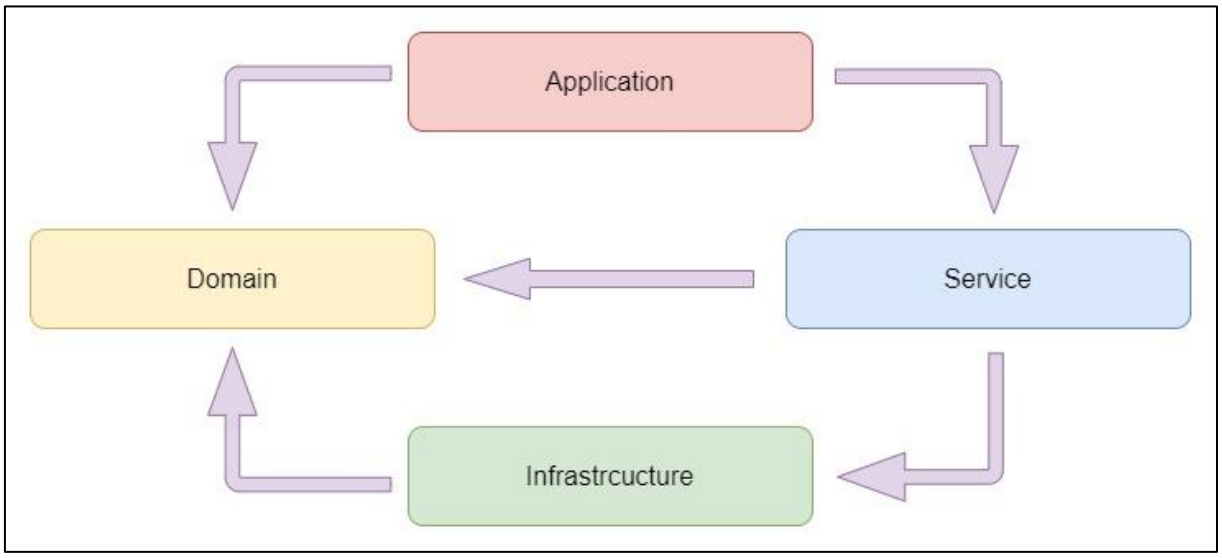

Figura 12 - Camadas propostas no DDD.

A figura a seguir demonstra a aplicação dos princípios abordados pelo DDD na divisão da estrutura do código fonte do projeto.

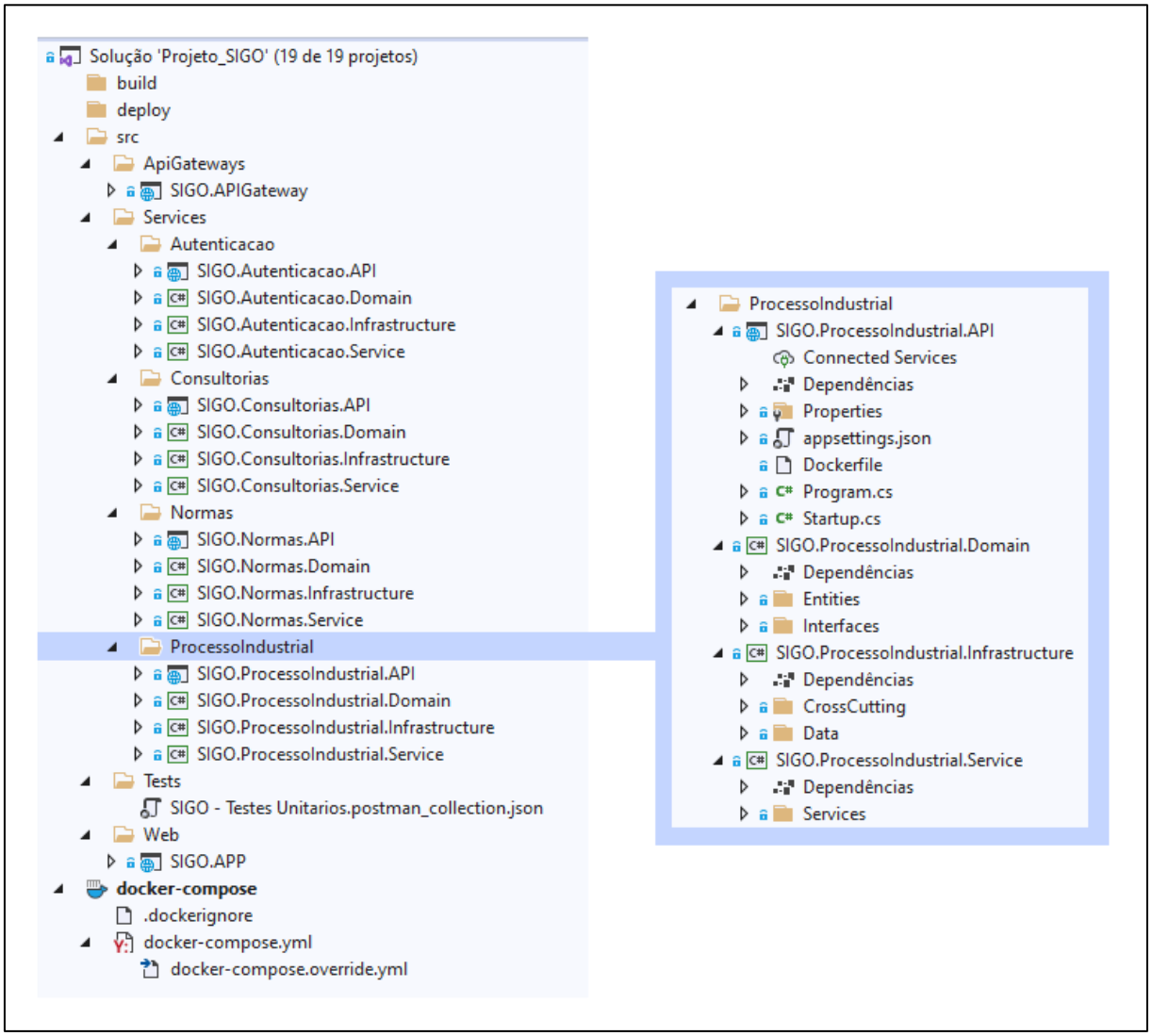

Figura 13 - Estrutura do Código do Sistema.

As APIs foram desenvolvidas através de múltiplos projetos independentes de microsserviços. Estes microsserviços foram construídos utilizando como base projetos de API Web do .Net Core 3.1. A segurança do acesso aos serviços se vale da implementação de filtros de autorização, que para cada chamada efetuada pelo cliente deve verificar se necessita de autorização (JWT token).

O API Gateway foi desenvolvido com base na biblioteca Ocelot para .Net. Esta biblioteca apresenta como funcionalidades: agregação de solicitações; roteamento; autenticação; cache; balanceamento de carga; log; *service fabric*; dentre outras. A porta de entrada para os clientes das APIs dos serviços é realizada unicamente pelo API Gateway. Este componente

direciona as solicitações para cada microsserviços apropriado, funcionado como uma camada intermediária.

A interface de usuário foi constituída por meio de uma aplicação web. As páginas web que compõem esta aplicação foram desenvolvidas utilizando os *frameworks* Javascript Angular e CSS Bootstrap. As bases para a estrutura de código e para a estrutura de funcionamento utilizam o conceito de aplicação de página única (*Single Page Application* - SPA).

Nesta aplicação todo acesso se inicia por meio de uma página pública de login, e após a autenticação do usuário, ocorre o direcionamento para a área das páginas privadas contendo as funcionalidades de cada módulo do sistema.

Para ter acesso a estas páginas privadas, o usuário deve possuir um token (*Json Web Token* – JWT), uma chave de acesso que é gerada pelo serviço de autenticação no login. Esta chave de acesso tem validade definida e pode ter seu conteúdo verificado e validado a qualquer tempo pelo sistema. Para as páginas públicas não é necessário o token.

5.2 Interfaces / APIs

Como os módulos do SIGO foram construídos como microsserviços, é necessária a apresentação da descrição de suas interfaces, incluindo a visualização e a forma de consumo tanto para testes quanto para desenvolvimento. Para isso foi utilizado o framework Swagger. A utilização do Swagger possibilita a implementação da especificação *OpenAPI* para documentação de APIs.

As APIs somente poderão ser acessadas por intermédio do *APIGateway*. Este componente possibilita implementar melhorias de segurança como a centralização e a padronização dos *endpoints*, bem como possibilita a alteração destes sem afetar as rotas definidas na aplicação cliente.

Os recursos de todas as APIs são protegidos, assim sendo o usuário deve informar um token de autorização válido para acessá-los. Os recursos dos módulos listados a seguir somente podem ser acessados e consumidos por intermédio do *APIGateway*.

O recurso “autenticar” da API “SIGO.Autenticacao.API” é acessível de forma aberta pois seu objetivo e possibilitar a autenticação e autorização dos usuários do sistema.

