
Documentação de Projeto

para o sistema

API Valve

Versão 2.4

Projeto de sistema elaborado pelo aluno Marlon Henrique da Silva e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor Marco Rodrigo Costa, orientação acadêmica do professor José Laerte Xavier e orientação de TCC II do professor José Laerte Xavier.

13/11/2022

Tabela de Conteúdo

1. Introdução	4
2. Modelos de Usuário e Requisitos	5
2.1 Descrição de Atores	5
2.2 Modelos de Usuários	5
2.3 Modelo de Casos de Uso e Histórias de Usuários	7
2.4 Diagrama de Sequência do Sistema e Contrato de Operações	8
3. Modelos de Projeto	14
3.1 Diagrama de Classes	14
3.2 Diagramas de Sequência	16
3.3 Diagramas de Comunicação	23
3.4 Arquitetura	24
3.5 Diagramas de Estados	25
3.6 Diagrama de Componentes e Implantação	27
4. Projeto de Interface com Usuário	28
5. Glossário e Modelos de Dados	32
6. Casos de Teste	35
6.1 Testes de Aceitação	35
6.2 Testes de Integração	41
7. Cronograma e Processo de Implementação	42
8. Post Mortem	45
8.1 Experiências Positivas	45
8.2 Experiências Negativas	45
8.3 Lições Aprendidas	46
8.4 Repositório do Projeto	46

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Criação	08/03/2022	Criação do Documento	1.0
Modificações	17/03/2022	Adição do diagrama de caso de uso e histórias de usuários	1.1
Modificações	22/03/2022	Adição dos modelos de usuário e um projeto de interface	1.2
Modificações	27/03/2022	Adição dos projetos de interface completos	1.3
Modificações	10/04/2022	Adição dos diagramas de classe, sequência e comunicação	1.4
Modificações	24/04/2022	Adição dos diagramas de arquitetura lógica, estados, componentes, implantação, modelos de dados e glossário.	1.5
Modificações	08/05/2022	Adição dos casos de teste e cronograma.	1.6
Modificações	17/05/2022	Correções gerais e formatação.	1.7
Modificações	07/06/2022	Aplicação das sugestões da pré-banca de TCC I.	1.8
Modificações	11/08/2022	Atualização do cronograma para TCC II.	2.0
Modificações	17/09/2022	Adição da categoria <i>Union</i> para <i>schemas</i> .	2.1
Modificações	29/10/2022	Adição da seção de <i>Post Mortem</i> .	2.2
Modificações	03/11/2022	Alteração na seção de Introdução.	2.3
Modificações	13/11/2022	Revisão geral e ajustes.	2.4

1. Introdução

Este documento contempla os modelos de domínio e modelos de projeto para o sistema **API Valve** — uma ferramenta *low code* para construção de aplicações *Back-end for Front-end* (BFF). Esta é a referência principal para a descrição geral do problema, domínio e requisitos do sistema por meio de especificações que descrevem a visão do projeto.

O conceito de BFF, apresentado por Phil Calçado (2015), propõe um padrão de arquitetura de software para melhorar a experiência de desenvolvimento de aplicativos móveis e distribuídos. Cada BFF pode estar acoplado a uma interface de usuário específica e, normalmente, ambos são mantidos pela mesma equipe, facilitando a definição e adaptação da Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*), conforme as necessidades da aplicação. Em um BFF, é comum definir estruturas específicas para cada interface de usuário, que não correspondem, necessariamente, a uma entidade do negócio. É possível construir a estrutura de dados para uma tela de perfil de usuário, por exemplo, que agrega informações de diversas fontes e pode variar para cada tipo de aplicação.

O sistema proposto é uma ferramenta de apoio para a construção de uma API, que utiliza GraphQL por padrão e realiza o envio, recebimento e agregação de dados a partir de outras aplicações. Por meio de especificações, é possível definir o seu funcionamento utilizando uma Notação de Dados Extensível (EDN, do inglês *Extensible Data Notation*) ou uma interface de usuário. As especificações habilitam o recebimento de requisições via Protocolo de Transferência de Hipertexto (HTTP, do inglês *Hypertext Transfer Protocol*) no formato GraphQL e a realização de requisições HTTP no padrão de Transferência de Estado Representacional (REST, do inglês *Representational State Transfer*).

A principal motivação para se construir um BFF ocorre em cenários em que múltiplos serviços de *back-end* fornecem dados para diferentes aplicações *front-end*, ocasionando em *endpoints* que retornam dados em excesso e que geram um consumo maior de banda de Internet dos usuários. Por outro lado, *endpoints* mais específicos podem ocasionar em excessos de requisições para construir visualizações mais completas. Utilizar um BFF com GraphQL pode simplificar o código e melhorar o desempenho. Em vez de fazer várias chamadas diferentes para múltiplos serviços e/ou consumir mais dados do que o necessário, o cliente pode solicitar um único recurso que já tenha as informações que ele precisa de forma agregada, permitindo que ele especifique exatamente quais delas deseja obter, podendo, inclusive, consultar mais de uma fonte com uma única requisição. O objetivo de um BFF não é ser uma API independente. O BFF é um complemento do aplicativo, funcionando como um agregador e filtro das informações que cada aplicativo necessita para construir as suas interfaces.

O sistema em questão pode ser definido como uma ferramenta *low code*, pois não propõe o desenvolvimento de código-fonte para a implementação das funcionalidades da aplicação por parte da pessoa desenvolvedora. É possível que isso seja feito de forma visual, utilizando uma interface gráfica interativa ou escrevendo diretamente em um arquivo EDN. Apenas com essas especificações e com o executável já fornecido pelo sistema, onde todas as funcionalidades são implementadas de forma genérica, é possível obter uma aplicação que recebe e realiza requisições de forma agregada. Ademais, a aplicação pode ser executada no próprio ambiente de desenvolvimento ou implantada em um servidor que utilize um sistema operacional Windows, Linux ou macOS com suporte à JVM.

2. Modelos de Usuário e Requisitos


Nesta seção, são apresentados os modelos de usuários e requisitos do sistema. A apresentação é feita pela descrição dos atores na Seção 2.1, a descrição dos modelos de usuários na Seção 2.2, a descrição de casos de uso e de histórias de usuários na Seção 2.3 e, por fim, a descrição dos diagramas de sequência e contratos de operações na Seção 2.4.

2.1 Descrição de Atores

Os atores identificados para a utilização do sistema são pessoas desenvolvedoras de *software*, possivelmente com qualificações em *back-end*, *front-end* e *mobile*. As qualificações dos usuários podem ser diversas, pois as ações não demandam conhecimento específico e aprofundado em uma determinada tecnologia, mas é necessário que entendam da ferramenta e das suas próprias necessidades. As interações dos usuários com o sistema são insumos que habilitam as suas funcionalidades e possibilitam a geração de uma aplicação BFF.

2.2 Modelos de Usuários

Para este projeto, os usuários foram modelados por meio de personas, conforme a definição de Courage e Baxter (2005), abordando os seguintes tópicos: identidade, status, objetivos, habilidades e tarefas. As personas apresentadas nas Tabelas 1, 2 e 3 são fictícias, criadas a fim de exemplificar a abrangência do público alvo e de possíveis usuários do sistema.

 <p>Fonte: freepik. Imagem livre de direitos autorais.</p>	Camila Alves, 30 anos, desenvolvedora <i>back-end</i> . Ela trabalha desenvolvendo e mantendo microsserviços de uma empresa de serviços digitais presente em diversas plataformas.
Status	Primária
Objetivos	Aprimorar seus conhecimentos sobre sistemas distribuídos, escalabilidade e arquitetura de <i>software</i> para contribuir mais em seu trabalho e assumir uma posição de liderança técnica.
Habilidades	Construção de APIs, padrão REST,

	WebSockets, Kafka, GraphQL, bancos de dados, monitoramento de sistemas.
Tarefas	Desenvolver novas funcionalidades para as aplicações já existentes em seu trabalho; revisão e refatoração de códigos; programação em pares; auxiliar novos membros da equipe; sugerir soluções e melhorias para as aplicações do seu time.

Tabela 1. Persona criada para representar uma pessoa desenvolvedora *back-end*.


 <p>Fonte: freepik. Imagem livre de direitos autorais.</p>	Murilo Souza, 28 anos, desenvolvedor <i>front-end</i> . Ele trabalha com desenvolvimento <i>front-end</i> há 10 anos. Se formou em um curso técnico de TI enquanto cursava o ensino médio e, em seguida, ingressou em um curso de graduação em Sistemas de Informação.
Status	Primário
Objetivos	Encontrar e/ou criar ferramentas para facilitar o desenvolvimento <i>front-end</i> ; descobrir novas maneiras de resolver problemas comuns no desenvolvimento Web; aprender mais sobre sistemas distribuídos e seus impactos em aplicações Web complexas.
Habilidades	Construção de aplicações <i>front-end</i> ; HTML (<i>HyperText Markup Language</i>); CSS (<i>Cascading Style Sheet</i>); JavaScript; integração com APIs REST e GraphQL; otimização de aplicações Web.
Tarefas	Liderança técnica de <i>front-end</i> Web; revisão e refatoração de código; mentoria de novos colaboradores; pesquisa de novas tecnologias e ferramentas para serem integradas ao processo de desenvolvimento das equipes.

Tabela 2. Persona criada para representar uma pessoa desenvolvedora *front-end*.


 Fonte: freepik. Imagem livre de direitos autorais.	Juliana Ferreira, 23 anos, desenvolvedora <i>mobile</i> . Recém-formada em um curso técnico de programação e desenvolvimento de aplicações móveis. Atua no mercado de trabalho há 3 anos realizando estágios remunerados, mas foi contratada há pouco mais de 8 meses pela empresa atual.
Status	Secundária
Objetivos	Expandir e aprofundar seus conhecimentos sobre desenvolvimento <i>mobile</i> , Interface de Usuário/Experiência de Usuário (UI/UX, do inglês <i>User Interface/User Experience</i>), consumo de APIs com REST e GraphQL.
Habilidades	Desenvolvimento de aplicativos para multiplataformas com React Native e Flutter; desenvolvimento Web com React JS; <i>design</i> e prototipação de interfaces com Figma.
Tarefas	Implementar novas funcionalidades e aprimorar a usabilidade do aplicativo da empresa em que trabalha, disponível para Android e iOS. Uma nova versão deve ser lançada semanalmente com correções e/ou novidades.

Tabela 3. Persona criada para representar uma pessoa desenvolvedora *mobile*.

2.3 Modelo de Casos de Uso e Histórias de Usuários

Nesta seção, estão listadas as histórias de usuário definidas para o sistema. Para fins de organização, as histórias se encontram numeradas de 1 a 8, rotuladas com a letra H (de história). Os números não significam ordem de prioridade e/ou de importância.

- H01: Camila, como desenvolvedora *back-end*, deseja abstrair o versionamentos das APIs para os clientes;
- H02: Pedro, como desenvolvedor *front-end*, deseja abstrair as fontes de dados da aplicação;
- H03: Murilo, como desenvolvedor *front-end*, deseja uma maneira simples de agregar os dados de diversas fontes para usos específicos do sistema;

- H04: Paulo, como desenvolvedor *mobile*, deseja uma maneira simples de consumir dados por demanda para um aplicativo;
- H05: Juliana, como desenvolvedora *back-end*, deseja uma maneira simples de criar uma API GraphQL;
- H06: Daniel, como desenvolvedor *back-end*, deseja automatizar e padronizar a criação de aplicações para o mesmo propósito, mas com diferentes contextos;
- H07: Giovana, como desenvolvedora *back-end*, deseja definir novos recursos em uma API apenas especificando os dados;
- H08: Walter, como desenvolvedor *back-end*, deseja gerenciar e testar APIs de forma visual, utilizando interfaces dedicadas.