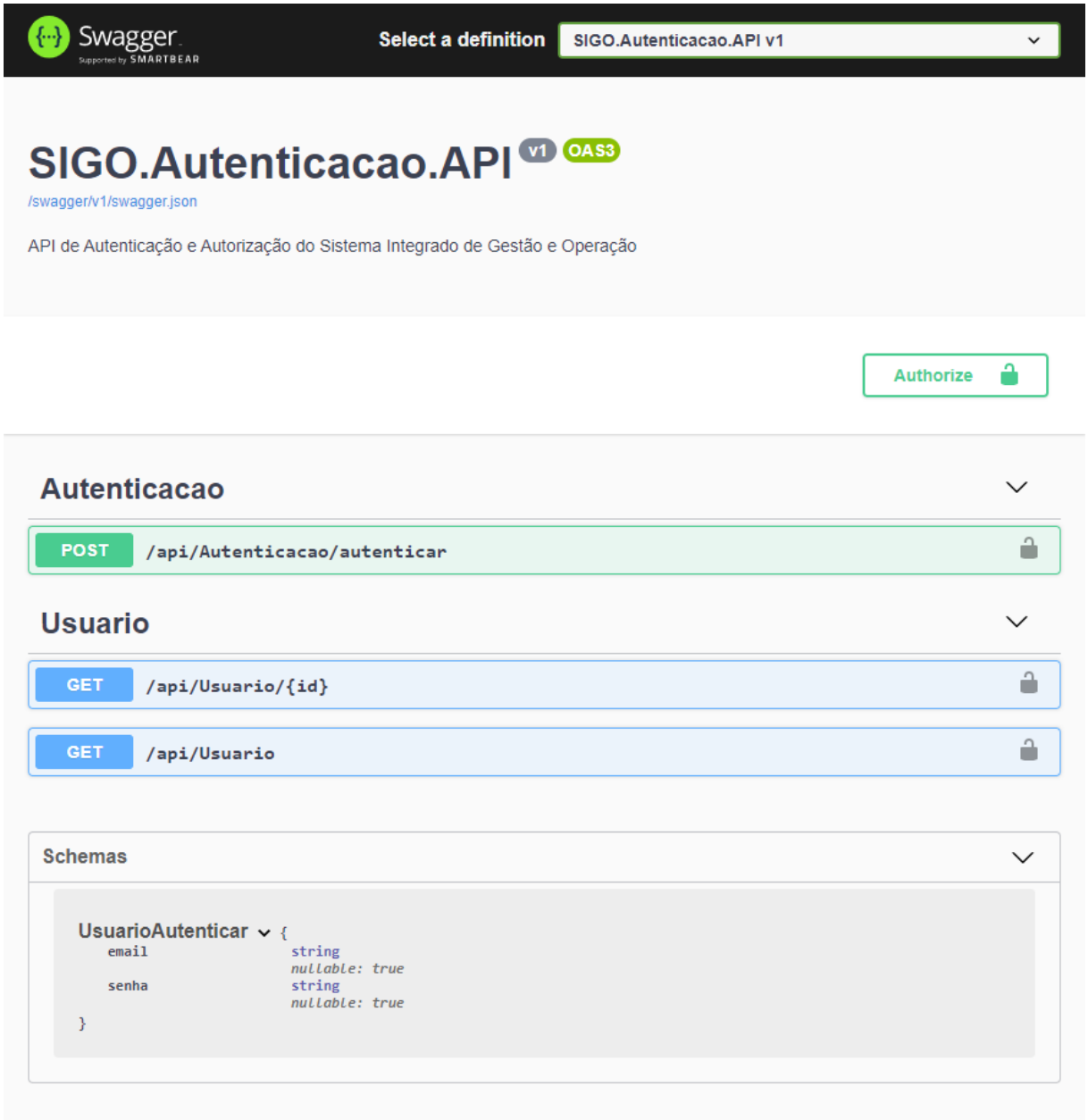

Figura 14 - Interface das APIs do Módulo Autenticação.

A **SIGO.Autenticacao.API** é responsável por prover as interfaces para:

- Autenticacao: Possibilitar a autenticação do usuário mediante o fornecimento de credenciais e a geração do Token de acesso ao sistema que inclui as informações de autorização de acesso aos recursos do sistema.
- Usuario: Possibilitar o gerenciamento completo dos usuários e perfis de acesso do sistema como um todo.

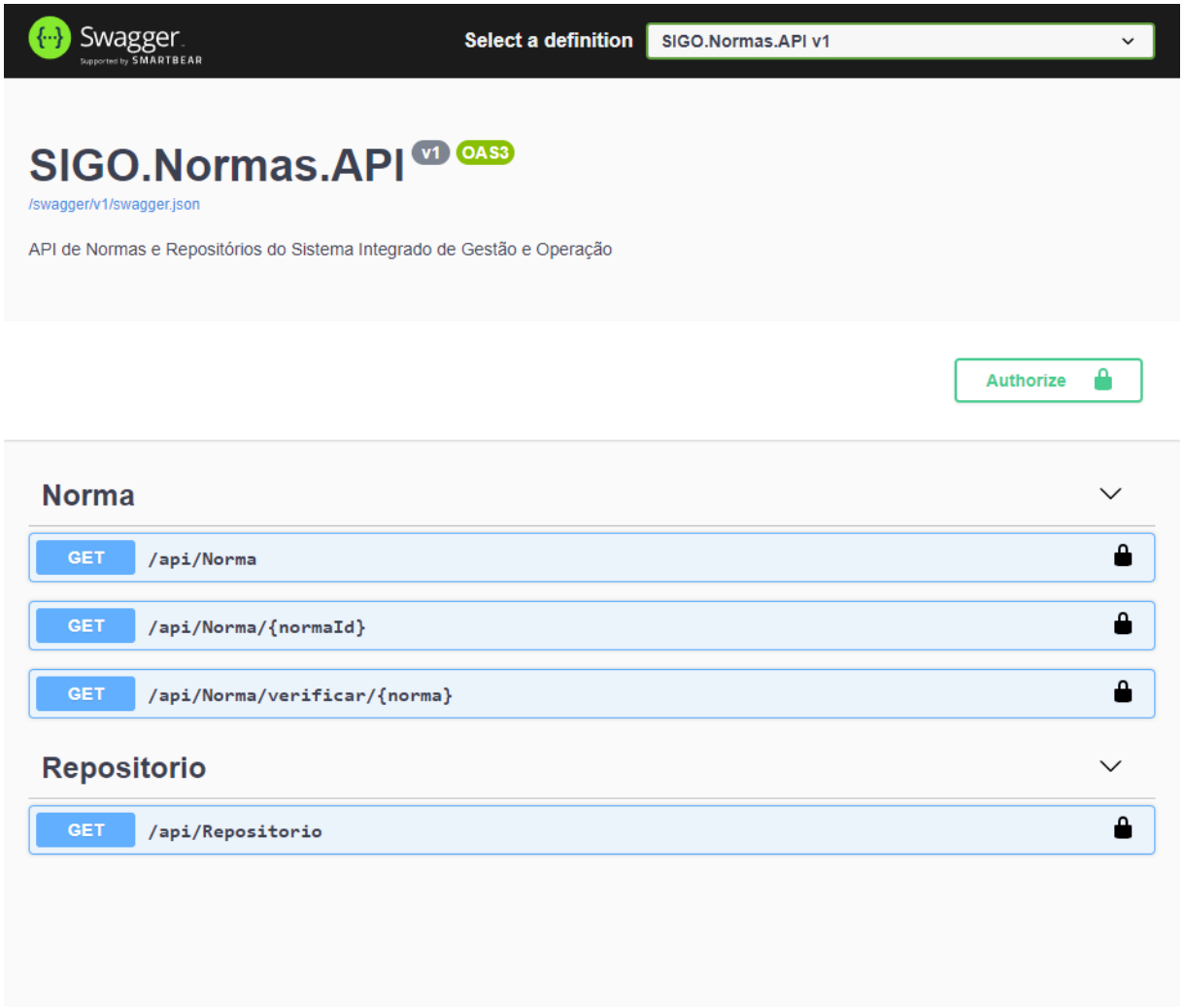

Figura 15 - Interface das APIs do Módulo Gestão de Normas.

A **SIGO.Normas.API** é responsável por prover as interfaces para:

- Norma: Possibilitar o gerenciamento completo do repositório eletrônico de normas técnicas. O *ponto de acesso "/api/Norma/Verificar"* faz a conexão ao repositório externo de normas para verificar a situação atual da norma.
- Respositorio: Possibilitar o gerenciamento completo dos repositórios externos dos quais o sistema fará consultas para identificar e notificar mudanças e evoluções das normas.