Na Figura 1, estão modelados os casos de uso que representam as ações dos usuários durante a utilização do sistema. Para fins de organização, os casos de uso se encontram numerados de 1 a 8, rotulados com a letra C (de caso). Os números não significam ordem de prioridade e/ou de importância.

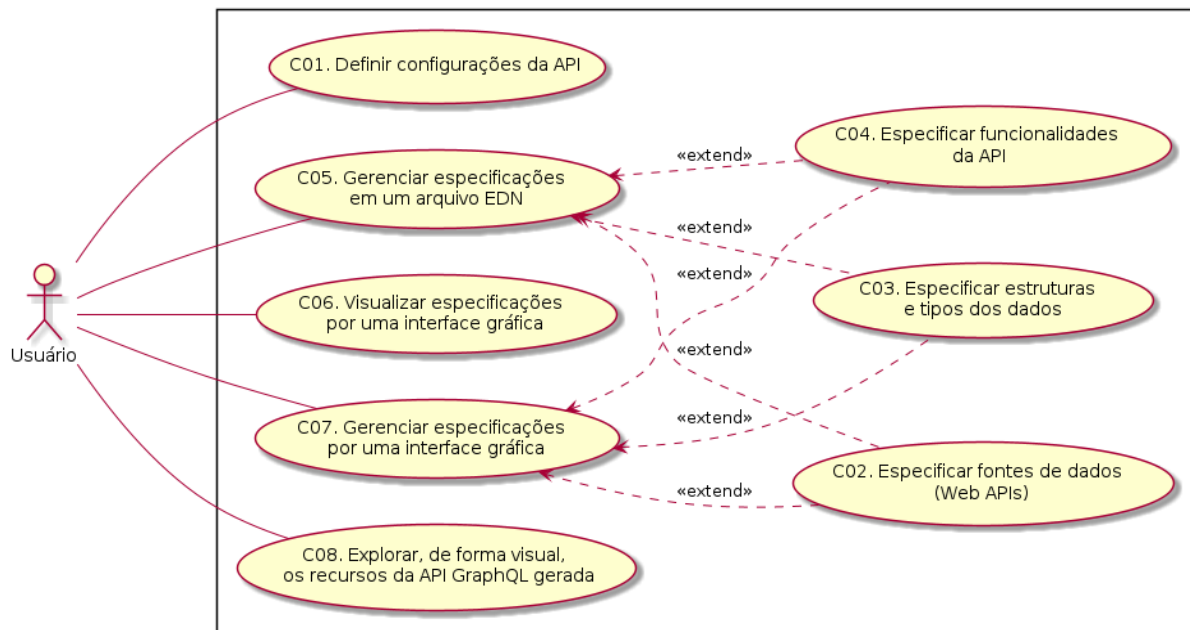


Figura 1. Diagrama de casos de uso do sistema.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Nesta seção, são apresentados os diagramas de sequência do sistema com seus respectivos contratos de operações, baseados no diagrama de casos de uso apresentado anteriormente. No entanto, como os casos de uso C02, C03 e C04 são extensões dos casos de uso C05 e C07, eles estão representados individualmente.

Na Figura 2, está representado o fluxo de definição das configurações da aplicação, que consiste, basicamente, em exibir, alterar e salvar essas informações no arquivo EDN. Esse diagrama contempla o caso de uso C01. Em seguida, são apresentados os contratos de operações

para cada uma das funções representadas no diagrama. A Tabela 4 representa a função de exibir configurações, a Tabela 5 representa a função de editar configurações e, por fim, a Tabela 6 representa a função de salvar configurações.

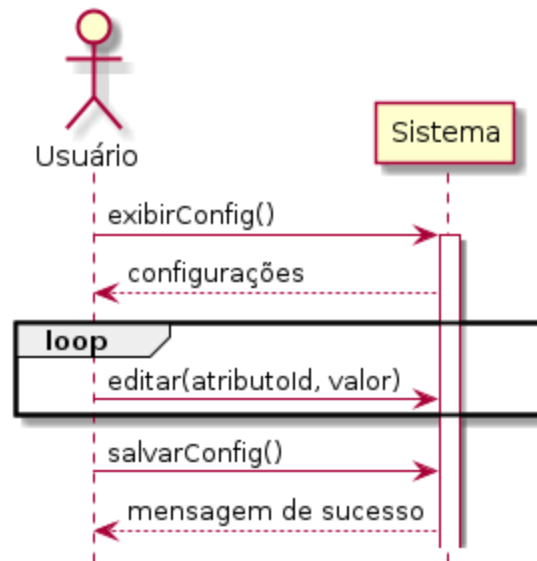


Figura 2. Diagrama de sequência do sistema para definir configurações da aplicação.

Contrato	exibirConfig()
Operação	exibirConfig()
Referências cruzadas	Definir configurações da API.
Pré-condições	Configurações definidas no arquivo EDN.
Pós-condições	O arquivo foi carregado.

Tabela 4. Contrato de operação para exibir configurações.

Contrato	editar()
Operação	editar(atributoId, valor)
Referências cruzadas	Definir configurações da API
Pré-condições	As configurações foram carregadas.
Pós-condições	As informações foram alteradas na interface.

Tabela 5. Contrato de operação para editar configurações.

Contrato	salvarConfig()
Operação	salvarConfig()
Referências cruzadas	Definir configurações da API.
Pré-condições	Todos os campos obrigatórios foram preenchidos.
Pós-condições	As alterações foram salvas no arquivo EDN.

Tabela 6. Contrato de operação para salvar configurações.

Na Figura 3, está representado o fluxo de gerenciamento das especificações diretamente pelo arquivo EDN. Nesse caso, não há muitas ações do sistema, pois o fluxo é baseado na interação direta do usuário com um arquivo de texto. No entanto, as informações definidas no arquivo referem-se ao sistema e são utilizadas por ele para o seu devido funcionamento, seguindo algumas regras. Esse diagrama contempla diretamente o caso de uso C05 e indiretamente os casos de uso C02, C03 e C04. Em seguida, na Tabela 7, é apresentado o contrato de operações para a função representada no diagrama referente à execução do sistema.

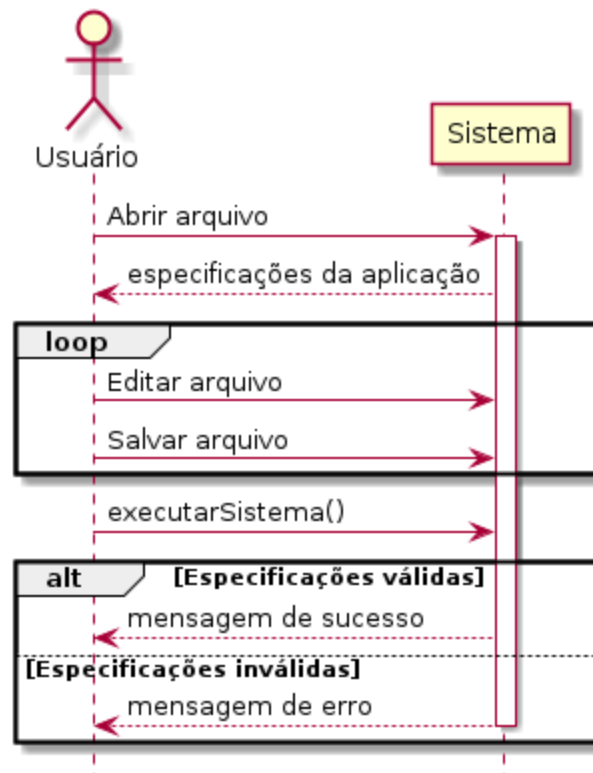


Figura 3. Diagrama de sequência do sistema para manipulação do arquivo EDN.

Contrato	executarSistema()
Operação	executarSistema()
Referências cruzadas	<ul style="list-style-type: none"> • Gerenciar especificações em um arquivo EDN; • Especificar fontes de dados (Web APIs); • Especificar estruturas e tipos dos dados; • Especificar funcionalidades da API.
Pré-condições	Arquivo EDN dentro do diretório do sistema.
Pós-condições	<ul style="list-style-type: none"> • As alterações foram salvas no arquivo EDN; • Os dados do arquivo foram validados pelo sistema.

Tabela 7. Contrato de operação para executar o sistema.

As outras especificações da aplicação armazenadas no arquivo EDN, que não são relacionadas às configurações, são estruturas pré-definidas que definem partes específicas da aplicação. Essas partes são chamadas de componentes e o gerenciamento deles é padronizado

para os diferentes tipos (*Schemas*, *Queries*, *Mutations* e APIs), mudando apenas os atributos que cada um possui. A aplicação é composta pela composição desses componentes.

Na Figura 4, está representado o fluxo de exibição de componentes que ocorre por meio da interface de usuário do sistema, utilizando os dados armazenados no arquivo EDN que há em seu diretório. Esse diagrama contempla o caso de uso C06. Em seguida, na Tabela 8, é apresentado o contrato de operações para a função representada no diagrama referente a exibição de componentes.

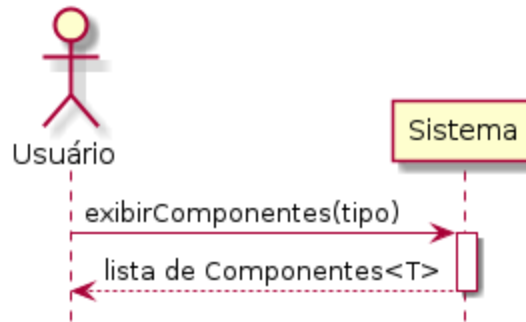


Figura 4. Diagrama de sequência do sistema para exibição dos componentes.

Contrato	exibirComponentes()
Operação	exibirComponentes(tipo)
Referências cruzadas	Visualizar especificações por uma interface gráfica.
Pré-condições	Componentes definidos no arquivo EDN.
Pós-condições	O arquivo foi carregado.

Tabela 8. Contrato de operação para exibir componentes.

Na Figura 5, está representado o fluxo de gerenciamento das especificações que ocorre por meio da interface de usuário do sistema. Esse fluxo está relacionado ao caso de uso C05, mas consiste na intermediação do sistema para o gerenciamento dos dados armazenados no arquivo EDN. Esse diagrama contempla diretamente o caso de uso C07 e indiretamente os casos de uso C02, C03 e C04. Em seguida, são apresentados os contratos de operações para cada uma das funções representadas no diagrama. A Tabela 9 representa a função de criar um componente; a Tabela 10 representa a função de selecionar um componente; a Tabela 11 representa a função de editar um componente; a Tabela 12 representa a função de salvar um componente e, por fim, a Tabela 13 representa a função de excluir um componente.

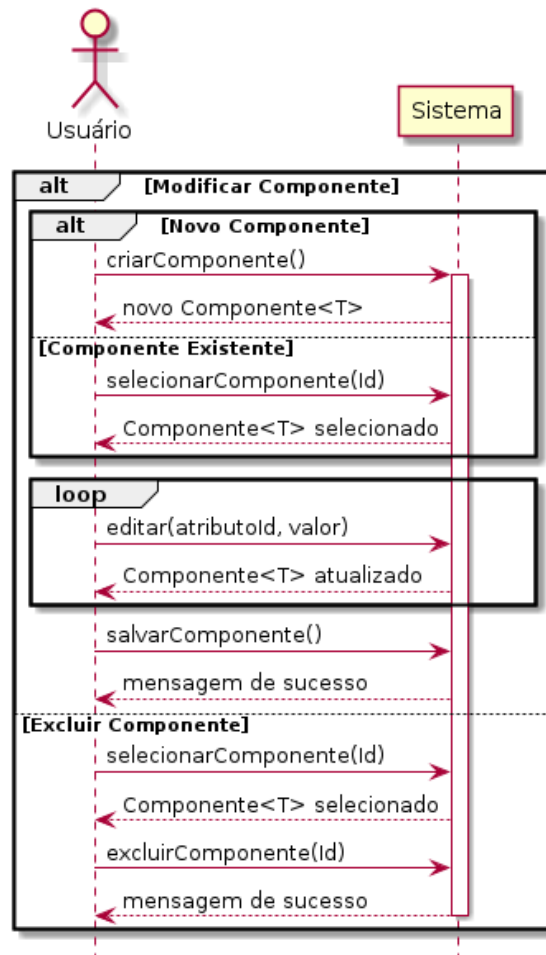


Figura 5. Diagrama de sequência do sistema para gerenciamento de componentes.