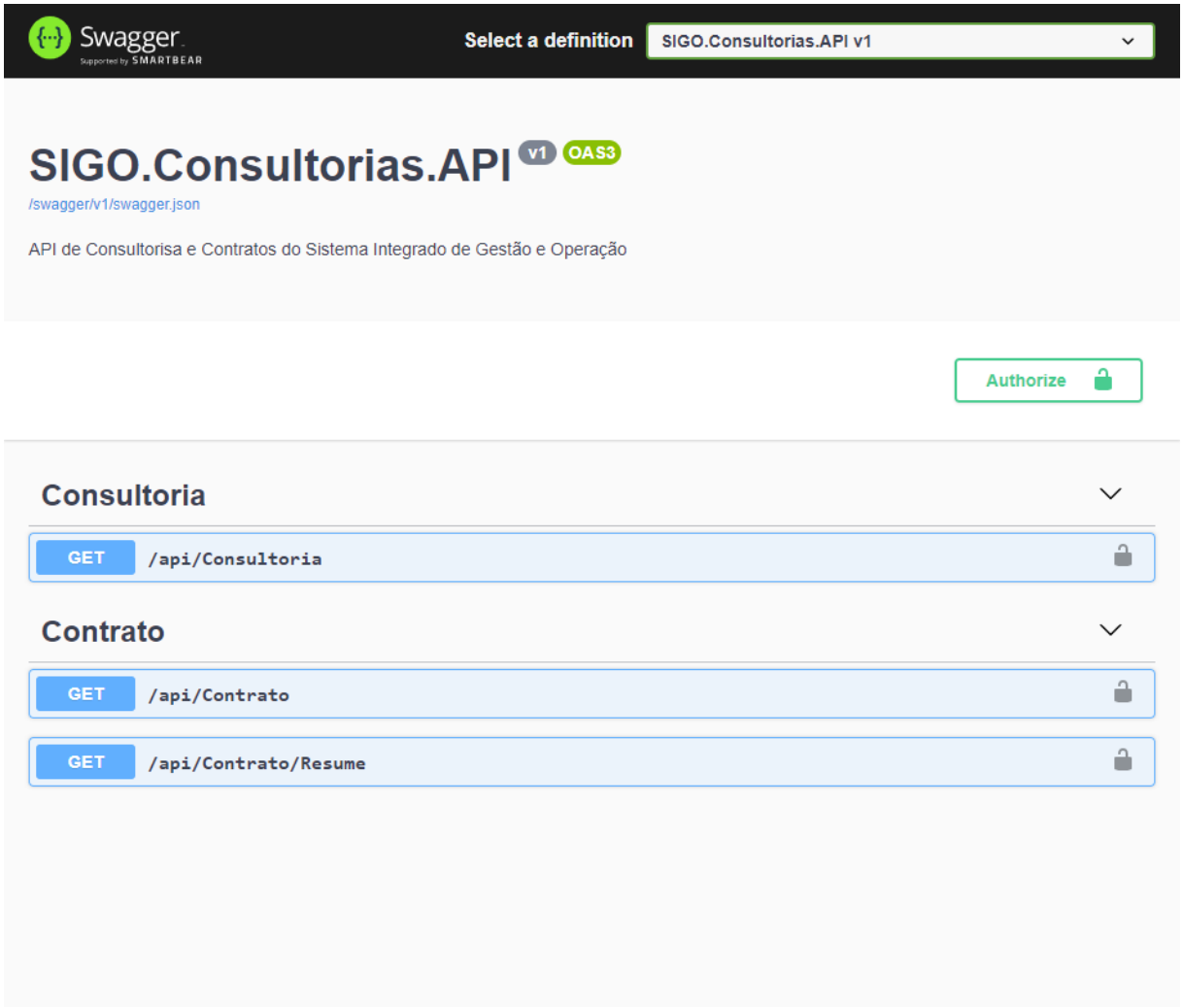

Figura 16 - Interface das APIs do Módulo Assessorias e Consultorias.

A **SIGO.Consultorias.API** é responsável por prover as interfaces para:

- Consultoria: Possibilitar o gerenciamento completo do cadastro das empresas de consultoria e assessoria.
- Contrato: Possibilitar o gerenciamento completo dos contratos de consultoria ou assessoria visando o planejamento e controle das atividades industriais.

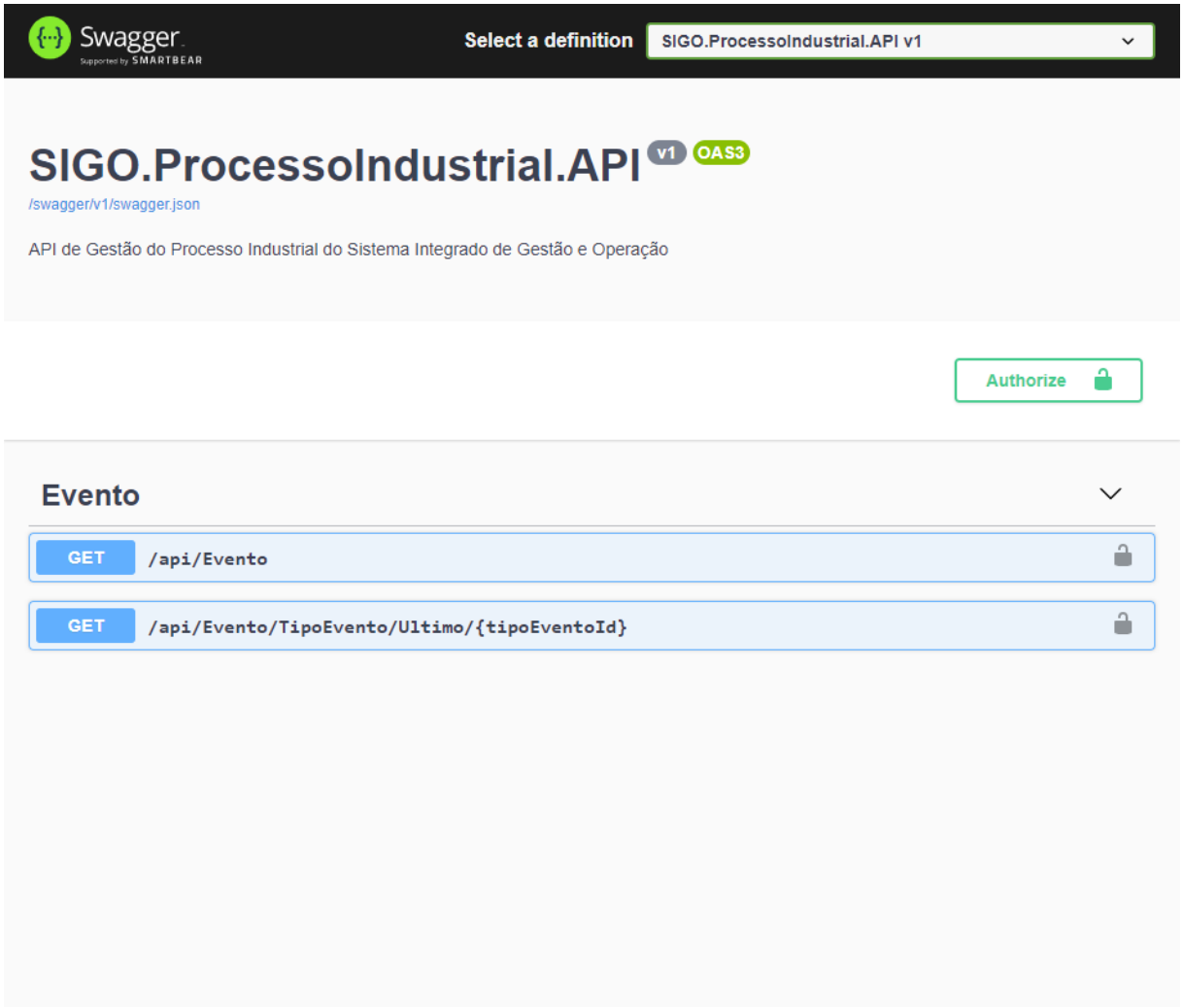

Figura 17 - Interface das APIs do Módulo Gestão do Processo Industrial.

A API **SIGO.ProcessoIndustrial.API** é responsável por prover as interfaces para:

- Evento: possibilitar a integração através de troca de mensagens com os demais módulos do SIGO e com os sistemas legados que incluem a solução ERP atualmente implantada.

6. Avaliação da Arquitetura

6.1. Análise das abordagens arquiteturais

A arquitetura foi projetada baseada em microsserviços, o que significa que cada serviço é fracamente acoplado e tem seu próprio contexto delimitado e bem definido. Esta arquitetura se mostra como uma opção adequada para aplicações que lidam com grandes volumes de tráfego, incluindo situações em que há a necessidade de escalabilidade horizontal face a aumentos de demanda, apresentando facilidade de manutenção eventual sem *downtime*, assim como na automação do processo de desenvolvimento, de testes, e de implantação e entrega.

Neste contexto, foram projetados componentes arquiteturais específicos para as funcionalidades de negócio, e selecionados componentes de mercado para funcionalidades genéricas, de forma que a solução resultante tenha a robustez necessária ao ambiente e cenário de aplicação, e esteja alinhada com as tecnologias emergentes quanto a composição. Todas as escolhas levaram em consideração os conhecimentos adquiridos no decorrer do curso através das aulas, dos estudos de caso, além do conhecimento compartilhado a respeito da adoção destas tecnologias por parte da comunidade de desenvolvimento de *software*.

De forma geral a arquitetura de microsserviços representa uma boa escolha para a implementação do projeto SIGO. Dentre suas principais vantagens podemos destacar para este caso em específico as seguintes:

- Possibilidade de utilização de diversas tecnologias simultaneamente para a composição da solução.
- Possibilidade de definição de escalabilidade automática baseada em parâmetros, incluindo a possibilidade de escalabilidade horizontal.
- Facilidade na construção e manutenção de aplicações, dado que os serviços têm escopo bem definido e isolado das demais funcionalidades.
- Possibilidade de dimensionar o serviço em partes separadas, viabilizando a definição de novas organizações nas equipes de desenvolvimento e manutenção.
- Maior produtividade no desenvolvimento e manutenção, pois cada serviço está relacionado a um processo específico do negócio, tem sua própria autonomia e oferece flexibilidade na tecnologia que será utilizada.

- Maior facilidade na implantação de novas funcionalidades, incluindo a possibilidade de implementar a automação nos processos de construção e entrega, dado que o *deploy* será realizado em um bloco isolado do sistema como um todo.

Para além deste projeto, a arquitetura baseada em microsserviços se mostra adequada aos mais diversos cenários de desenvolvimento de software da atualidade, nos quais o sistema deve estar preparado para lidar com a evolução constante pela adição de novas funcionalidades, seja para usuários humanos, seja para a integração com sistemas de diferentes fornecedores e plataformas, incluindo os sistemas legados.

6.2. Cenários

- **Cenário 1**

  Um usuário não autenticado no sistema tenta acessar algum recurso protegido do sistema, de forma direta, através dos componentes de tela providos pelo sistema, ou de forma indireta, através da URL de acesso direto ao recurso. O sistema deve permitir que páginas públicas sejam acessíveis a qualquer usuário, autenticado e não autenticado. O sistema deve restringir o acesso a recursos protegidos somente a usuários autenticados e autorizados.

  Este cenário garante o RNF de segurança.
- **Cenário 2**

  Um usuário registrado para ser notificado quando ocorrer determinado evento de sistema. Na ocorrência de eventos, como uma norma cadastrada no módulo de Gestão de Normas sofre atualização ou um evento gerado por algum dos sistemas legados, o módulo de Gestão do Processo Industrial deve emitir uma mensagem de alerta para o usuário.

  Este cenário em conjunto com o cenário 3 garante o RNF de interoperabilidade.
- **Cenário 3**

  Na visualização do cadastro de uma norma, o sistema deve se conectar ao repositório de normas automaticamente e realizar uma consulta dos dados mais atuais da norma em questão. Ao receber o resultado, o sistema deve emitir uma mensagem de alerta ao usuário com os dados da consulta realizada.

  Este cenário em conjunto com o cenário 2 garante o RNF de interoperabilidade.

- **Cenário 4**

  Um usuário, autenticado ou não autenticado, acessando o sistema por meio de um dispositivo conectado a web e navegando pelas telas do sistema. As telas visualizadas devem apresentar a funcionalidade para qual foi desenvolvida de forma simples e objetiva, incluindo o conjunto de controles de navegação bem destacados.

  Este cenário garante o RNF de usabilidade.

- **Cenário 5**

  Um usuário, autenticado ou não autenticado, acessando o sistema através de um dispositivo conectado a web e navegando pelas telas do sistema. Ao navegar pelo sistema utilizando diferentes navegadores de web, sistemas operacionais e resoluções de tela, o sistema deve apresentar na tela o conjunto elementos específico da funcionalidade, de forma adequada à resolução e mantendo cores e estilos consistentes.

  Este cenário garante o RNF de acessibilidade.

- **Cenário 6**

  O sistema funcionando em carga superior ao volume dimensionado para um componente, deve permitir o escalonamento horizontal do componente afetado pelo aumento ou pela diminuição no volume de requisições de usuários.

  Este cenário garante o RNF de disponibilidade.

6.3. Avaliação

Esta seção apresenta a avaliação dos cenários identificados e tem como objetivo determinar os riscos, não riscos, pontos de sensibilidade e *trade-offs*, além disso apresenta as evidências de atendimento do requisito de qualidade.

Cenário 1:

Atributo de Qualidade:	Segurança
Requisito de Qualidade:	O acesso é protegido por mecanismo de autenticação e autorização.
Preocupação:	Acesso a recursos privados do sistema está disponível apenas com autenticação e autorização.
Cenários:	Cenário 1
Ambiente:	Produção, carga normal.

Estímulo:	Usuário tentando acessar recurso privado sem estar autenticado.
Mecanismo:	Mecanismo de geração de *tokens* contendo e-mail e permissões associadas.
Medida de Reposta:	O usuário não autenticado deve ser redirecionado para a tela de autenticação.
Considerações sobre a arquitetura:	
Riscos:	Falhas nas políticas de autenticação e autorização podem expor dados sigilosos e uso malicioso dos recursos.
Pontos de Sensibilidade:	Servidor respondendo apenas a requisições HTTPS.
Trade-off:	---

Evidências do cenário 1:

No acesso a uma página pública do sistema, neste caso a página de autenticação, não há a geração e fornecimento de um token de acesso pelo sistema.

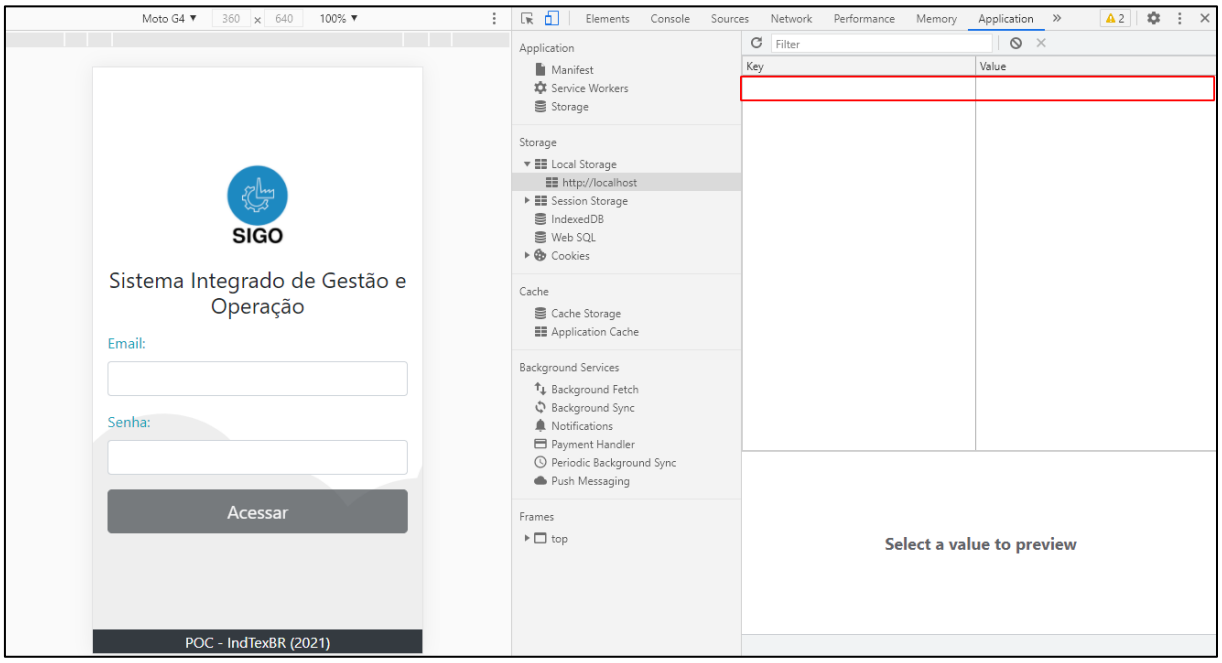

Figura 18 - Acesso a uma página pública do sistema.

Após a autenticação do usuário pelo serviço de autenticação, um token de acesso é gerado pelo sistema e armazenado pelo navegador de internet. Assim o usuário consegue acesso a uma página restrita, neste caso a página principal do módulo de Gestão do Processo Industrial.

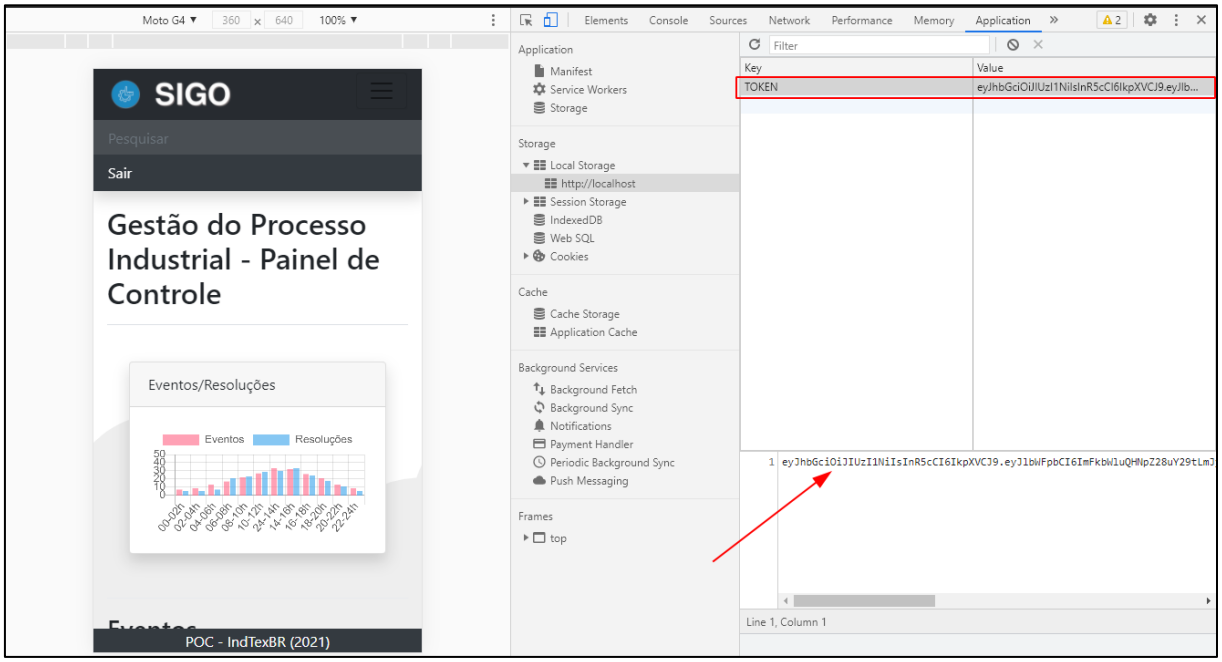

Figura 19 - Acesso a uma página restrita do sistema.