Contrato	criarComponente()
Operação	criarComponente()
Referências cruzadas	Gerenciar especificações por uma interface gráfica.
Pré-condições	Tipo de componente selecionado pela listagem.
Pós-condições	Um novo componente foi instanciado.

Tabela 9. Contrato de operação para criar um componente.

Contrato	selecionarComponente()
Operação	selecionarComponente(Id)
Referências cruzadas	Gerenciar especificações por uma interface gráfica.
Pré-condições	Componente já existente.
Pós-condições	O componente foi carregado na interface.

Tabela 10. Contrato de operação para selecionar um componente.

Contrato	editar()
Operação	editar(atributoId, valor)

Referências cruzadas	<ul style="list-style-type: none"> • Gerenciar especificações por uma interface gráfica; • Especificar fontes de dados (Web APIs); • Especificar estruturas e tipos dos dados; • Especificar funcionalidades da API.
Pré-condições	Um componente instanciado.
Pós-condições	As informações foram alteradas na interface.

Tabela 11. Contrato de operação para editar um componente.

Contrato	salvarComponente()
Operação	salvarComponente()
Referências cruzadas	Gerenciar especificações por uma interface gráfica.
Pré-condições	Todos os campos obrigatórios foram preenchidos.
Pós-condições	As alterações foram salvas no arquivo EDN.

Tabela 12. Contrato de operação para salvar um componente.

Contrato	excluirComponente()
Operação	excluirComponente(Id)
Referências cruzadas	Gerenciar especificações por uma interface gráfica.
Pré-condições	Componente já existente.
Pós-condições	O componente foi removido do arquivo EDN.

Tabela 13. Contrato de operação para excluir um componente.

Na Figura 6, está representado o fluxo de exploração da aplicação que ocorre por meio da interface gráfica GraphiQL incorporada ao sistema para visualizar detalhes da API e realizar requisições de forma manual. Esse diagrama contempla o caso de uso C08. Em seguida, são apresentados os contratos de operações para cada uma das funções representadas no diagrama. A Tabela 14 representa a função de exibir a documentação da API, enquanto a Tabela 15 representa a função de realizar uma requisição para a API.

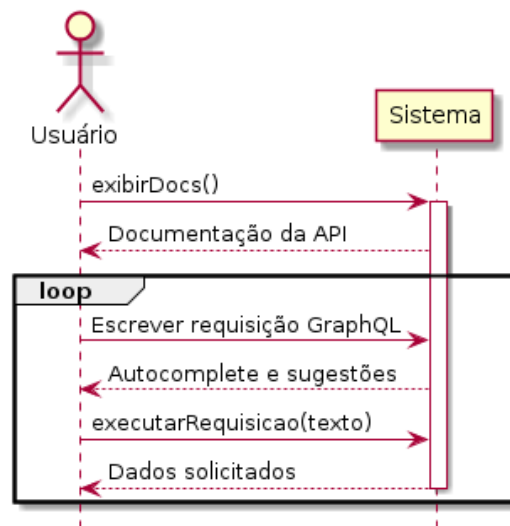


Figura 6. Diagrama de sequência do sistema para exploração da aplicação.

Contrato	exibirDocs()
Operação	exibirDocs(Id)
Referências cruzadas	Explorar, de forma visual, os recursos da API GraphQL gerada.
Pré-condições	Especificações definidas no arquivo EDN.
Pós-condições	As informações foram carregadas do arquivo.

Tabela 14. Contrato de operação para exibir a documentação da API.

Contrato	executarRequisicao()
Operação	executarRequisicao(texto)
Referências cruzadas	Explorar, de forma visual, os recursos da API GraphQL gerada.
Pré-condições	<ul style="list-style-type: none"> • <i>Query</i> ou <i>Mutation</i> escrita na interface; • Formato de requisição válido.
Pós-condições	Os dados solicitados foram retornados.

Tabela 15. Contrato de operação para realizar uma requisição na API.

3. Modelos de Projeto

Nesta seção, são apresentados os modelos de projeto do sistema proposto. A seção é composta pelo diagrama de Classes apresentado na Seção 3.1, os diagramas de sequência apresentados na Seção 3.2, os diagramas de comunicação na Seção 3.3, a arquitetura lógica apresentada na Seção 3.4, o diagramas de estado apresentado na Seção 3.5 e, por fim, os diagramas de componentes e implantação na Seção 3.6.

3.1 Diagrama de Classes

Na Figura 7, está representado o diagrama de classes do sistema, com todas as entidades que são gerenciadas por ele. As entidades são equivalentes aos componentes citados anteriormente. São eles: *Type*, *Input*, *Interface*, *Union*, *Enum*, *Query*, *Mutation* e API. Alguns desses componentes possuem estruturas compostas e, por isso, se relacionam com entidades secundárias como: *Arg*, *Field* e *Value*. Essas entidades representam as especificações que são definidas no arquivo EDN e, consequentemente, manipuladas pelo sistema. Todas as entidades são persistidas no arquivo, mas seguindo outra organização, conforme detalhado na Seção 5. A estrutura ilustrada no diagrama representa a maneira como o sistema lida com cada uma das entidades e suas respectivas composições e relacionamentos. Essa estrutura é utilizada para construir as interfaces de usuário e realizar a lógica do sistema que manipula esses componentes entre o arquivo EDN e a interface gráfica. As funcionalidades da aplicação são construídas a partir da definição e agregação desses componentes.

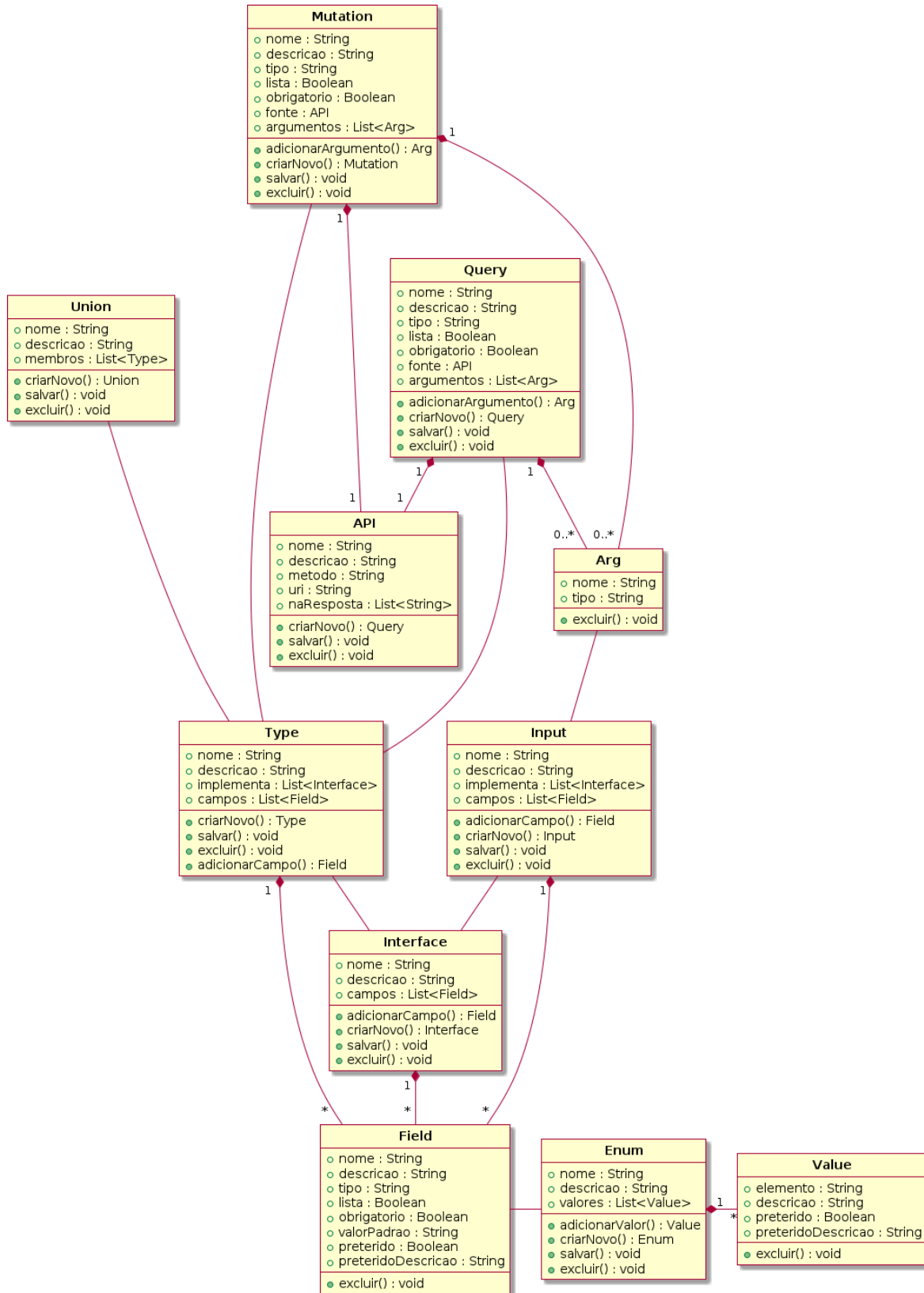


Figura 7. Diagrama de classes do sistema.

3.2 Diagramas de Sequência

Nesta seção, são apresentados os diagramas de sequência referentes aos casos de uso do sistema, mostrando o fluxo entre seus componentes internos para cada cenário. As entidades do diagrama de classes não são apresentadas nos diagramas de sequência, pois elas representam apenas os modelos internos do sistema e não os módulos de execução. Os participantes dos diagramas são os componentes do sistema apresentados na Figura 24.

C01. Definir configurações da API: O usuário pode utilizar a interface gráfica do sistema destinada ao gerenciamento das especificações (*ManagerUI*) para gerenciar as configurações da aplicação. Todas as alterações salvas na interface são refletidas no arquivo EDN. O usuário também pode realizar as alterações diretamente no arquivo EDN. Esse comportamento é representado na Figura 8.

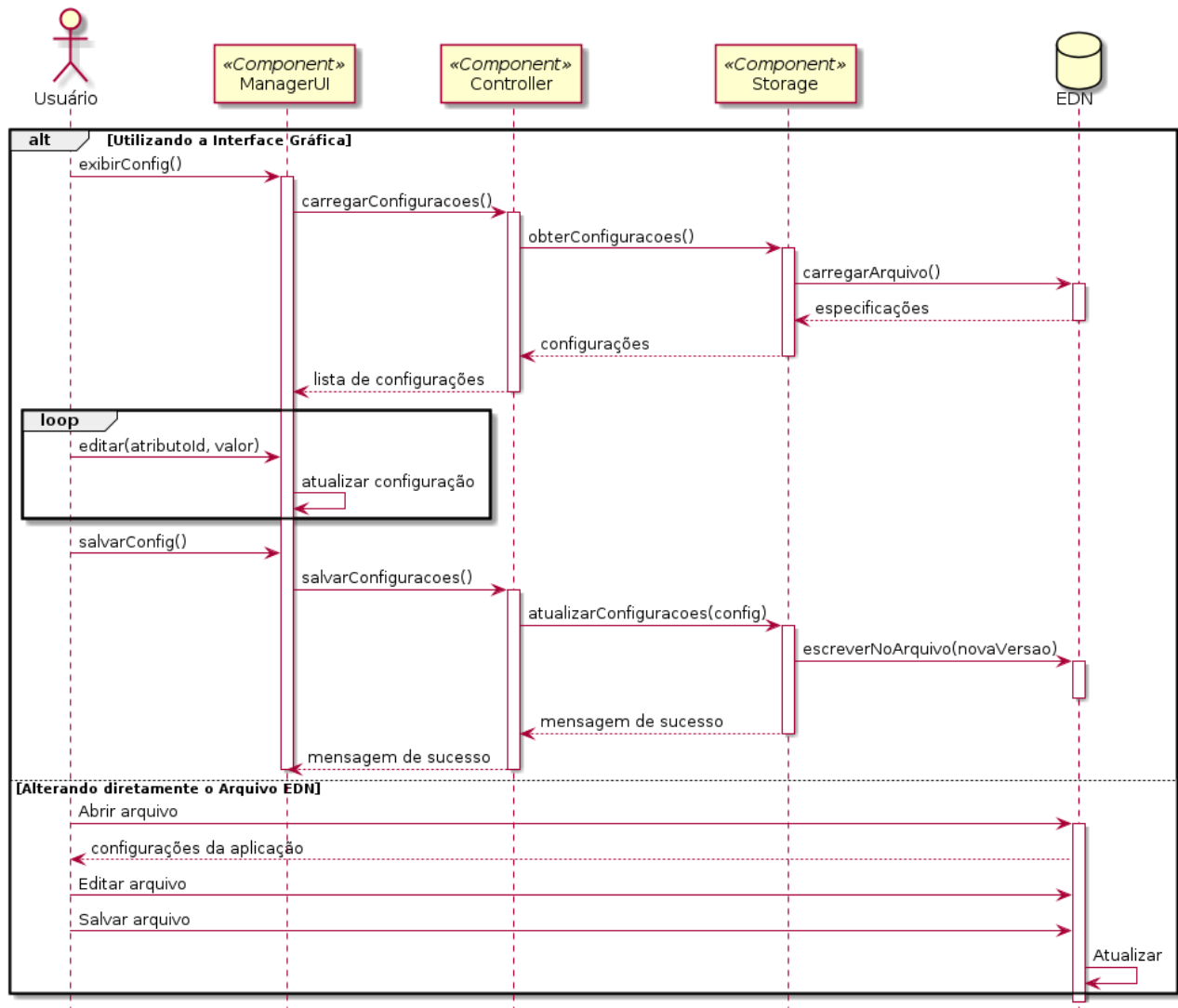


Figura 8. Diagrama de sequência do sistema para o caso de uso C01.

C02. Especificar fontes de dados (Web APIs): O usuário precisa especificar as APIs que são consumidas pela aplicação, fornecendo algumas informações pré-definidas sobre elas. Isso pode ser realizado diretamente pelo arquivo EDN ou utilizando a interface de usuário. Esse comportamento é representado na Figura 9.

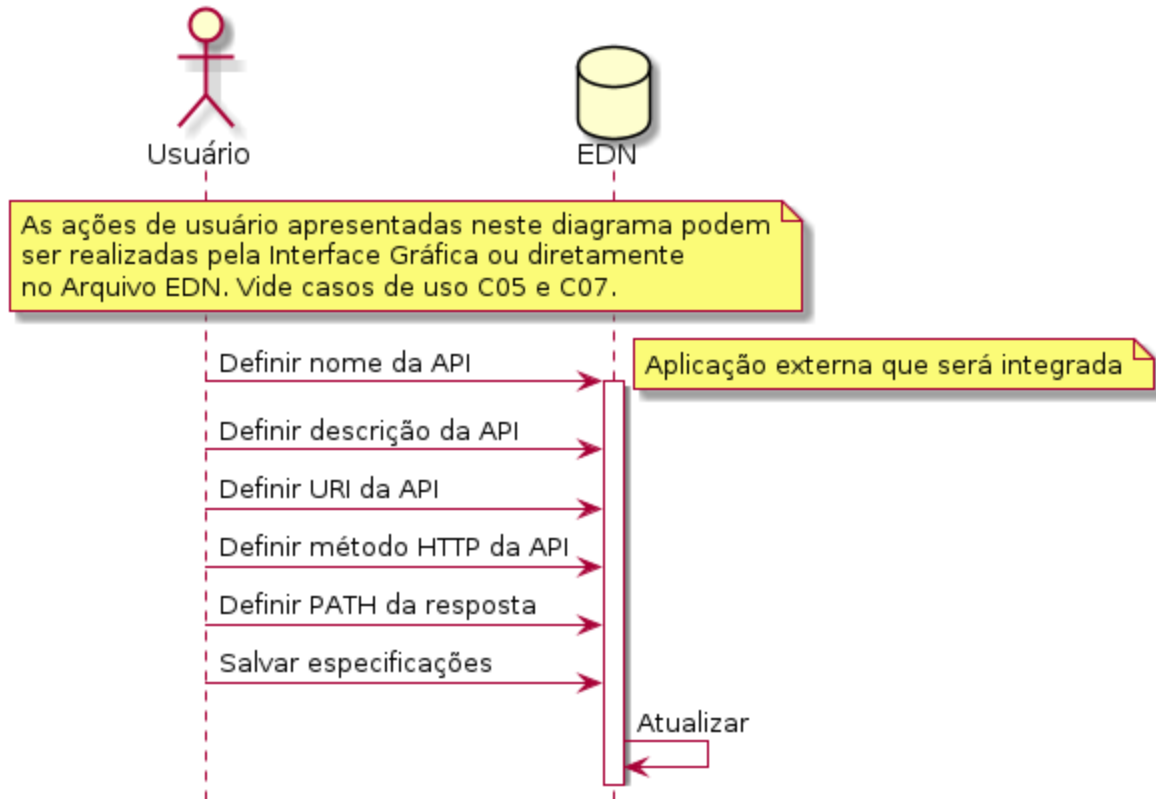


Figura 9. Diagrama de sequência do sistema para o caso de uso C02.

C03. Especificar estruturas e tipos dos dados: O usuário precisa definir as estruturas e formatos de dados que são consumidos e retornados pela aplicação. Há quatro categorias de estruturas, mas o gerenciamento delas segue um padrão com as informações que precisam ser definidas sobre cada uma. Isso pode ser realizado diretamente no arquivo EDN ou pela interface de usuário. Esse comportamento é representado na Figura 10.

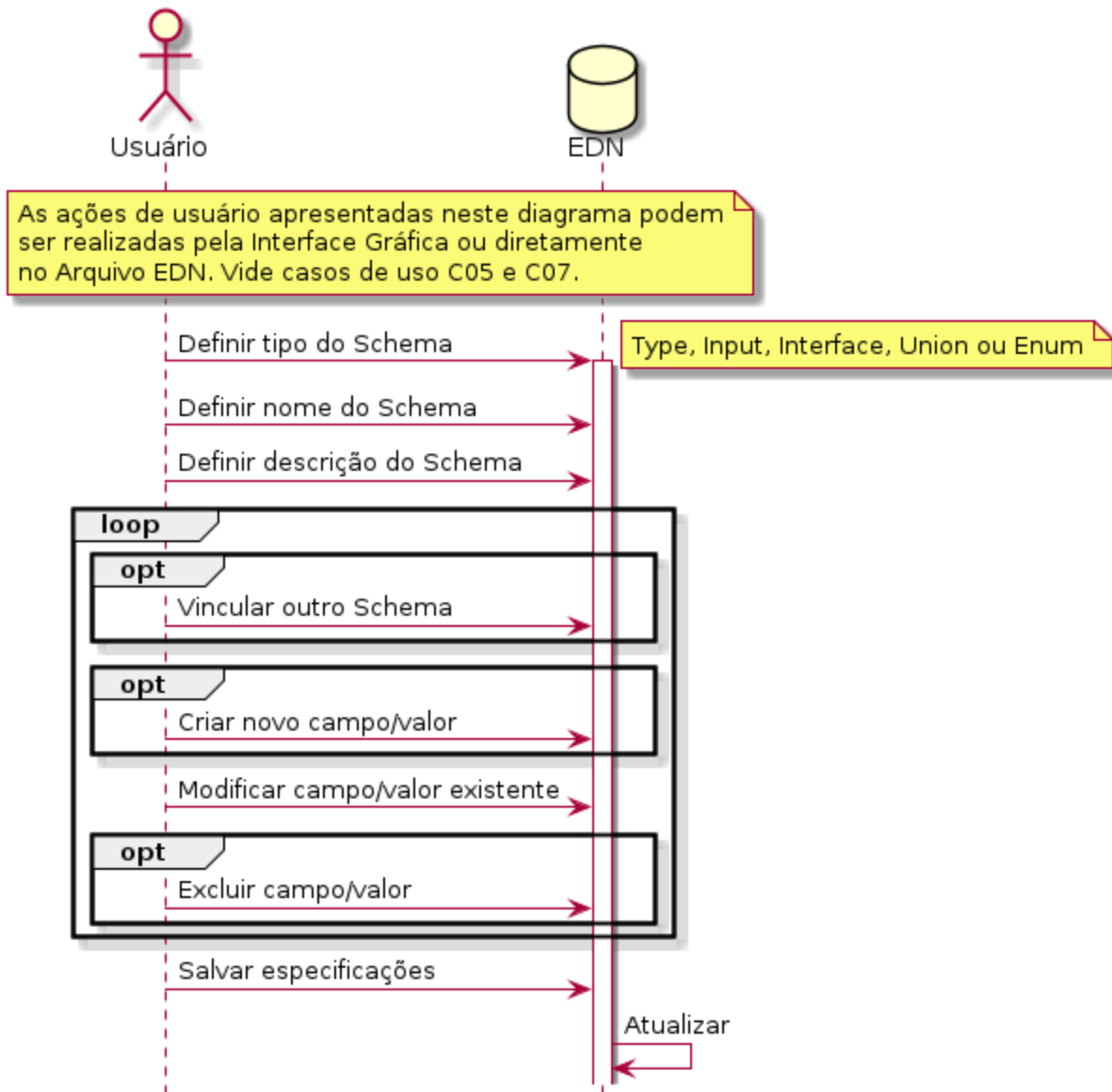


Figura 10. Diagrama de sequência do sistema para o caso de uso C03.

C04. Especificar funcionalidades da API: O usuário precisa definir as operações de consulta (leitura) e mutação (modificação) que a API terá suporte para realizar nas APIs externas. Para isso, é necessário que algumas informações pré-definidas (para ambas as operações) sejam especificadas diretamente no arquivo EDN ou utilizando a interface de usuário. Esse comportamento é representado na Figura 11.