Para um usuário ainda não autenticado, ou que tenha realizado o logout do sistema, ou ainda que logado tenha sido excluído o token de acesso, caso este tente acessar algum recurso privado do sistema, o usuário será redirecionado para a tela de autenticação.

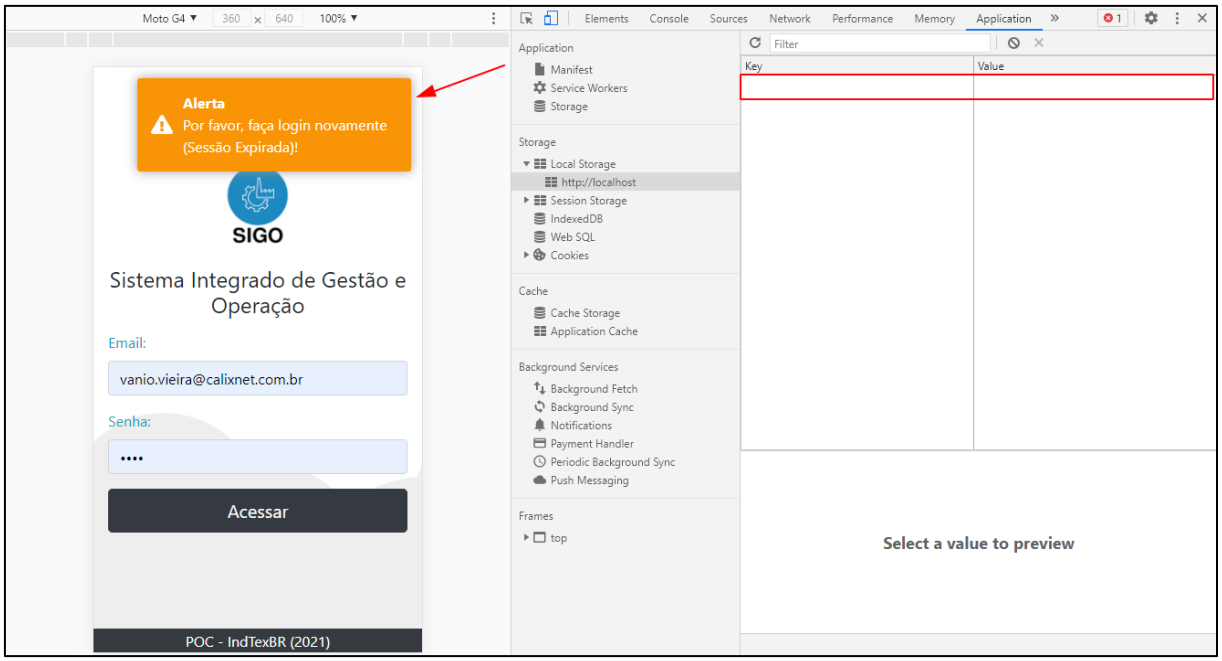

Figura 20 - Redirecionamento para a tela de autenticação após a exclusão do token.

Cenário 2:

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	O sistema deve ter integração entre seus módulos e deve se comunicar com sistemas legados.
Preocupação:	Prover integração entre os módulos e com os sistemas legados
Cenários:	Cenário 2
Ambiente:	Produção, carga normal.
Estímulo:	Módulos e sistemas legados gerando eventos e módulo de Gestão do Processo Industrial exibindo eventos recebidos.
Mecanismo:	Configuração de servidor de mensagens para integração de módulos e do ESB para integração com legados.
Medida de Reposta:	Os módulos que "assinam" determinado canal de eventos do servidor de mensagens recebem as informações.
Considerações sobre a arquitetura:	
Riscos:	Indisponibilidade do servidor de mensagens e do ESB.
Pontos de Sensibilidade:	Indisponibilidade dos servidores de integração.
Trade-off:	---

Evidências do cenário 2:

A publicação de um evento enviado por algum dos sistemas componentes da solução ou dos sistemas legados. Neste caso, foi realizada a simulação do envio do evento através do Postman.

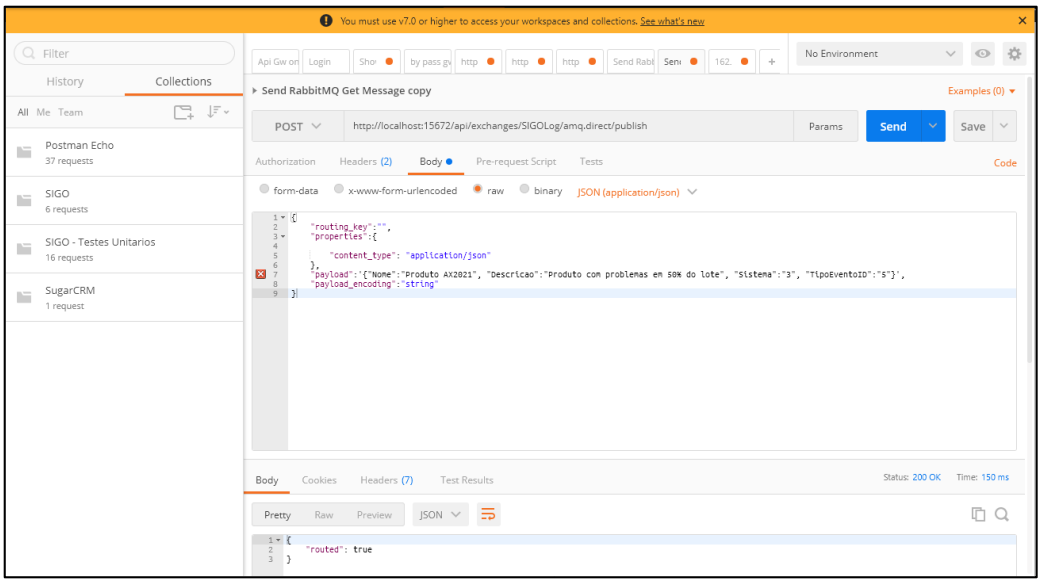

Figura 21 - Publicação de um evento enviado por algum dos sistemas legados.

A publicação da mensagem é recebida e roteada para a fila apropriada no servidor RabbitMQ.

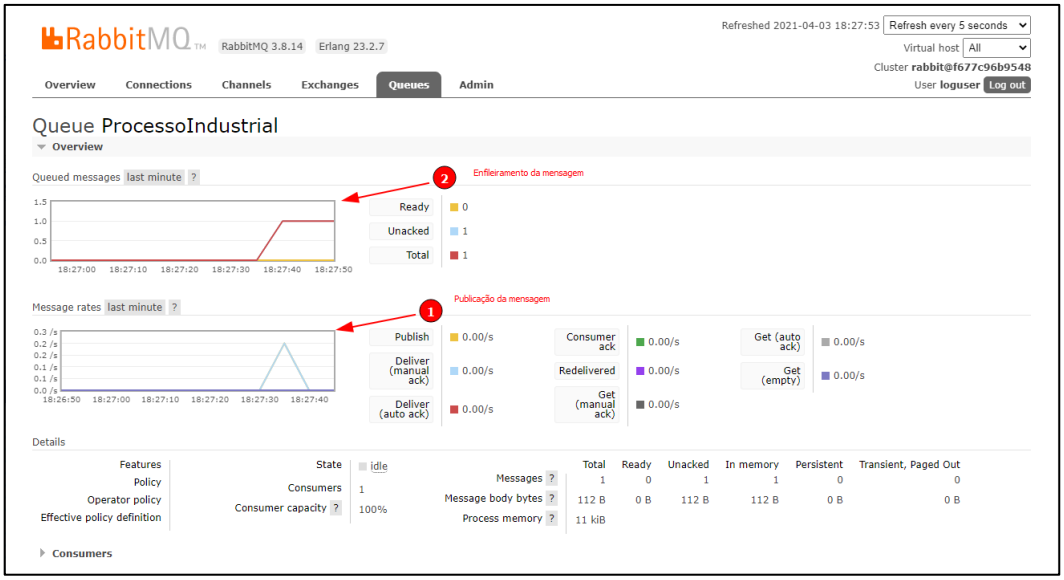

Figura 22 - Fila de mensagens do Processo Industrial no RabbitMQ.

O módulo de Gestão do Processo Industrial é assinante da fila “ProcessoIndustrial”, e tão logo a mensagem é enfileirada ocorre o recebimento de um novo evento.

Figura 23 - Situação da lista de eventos antes da publicação.

Figura 24 - Situação da lista de eventos após a publicação.