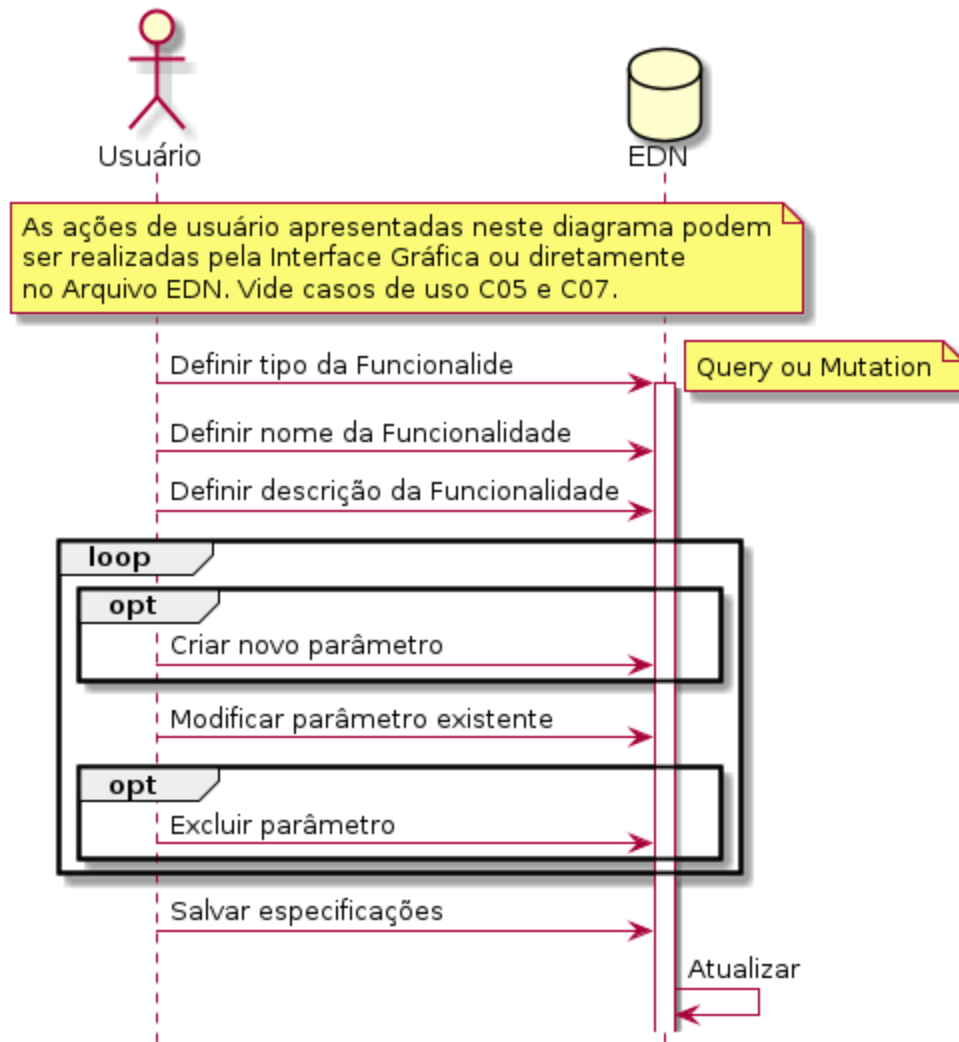


Figura 11. Diagrama de sequência do sistema para o caso de uso C04.

C05. Gerenciar especificações em um arquivo EDN: O usuário pode gerenciar todas as especificações da aplicação diretamente no arquivo EDN. É possível fornecer todas as informações necessárias e modificá-las apenas editando o conteúdo em texto do arquivo EDN. Esse comportamento é representado na Figura 12.

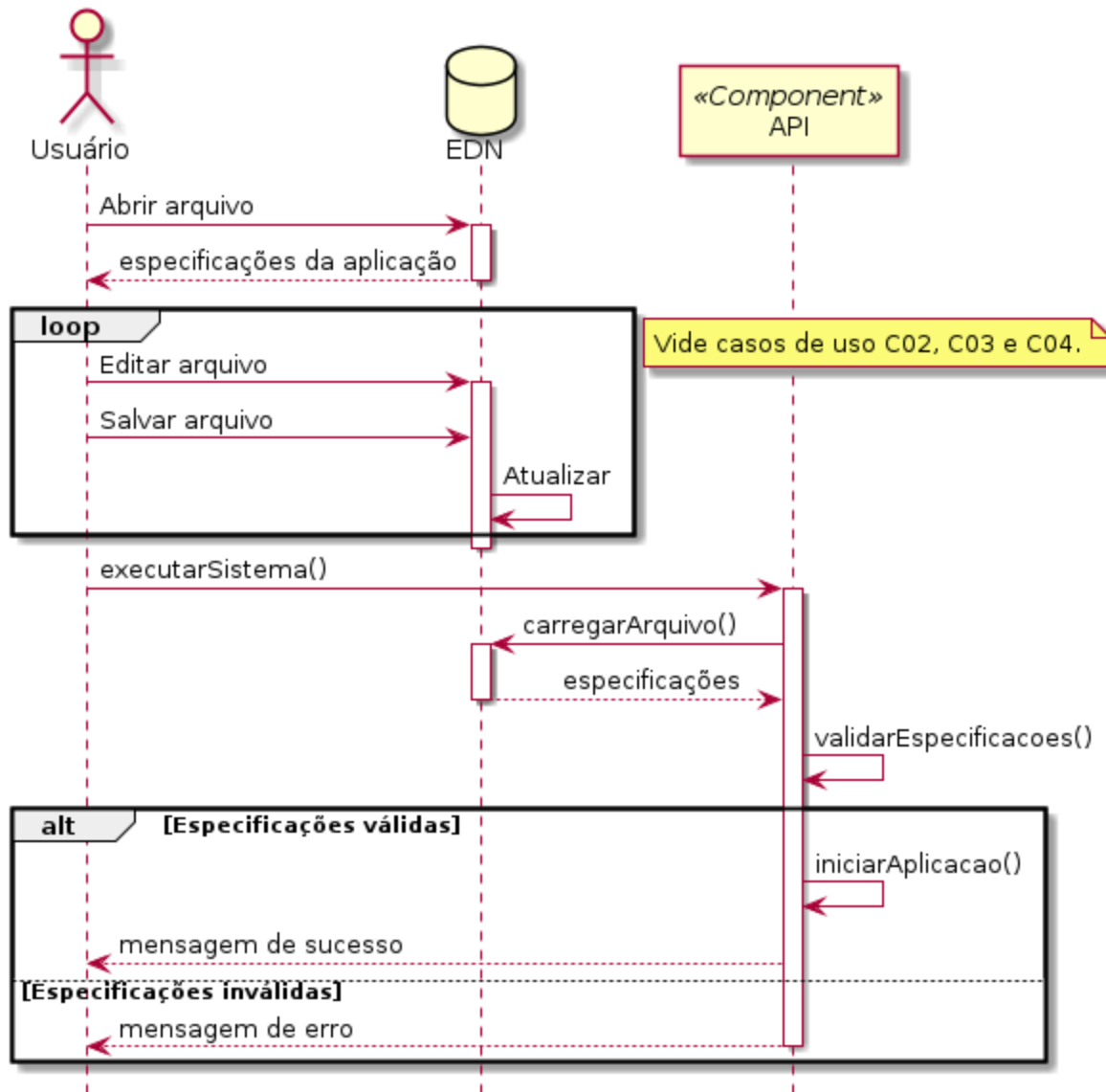


Figura 12. Diagrama de sequência do sistema para o caso de uso C05.

C06. Visualizar especificações por uma interface gráfica: O usuário pode visualizar todas as especificações que estiverem definidas no arquivo EDN por meio da interface de usuário. O sistema carrega os dados do arquivo e os exibe na interface. Esse comportamento é representado na Figura 13.

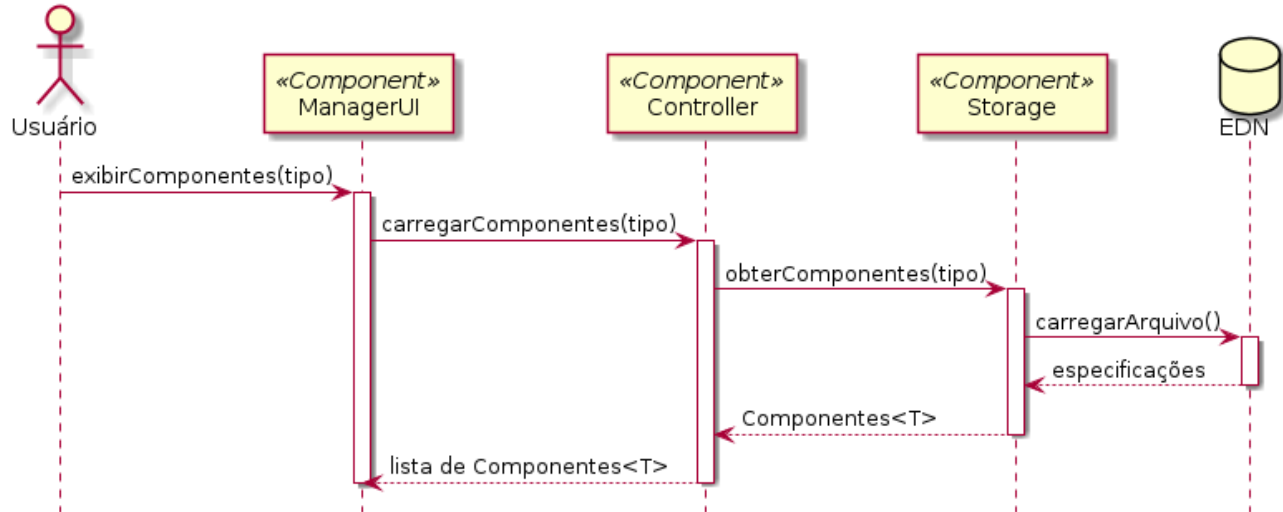


Figura 13. Diagrama de sequência do sistema para o caso de uso C06.

C07. Gerenciar especificações por uma interface gráfica: O usuário pode gerenciar todas as especificações da aplicação utilizando a interface gráfica do sistema (*ManagerUI*). As alterações salvas na interface são refletidas no arquivo EDN. Esse comportamento é representado na Figura 14.

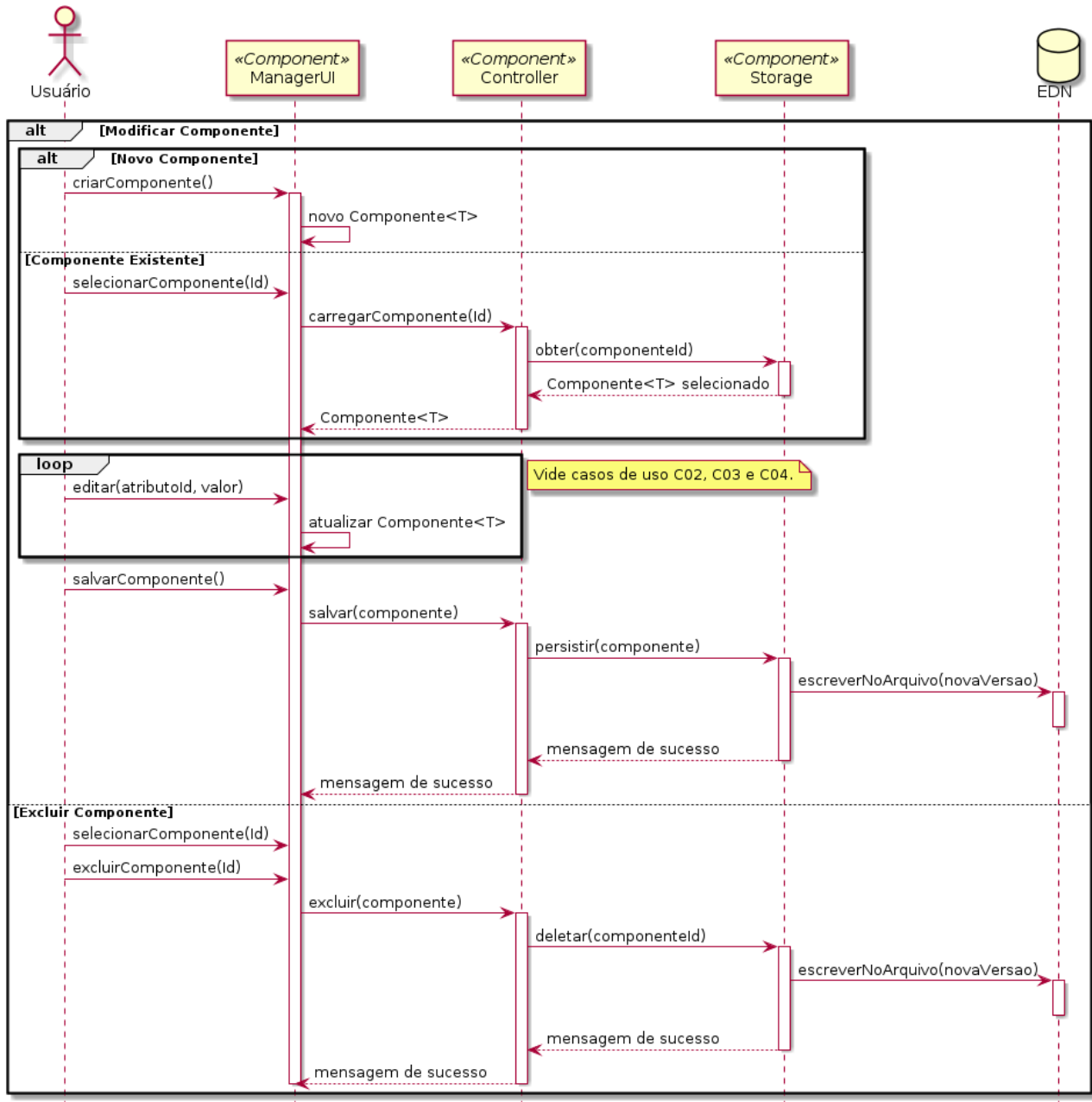


Figura 14. Diagrama de sequência do sistema para o caso de uso C07.

C08. Explorar, de forma visual, os recursos da API GraphQL gerada: O usuário pode visualizar a documentação da API gerada automaticamente, com base nas especificações, e realizar requisições manuais por meio de uma interface gráfica chamada GraphiQL. Essa é uma ferramenta para APIs GraphQL incorporada no sistema, que interage diretamente com a aplicação. Esse comportamento é representado na Figura 15.

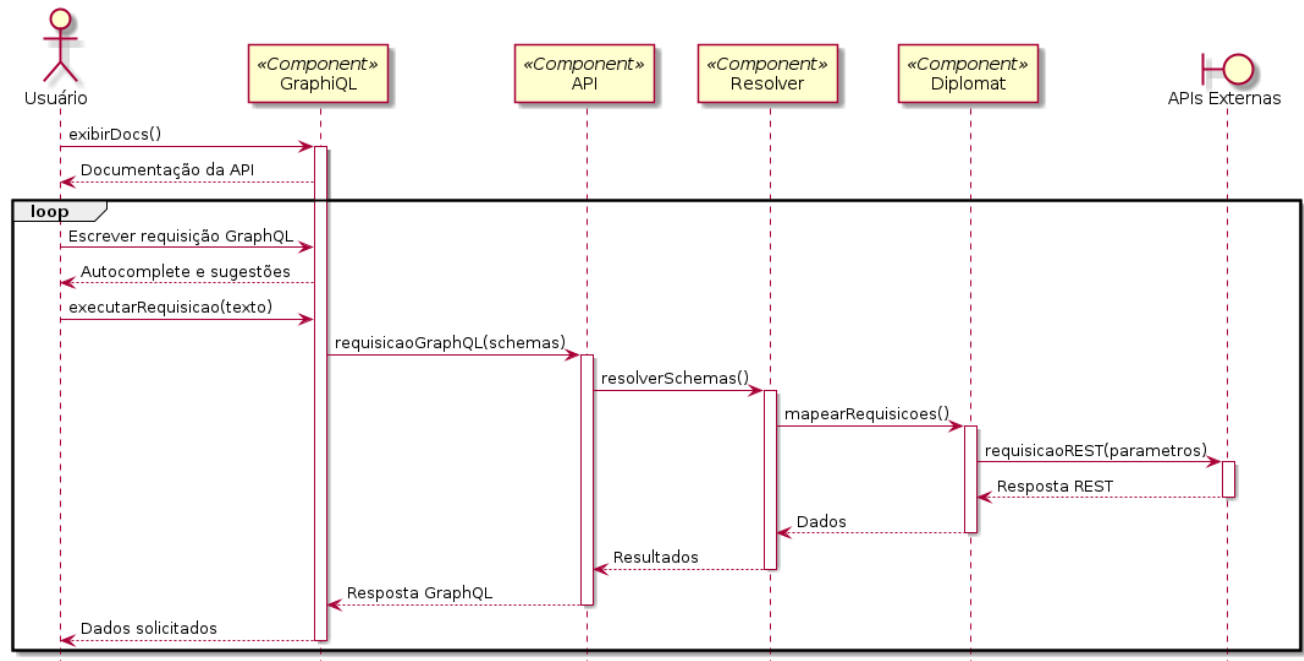


Figura 15. Diagrama de sequência do sistema para o caso de uso C08.

3.3 Diagramas de Comunicação

Nesta seção, estão representados os diagramas de comunicação do sistema, baseados nos diagramas de sequência apresentados na seção anterior. As figuras a seguir ilustram o fluxo de mensagens entre os componentes internos do sistema para todos os casos de uso. A Figura 16 representa a troca de mensagens no fluxo de gerenciamento de configurações do sistema.

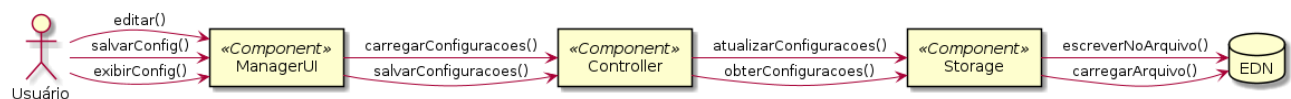


Figura 16. Diagrama de comunicação do sistema para o caso de uso C01.

A Figura 17 representa diretamente o caso de uso C05 e indiretamente os casos de uso C02, C03 e C04. Eles são extensões do caso de uso C05 e não há um fluxo de mensagens específico para cada um deles. O diagrama representa a troca de mensagens para o fluxo de gerenciamento de componentes realizado diretamente pelo arquivo EDN.



Figura 17. Diagrama de comunicação do sistema para os casos de uso C05, C02, C03 e C04.

A Figura 18 representa o caso de uso C06. O diagrama representa a troca de mensagens entre os componentes internos para o fluxo de visualização de componentes por meio da interface de usuário do sistema.

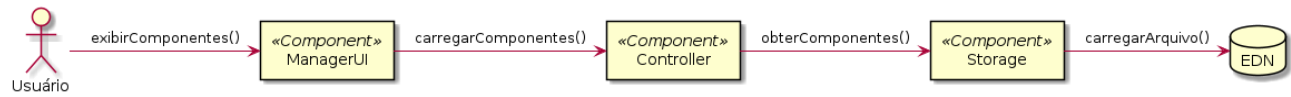


Figura 18. Diagrama de comunicação do sistema para o caso de uso C06.

A Figura 19 representa diretamente o caso de uso C07 e indiretamente os casos de uso C02, C03 e C04. Eles são extensões do caso de uso C07 e não há um fluxo de mensagens específico para cada um deles. O diagrama representa a troca de mensagens para o fluxo de gerenciamento de componentes realizado pela interface de usuário do sistema.

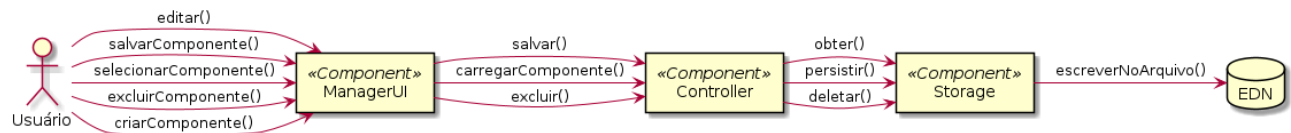


Figura 19. Diagrama de comunicação do sistema para os casos de uso C07, C02, C03 e C04.

A Figura 20 representa o caso de uso C08. O diagrama representa a troca de mensagens entre os componentes internos para o fluxo de utilização da ferramenta GraphiQL incorporada ao sistema.

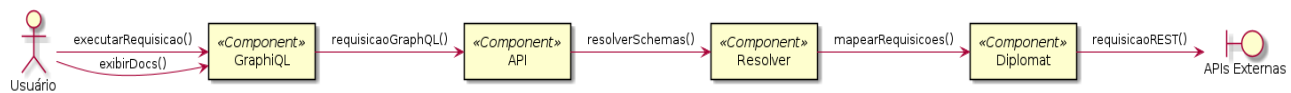


Figura 20. Diagrama de comunicação do sistema para o caso de uso C08.

3.4 Arquitetura

Na Figura 21, está representado o diagrama de arquitetura lógica do sistema, que ilustra a estrutura e organização interna do sistema. Para tal, é definida uma arquitetura baseada no padrão MVC (*Model-View-Controller*). No entanto, as camadas deste sistema são chamadas de *View*, *Logic* e *I/O* (Entrada/Saída, do inglês *Input/Output*). Essa divisão é semelhante ao MVC, porém com uma nomenclatura mais genérica e funcional para se adequar ao contexto do sistema em questão.

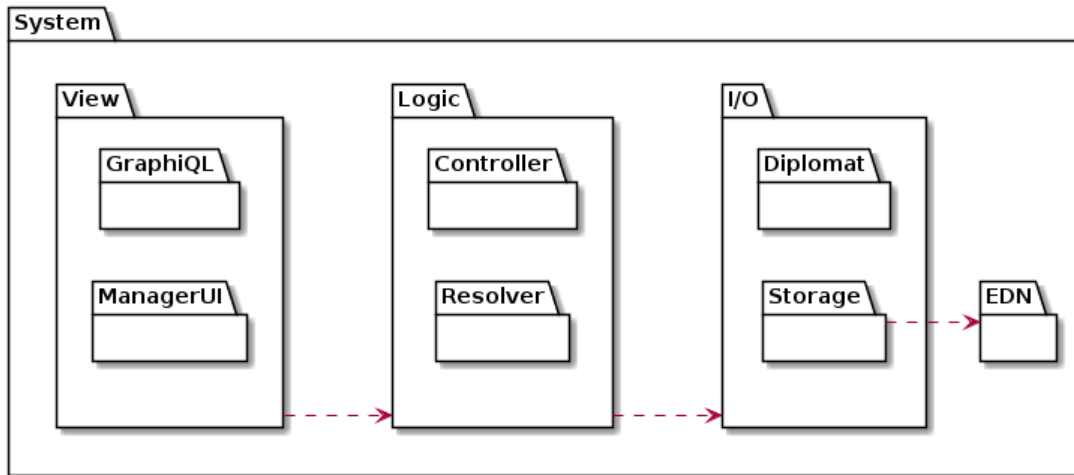


Figura 21. Diagrama de arquitetura lógica do sistema.

3.5 Diagramas de Estados

Na Figura 22, está representado o diagrama de estados do sistema, que intercala entre o gerenciamento, a preparação e o carregamento de uma aplicação BFF. Dessa forma, o sistema possui diferentes estados, que se alternam de acordo com o estágio em que a aplicação está.