Cenário 3:

Atributo de Qualidade:	Interoperabilidade
Requisito de Qualidade:	Comunicação com serviços externos.
Preocupação:	O sistema deve se comunicar com serviços externos.
Cenários:	Cenário 3
Ambiente:	Produção, carga normal.
Estímulo:	O módulo Gestão de Normas exibe o registro de uma norma previamente cadastrada ao usuário do sistema.
Mecanismo:	O módulo de Gestão de Normas faz uma verificação da situação da norma no repositório por meio de requisições HTTP/HTTPS.
Medida de Reposta:	O sistema consegue se comunicar e obtém a situação da respectiva norma no repositório.
Considerações sobre a arquitetura:	
Riscos:	Indisponibilidade ou recusa de conexão com o servidor de repositório de normas.
Pontos de Sensibilidade:	Disponibilidade do servidor do repositório.
Trade-off:	---

Evidências do cenário 3:

O usuário seleciona um registro de norma previamente cadastrada para visualização.

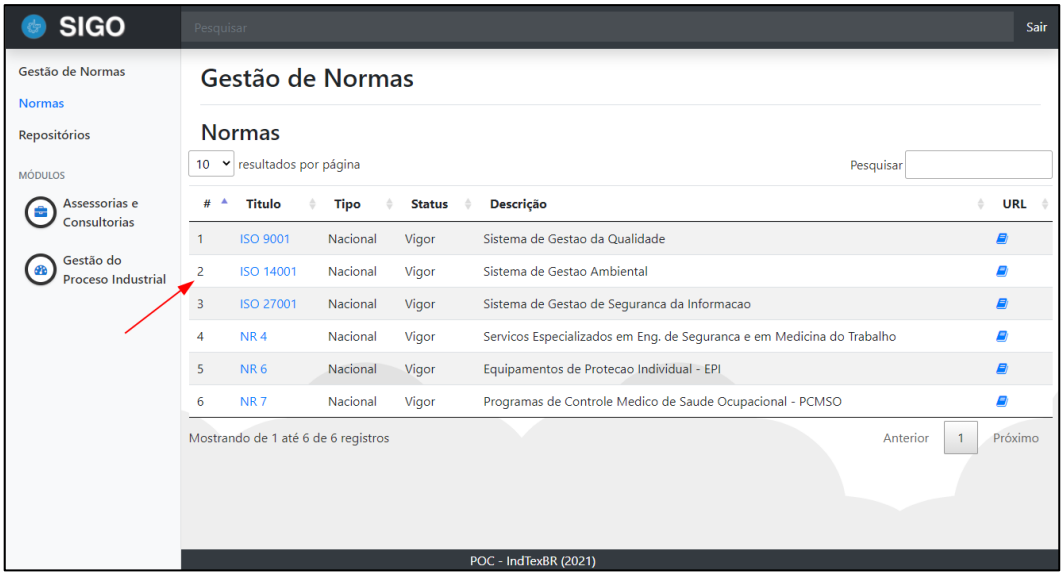

Figura 25 - Seleção do registro na listagem de normas.

O sistema obtém os dados do registro da norma e na sequência faz a verificação da situação atual da norma. Esta verificação é realizada por meio do API Gateway através *ponto de acesso "/api/Norma/Verificar"* do módulo Gestão de Normas, que faz então acesso ao repositório externo para solicitar os dados da norma.

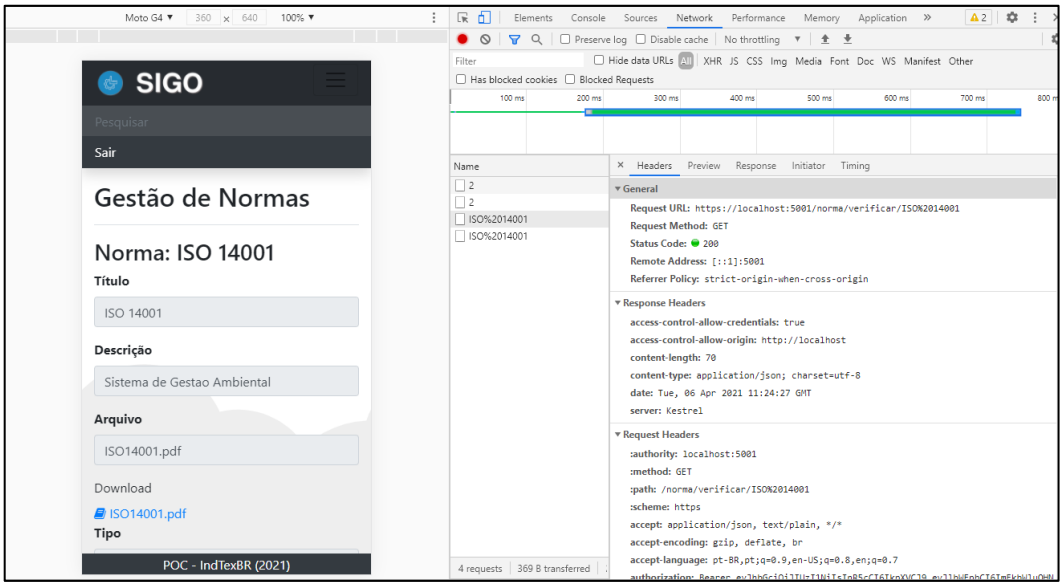

Figura 26 - Consulta aos dados da norma no repositório externo.

O sistema recebe os dados do repositório remoto, e no caso de a norma ter sido atualizada ou cancelada, realiza publicação de um evento no servidor RabbitMQ. O módulo Gestão do Processo Industrial, como assinante da fila, recebe a notificação e registra o evento relacionado.

Figura 27 - Visualização do evento no módulo Gestão do Processo Industrial.

No módulo Gestão de Normas, seguindo o fluxo iniciado pelo usuário, o registro da norma é apresentado na tela. O sistema recebe os dados do repositório remoto e apresenta uma mensagem ao usuário informado a situação atual da norma selecionada.

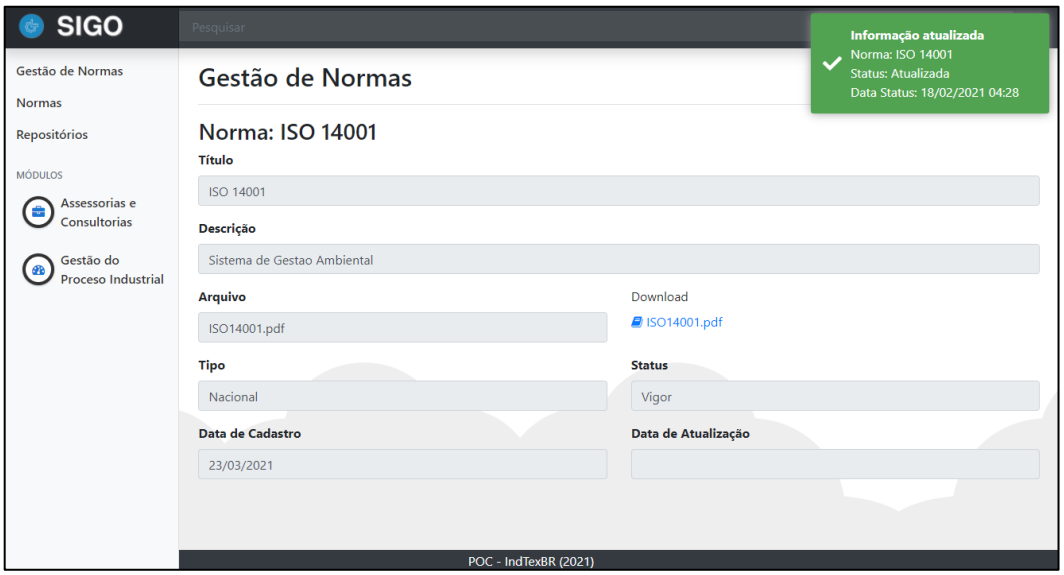

Figura 28 - Visualização do registro da norma selecionada.

Cenário 4:

Atributo de Qualidade:	Usabilidade
Requisito de Qualidade:	O sistema deve prover boa usabilidade.
Preocupação:	As telas devem apresentar funcionalidade simples e objetivas além de controles de navegação bem destacados.
Cenários:	Cenário 4
Ambiente:	Produção, carga normal.
Estímulo:	Usuário navegando pelo sistema.
Mecanismo:	As telas contendo as funcionalidades do sistema são simples e funcionais, contendo apenas elementos essenciais ao atendimento da funcionalidade.
Medida de Reposta:	O usuário deve conseguir utilizar o sistema de forma intuitiva.
Considerações sobre a arquitetura:	
Riscos:	O servidor pode apresentar instabilidade, o que pode prejudicar a experiência do usuário.
Pontos de Sensibilidade:	Alta disponibilidade.
Trade-off:	---

Evidências do cenário 4:

As telas do sistema sempre apresentam os menus de acesso das funcionalidades disponíveis no módulo (1) assim como dos módulos disponíveis (2). Esta abordagem mantem a coerência na interface e facilita a identificação do fluxo de navegação pelo o usuário.