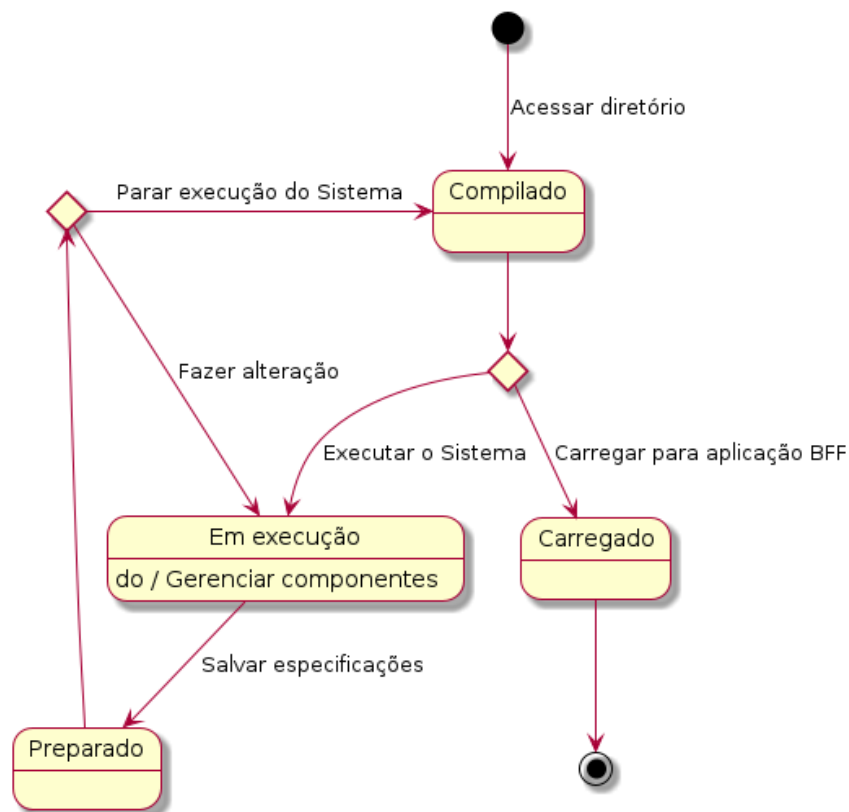


Figura 22. Diagrama de estados do sistema ao gerenciar a aplicação BFF.

Por sua vez, o estado “Em execução” do diagrama apresentado na Figura 22 também intercala entre diferentes estados, enquanto o usuário gerencia os componentes da aplicação utilizando a interface gráfica. Esse fluxo está representado no diagrama de estados da Figura 23.

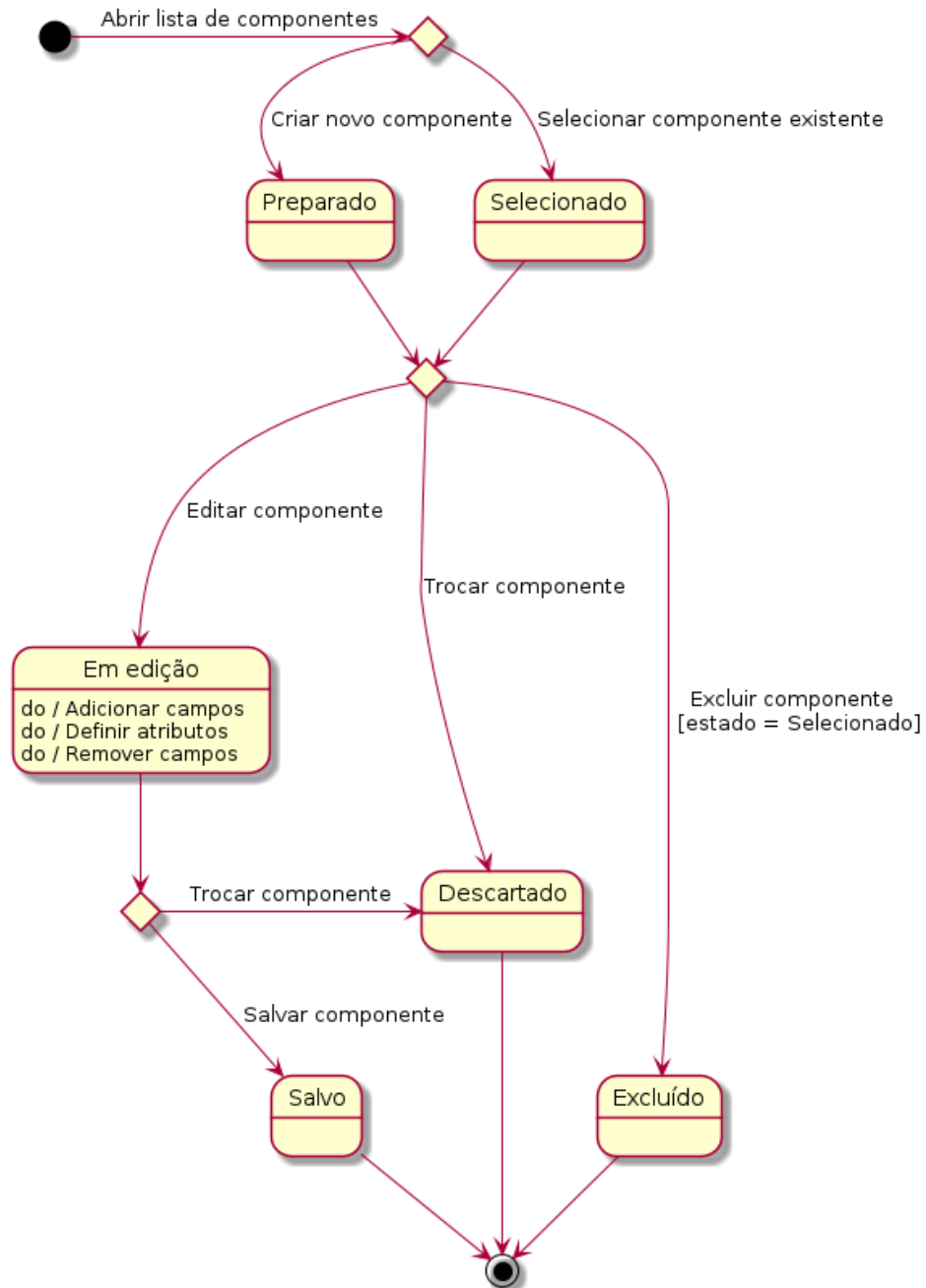


Figura 23. Diagrama de estados do sistema ao gerenciar os componentes da aplicação.

3.6 Diagrama de Componentes e Implantação

Na Figura 24, está modelado o diagrama de componentes do sistema, no qual estão representados os dois módulos do sistema: App (Aplicação) e Manager (Gerenciador). A aplicação é, de fato, a aplicação BFF que é construída pelo sistema, ao passo que o gerenciador inclui o ferramental que possibilita a visualização e manipulação das especificações definidas no arquivo EDN por meio da interface de usuário.

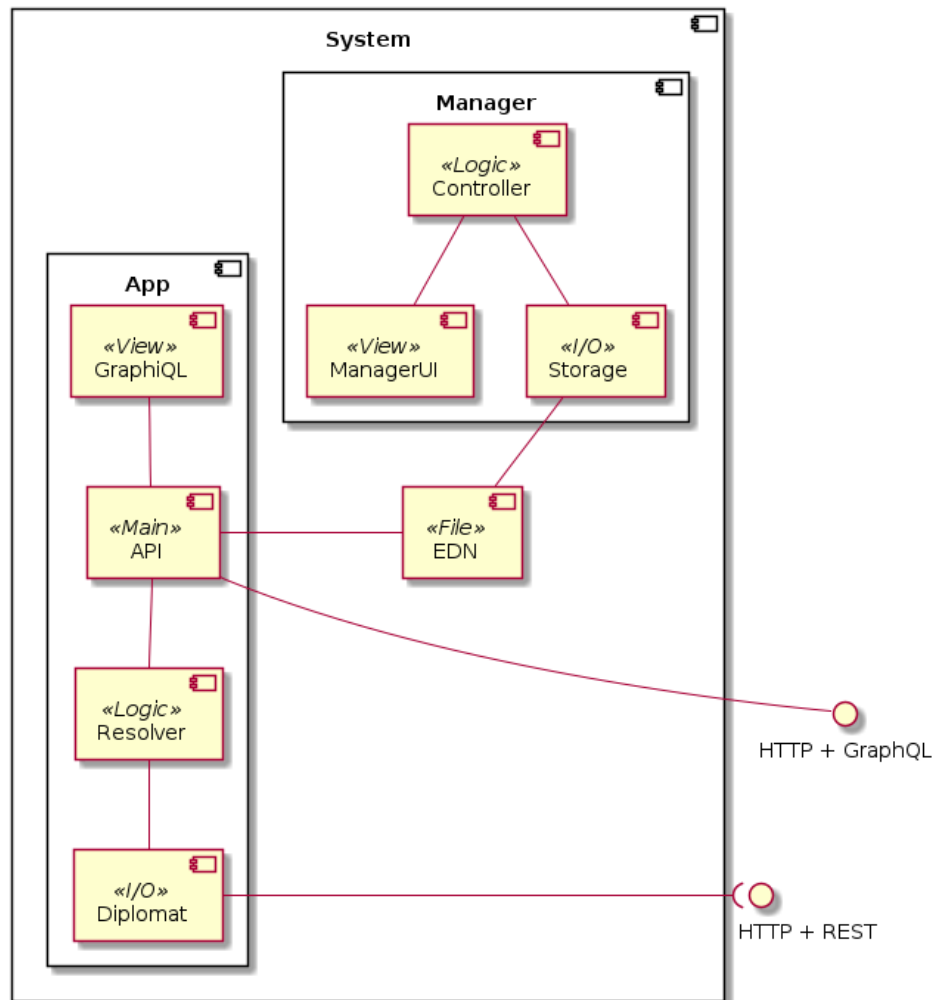


Figura 24. Diagrama de componentes do sistema.

Na Figura 25, está modelado o diagrama de implantação do sistema. Nesse caso, por se tratar de uma aplicação BFF, que é um componente intermediário em uma arquitetura de sistemas distribuídos baseada em microsserviços, o diagrama representa como essa aplicação seria integrada em um ambiente genérico com outras aplicações quaisquer. Todas as integrações são feitas pelas abstrações do sistema combinadas às especificações definidas pelo usuário, como os componentes de APIs, nesse caso.