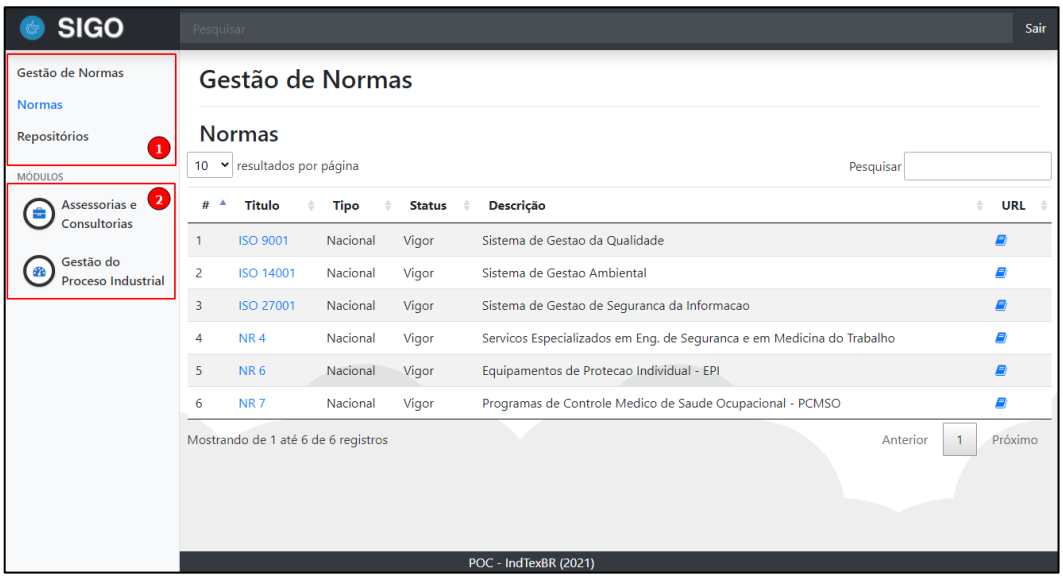

Figura 29 - Navegação no Módulo Gestão de Normas.

A interface do sistema apresenta uma padronização do layout dos elementos de tela, de cores e de fontes independente do módulo acessado.

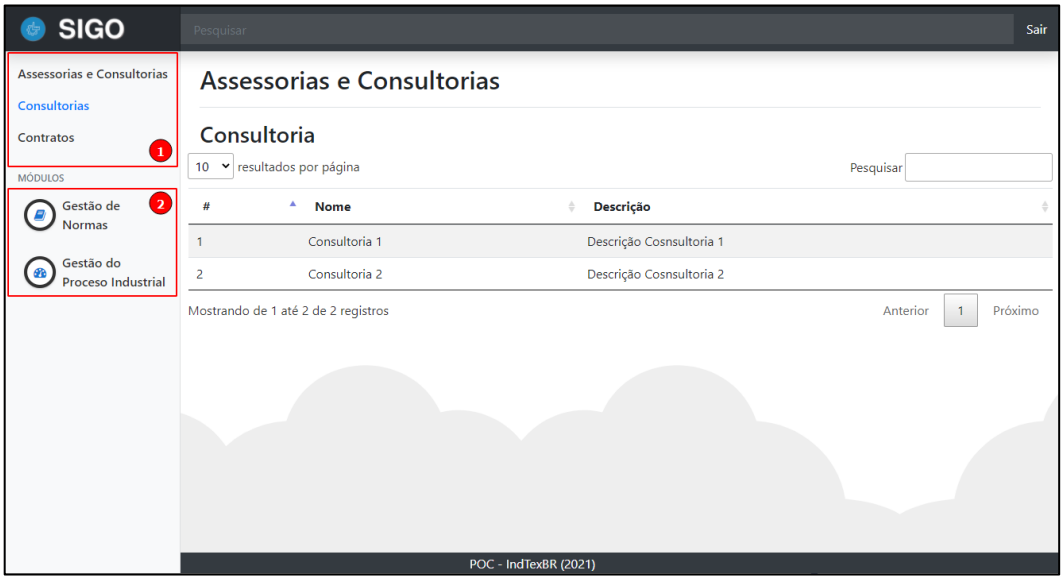

Figura 30 - Navegação no Módulo Gestão de Assessorias e Consultorias.

Cenário 5:

Atributo de Qualidade:	Acessibilidade
Requisito de Qualidade:	O sistema deve apresentar boa acessibilidade.
Preocupação:	Ao navegar pelo sistema utilizando diferentes navegadores, sistemas operacionais e resoluções de tela, o sistema deve apresentar o conjunto completo de funcionalidades, mesmo que rearranjadas, mantendo cores e estilos consistentes.
Cenários:	Cenário 5
Ambiente:	Produção, carga normal.
Estímulo:	Usuário navegando pelo sistema.
Mecanismo:	Criação de interfaces responsivas, que se adequam ao dimensionamento de tela, e com recursos suportados por diferentes navegadores.
Medida de Reposta:	O sistema deve se adaptar a diferentes ambientes do cliente sem perder funcionalidades e sem prejudicar a usabilidade das interfaces.
Considerações sobre a arquitetura:	
Riscos:	Existência de muitos navegadores e de muitas versões de cada um desses navegadores.
Pontos de Sensibilidade:	Compatibilidade com navegadores.
Trade-off:	---

Evidências do cenário 5:

Um usuário acessa uma página do SIGO a partir de um dispositivo *desktop*.

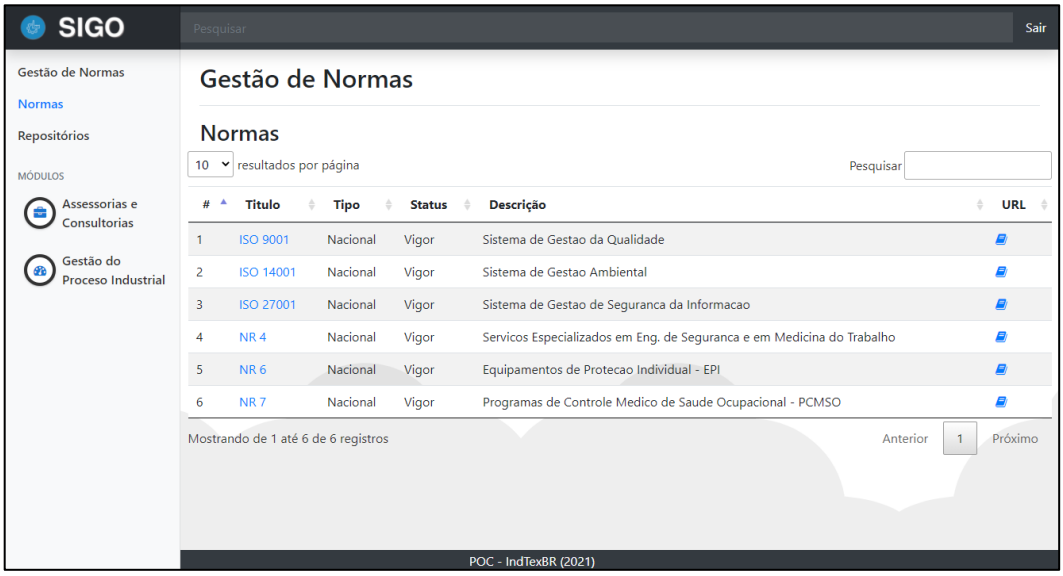

Figura 31 - Acesso a Listagem de Normas cadastradas (dispositivo desktop).

Um usuário acessa uma página do SIGO a partir de um dispositivo *mobile* que possui dimensões reduzidas de tela.

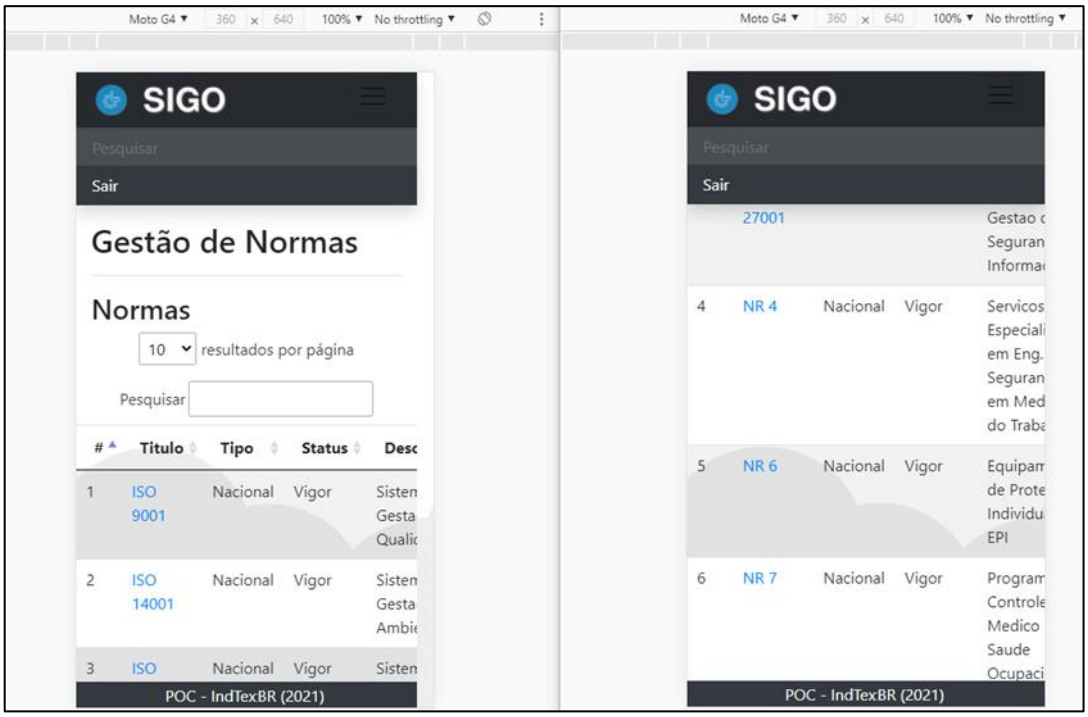

Figura 32 - Acesso a Listagem de Normas cadastradas (dispositivo mobile).

Cenário 6:

Atributo de Qualidade:	Disponibilidade
Requisito de Qualidade:	O sistema deve estar disponível 24 horas por dia nos sete dias por semana.
Preocupação:	Garantir que a aplicação seja capaz de tratar grandes volumes de requisições, permitindo que recursos sejam liberados tão logo o volume diminua.
Cenários:	Cenário 6
Ambiente:	Produção, carga acima do normal, com muitos usuários realizando operações de alto custo operacional.
Estímulo:	Aumento do volume de requisições.
Mecanismo:	Implementação de uma arquitetura *stateless*, com execução em ambiente de containers, e utilização de ferramentas de orquestração.
Medida de Reposta:	A aplicação permite o fácil escalonamento horizontal dos componentes de arquitetura para tratar o aumento de demanda.
Considerações sobre a arquitetura:	
Riscos:	Se a aplicação não for capaz de tratar volumes variáveis de requisições há o risco de indisponibilidade temporária, podendo gerar descrédito por parte dos usuários e consequentemente entrar em desuso.
Pontos de Sensibilidade:	---
Trade-off:	---

Evidências do cenário 6:

Na ocorrência de aumento do volume de requisições, há possibilidade realizar o escalonamento horizontal para atendimento da demanda.

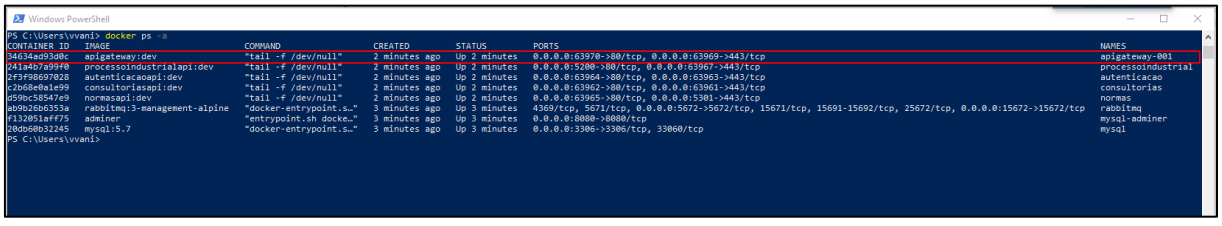

Figura 33 - Apenas uma instância do API Gateway em execução.

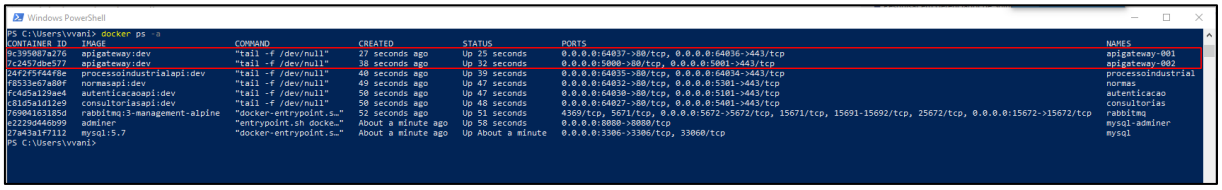

Figura 34 - Duas instâncias do API Gateway em execução.

6.4. Resultado

Tomando por base o desenvolvimento da prova de conceito e tendo realizado a validação da arquitetura baseada nos requisitos de qualidade determinados para a plataforma, é possível afirmar que os requisitos demandados foram atendidos.

O quadro a seguir apresenta os requisitos não funcionais validados:

Requisito não funcional		**Testado**	**Homologado**
Segurança	Acesso a recursos privados com autenticação.	Sim	Sim
Interoperabilidade	O sistema deve prover comunicação entre módulos e sistemas legados.	Sim	Sim
Usabilidade	O sistema deve apresentar funcionalidades simples e objetivas e navegação intuitiva.	Sim	Sim
Acessibilidade	O sistema deve funcionar de forma consistente em diferentes telas, resoluções, sistemas operacionais e navegadores.	Sim	Sim
Disponibilidade	O sistema deve estar disponível 24 horas por dia, nos sete dias da semana.	Sim	Sim

Considerando os cenários escolhidos para validação dos respectivos atributos de qualidade e as evidências produzidas durante o processo de avaliação, a convalidação da proposta arquitetural foi exitosa na medida em que esta atende às necessidades identificadas e aos requisitos especificados para o projeto.

Cabe destacar que a proposta arquitetural especificada neste documento está completa quanto ao atendimento aos requisitos especificados. Entretanto a proposta não está fechada, pois há ainda na plataforma abertura para possibilitar a adição de funcionalidades, bem como para a realização de melhorias e aprimoramentos futuros conforme necessidades resultantes da evolução tecnológica ou do negócio.

Uma das melhorias possíveis e prováveis para o projeto seria a agregação e centralização dos registros de logs de todos os componentes, novos e legados, por meio de um sistema específico para esta finalidade. O Logstash é uma das ferramentas indicadas para este caso. O Logstashé parte da pilha **ELK**, composta também pelo Elasticsearch e o Kibana. O Logstash é um pipeline de dados que ajuda a processar logs a partir de uma variedade de sistemas e uma variedade de fontes, além de possibilitar a criação de sistema de análise central altamente escalável. O Elasticsearch é uma ferramenta de indexação textual altamente difundida, e o Kibana permite a criação de gráficos a partir de dados indexados no Elasticsearch. A adição destes componentes ao projeto SIGO resultaria em ganhos consideráveis na administração do sistema como um todo.

7. Conclusão

Este projeto define os princípios arquiteturais de software que serão utilizados na construção de uma solução customizada baseada nas necessidades do negócio, fazendo uso de serviços e recursos tecnológicos de software e hardware já existentes na empresa, bem como o desenvolvimento de novos componentes e serviços, visando atender ao projeto de restruturação demandado pela mudança na estratégia organizacional. Como resultado do trabalho, foi projetada uma arquitetura de software distribuído baseada em microsserviços contemplando um conjunto de módulos desenvolvidos, mecanismos de integração, componentes e mecanismos de comunicação, em conjunto com boas práticas de engenharia de software.

A arquitetura baseada em microsserviços mostra-se plenamente capaz de satisfazer os requisitos funcionais e não funcionais definidos para o projeto de transformação digital da IndTexBr. Utilizando esta arquitetura, se implementado, o projeto SIGO se constituirá numa plataforma robusta e moderna, que se manterá viável a longo prazo, e que possibilitará a constante evolução e adaptação às necessidades do negócio. A plataforma SIGO como um todo, representada pelos componentes arquiteturais desta prova de conceito, apresenta o comportamento esperado na medida em que possibilita a integração do novo sistema como os sistemas existentes e viabiliza o rápido e fácil escalonamento horizontal dos principais componentes arquiteturais.

Por se tratar de um projeto com escopo complexo, foi possível evidenciar as dificuldades do tema e a importância do desenvolvimento de uma especificação arquitetural que atenda aos requisitos sem, contudo, se prender a apenas uma tecnologia ou fornecedor. No cenário atual, marcado pela constante evolução tecnológica e pela necessidade de interoperabilidade entre sistemas e tecnologias, quanto maior o leque de opções viáveis mais assertiva será a arquitetura do sistema.

REFERÊNCIAS

DOCKER Documentation. **Docker**, 2020. Disponível em: < https://docs.docker.com/ >. Acesso em: 03 de jan. de 2021.

LARMAN, Graig. **Utilizando UML e Padrões: Uma introdução a análise e ao projeto orientados a objetos**. 3ª Edição. Porto Alegre: Bookman, 2011.

MARTIN, Robert C. **Arquitetura Limpa: O guia do Artesão para Estrutura e Design de Software**. 1ª Edição. Rio de Janeiro: Alta Books, 2018.

MICROSSERVIÇOS do .NET: Arquitetura de aplicativos .NET em contêineres. **Microsoft**, 2020. Disponível em: < https://docs.microsoft.com/pt-br/dotnet/architecture/microservices/ >. Acesso em: 17 de jan. de 2021.

OPENAPI Specification. **Swagger**, 2020. Disponível em: < https://swagger.io/specification/ >. Acesso em: 24 de jan. de 2021.

RICHARDSON, Chris. Pattern: Microservice Architecture. **Microservice Architecture**, 2018. Disponível em: < http://microservices.io/patterns/microservices.html >. Acesso em: 31 de jan. de 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 9ª edição. São Paulo: Pearson, 2011.

UNDERSTANDING Enterprise Application Integration - The benefits of ESB for EAI. **Mulesoft**, 2020. Disponível em: < https://www.mulesoft.com/resources/esb/enterprise-application-integration-eai-and-esb/ >. Acesso em: 07 de fev. de 2021.

APÊNDICES

Repositório de Código. Disponível em < https://github.com/puc-vvanio/Projeto_SIGO >.

Vídeo de Apresentação da POC. Disponível em < https://youtu.be/noH0no7J9Vc >.