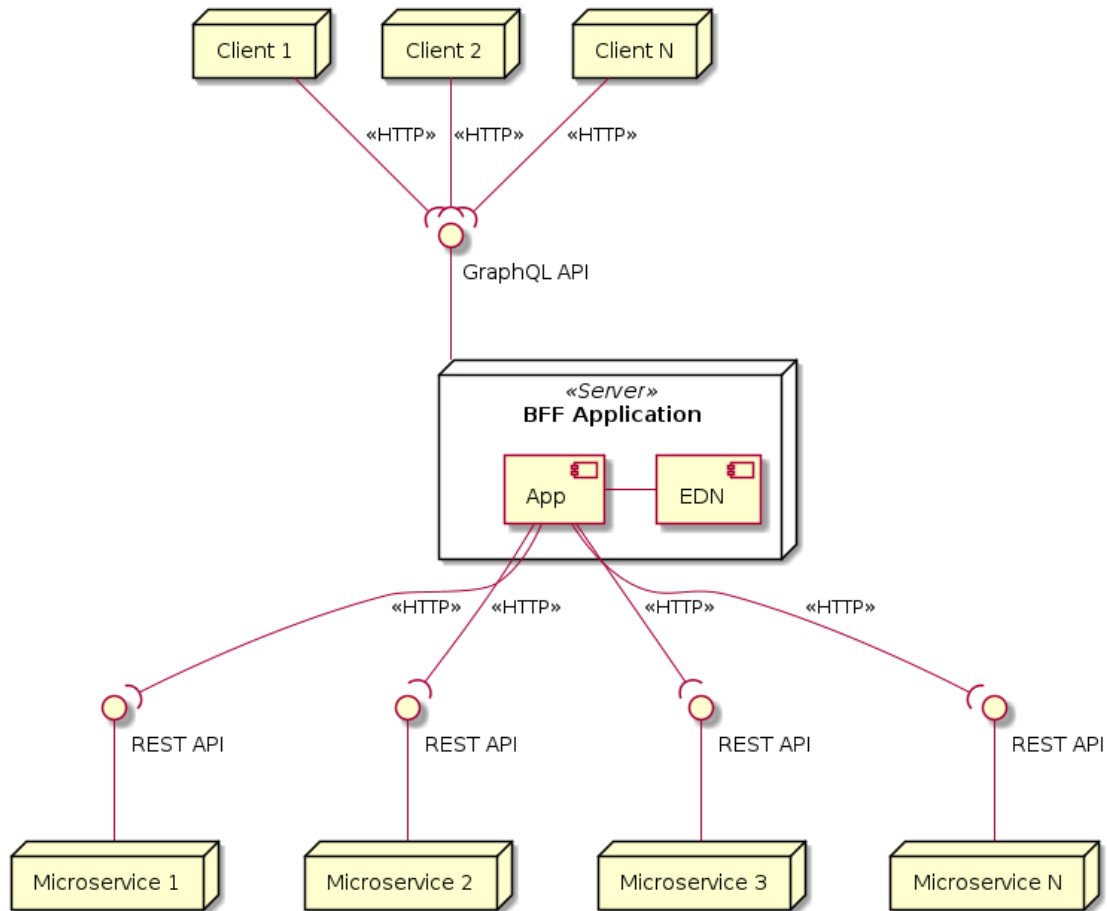


Figura 25. Diagrama de implantação do sistema.

4. Projeto de Interface com Usuário

Nesta seção, são apresentados alguns *wireframes* (protótipos) das interfaces gráficas do sistema, construídas para a interação visual dos usuários com o sistema proposto. Na Figura 26, está representada a tela principal do sistema, que é a tela de gerenciamento de *schemas*, na qual são listados, criados, editados e excluídos todos os modelos de dados tratados pelo sistema. Como todos os outros elementos do sistema a serem definidos pelo usuário, os *schemas* são apenas especificações de dados, com seus atributos e seus respectivos valores. Essa tela contempla os casos de uso C03, C06 e C07.

Header

Schemas APIs Queries Mutations Settings

Enums	Interfaces	Types	Unions	Inputs
		Type 1 >		
		Type 2 >		
		Type 3 >		
		Type 4 >		
		Type 5 >		
		Type 6 >		
		...		

Create new Delete

Name: Input Kind v

Description: Input

Implements / Members: Select v

Fields / Items : Add

Name / Value : Input

Description: Input

Type: Select v ☐ List ☐ Required

Default value: Input

Deprecated? ☐ Input Delete

Save

Figura 26. Wireframe da aba de *Schemas* do sistema.

Na Figura 27, está representada a tela de gerenciamento de APIs, na qual são listadas, criadas, editadas e excluídas todas as fontes de dados externas tratadas pelo sistema. Nesse caso, cada fonte de dados é um *endpoint* de uma outra API, que é consumida para obter e/ou modificar dados, a depender do funcionamento de cada recurso. Nessa tela, são definidas apenas as informações necessárias para realizar uma determinada requisição, já a especificação dos dados recebidos e enviados, assim como os parâmetros suportados, é feita nas abas de *Queries* e *Mutations*. Essa tela contempla os casos de uso C02, C06 e C07.

Header

Schemas APIs Queries Mutations Settings

Endpoint 1 >
Endpoint 2 >
Endpoint 3 >
Endpoint 4 >
Endpoint 5 >
Endpoint 6 >
...

Create new Delete

Name: Input Method v

Description: Input

URI: Input

Save

Figura 27. Wireframe da aba de *APIs* do sistema.

Na Figura 28, está representada a tela de gerenciamento de *Queries*, na qual são listadas, criadas, editadas e excluídas todas as consultas de dados externas tratadas pelo sistema. Nesse caso, cada consulta é um recurso da aplicação fornecido para obter dados de uma determinada API. Nessa tela são definidas apenas as informações necessárias para realizar uma determinada consulta, utilizando modelos de dados e uma API previamente definidos. Além disso, também é feita a especificação dos dados recebidos e enviados, assim como os parâmetros suportados. Essa tela contempla os casos de uso C04, C06 e C07.

Header

Schemas APIs **Queries** Mutations Settings

Query 1 >

Query 2 >

Query 3 >

Query 4 >

Query 5 >

Query 6 >

...

Create new Delete

Name: Input

Description: Input

Type: Select V ☐ List ☐ Required

Source: Select V

Args: Add

Name: Select V

Type: Select V Delete

Name: Select V

Type: Select V Delete

Name: Select V

Type: Select V Delete

Save

Figura 28. Wireframe da aba de *Queries* do sistema.

Na Figura 29, está representada a tela de gerenciamento de *Mutations* (semelhante à tela de gerenciamento de *Queries*), na qual são listadas, criadas, editadas e excluídas todas as modificações de dados externos tratados pelo sistema. Nesse caso, cada modificação é um recurso da aplicação fornecido para enviar dados para uma determinada API. Nessa tela, são definidas apenas as informações necessárias para realizar uma determinada modificação, utilizando modelos de dados e uma API previamente definidos. Além disso, também são definidos os dados enviados e recebidos, assim como os parâmetros suportados. Essa tela contempla os casos de uso C04, C06 e C07.

Header

Schemas APIs Queries **Mutations** Settings

Mutation 1 >

Mutation 2 >

Mutation 3 >

Mutation 4 >

Mutation 5 >

Mutation 6 >

...

Create new Delete

Name: Input

Description: Input

Type: Select ☒ List ☐ Required

Source: Select v

Args: Add

Name: Select v

Type: Select v Delete

Name: Select v

Type: Select v Delete

Name: Select v

Type: Select v Delete

Save

Figura 29. Wireframe da aba de *Mutations* do sistema.

Na Figura 30, está representada a tela de *Settings*, na qual são definidas as configurações de servidor necessárias para a execução da aplicação nos ambientes desejados, expondo uma API GraphQL com os recursos definidos nas outras telas. Essa tela contempla os casos de uso C01, C06 e C07.

Header

Schemas APIs Queries Mutations **Settings**

Name: Input Env: Select v

Description: Input Host: Input

API Path: Input Port: Input

IDE Path: Input CORS: Input

Save

Figura 30. Wireframe da aba de *Settings* do sistema.

Na Figura 31, está representada a tela do GraphiQL (uma ferramenta para testes e exploração de APIs GraphQL), que é incorporada no sistema como parte da aplicação gerada. Essa ferramenta é de código aberto, geralmente disponibilizada para APIs públicas ou como complemento de *frameworks* e bibliotecas de GraphQL para auxiliar no desenvolvimento. Essa tela contempla o caso de uso C08.

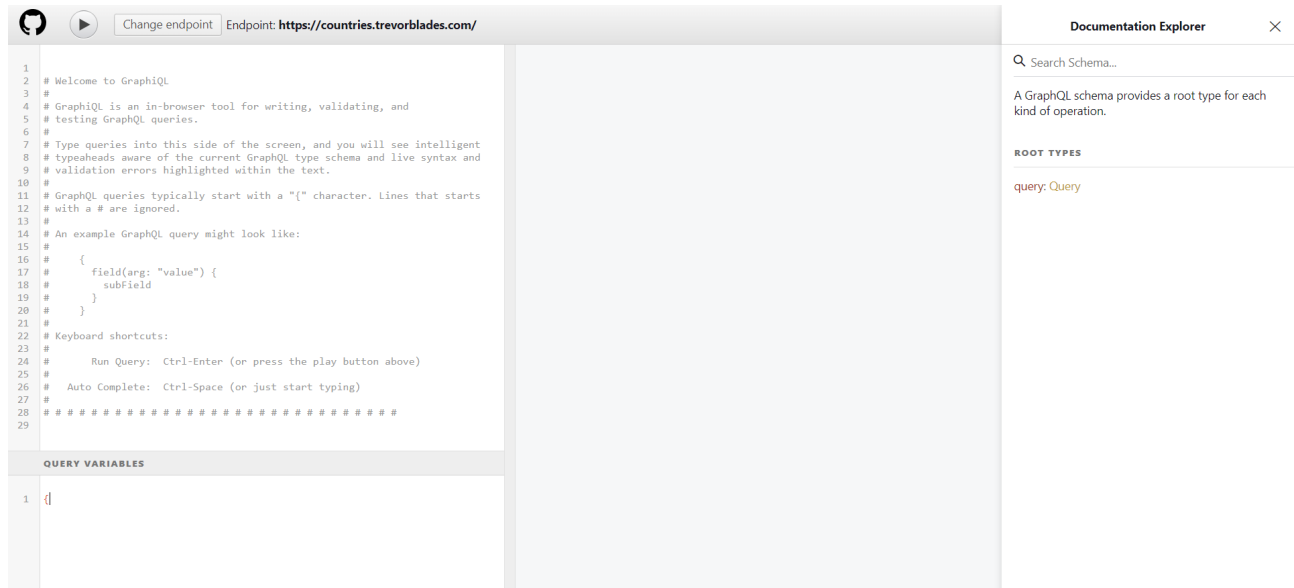


Figura 31. Interface da ferramenta GraphiQL incorporada no sistema.

5. Glossário e Modelos de Dados

Nesta seção, é apresentado o glossário do sistema, contemplado na Tabela 16. Nele, estão os atributos que podem ser visualizados pelo usuário com as suas respectivas descrições. Alguns campos são comuns em diversas telas do sistema, mas a sua nomenclatura, assim como a sua utilização são padronizadas, então eles são, de fato, o mesmo campo para contextos diferentes. No entanto, a definição é dada uma única vez e se aplica a todas as ocorrências dentro do sistema.

Atributo	Formato	Descrição
<i>Name</i>	<i>String</i>	Nome obrigatório e único que cada componente e seus respectivos atributos precisam ter.
<i>Description</i>	<i>String</i>	Descrição opcional para cada componente, assim como para seus atributos e valores.
<i>Kind</i> (<i>Schema</i>)	<i>String</i>	Categoria do <i>Schema</i> que está sendo manipulado, podendo ser um <i>Type</i> , <i>Input</i> , <i>Interface</i> , <i>Union</i> ou <i>Enum</i> .

<i>Implements</i>	Lista de <i>Schemas</i>	<i>Schemas</i> da categoria <i>Interface</i> que estão sendo implementados por outro <i>Schema</i> da categoria <i>Type</i> ou <i>Input</i> .
<i>Members (Union)</i>	Lista de <i>Schemas</i>	<i>Schemas</i> da categoria <i>Type</i> que fazem parte de um mesmo conjunto e podem ser usados de forma condicional.
<i>Type</i>	<i>Schema</i>	Nome de um <i>Schema</i> (primitivo ou composto) que define o tipo de um determinado atributo ou de um componente.
<i>List</i>	<i>Boolean</i>	Define se o tipo de um atributo é um objeto único ou uma lista deles.
<i>Required</i>	<i>Boolean</i>	Define se um determinado atributo é obrigatório para o componente.
<i>Default Value</i>	<i>String</i>	Valor padrão para um determinado atributo de um componente.
<i>Deprecated</i>	<i>String</i>	Indicação e descrição de um determinado atributo de um componente que foi preterido.
<i>Value (Enum)</i>	<i>String</i>	Valor de um determinado elemento que compõem uma enumeração.
<i>Method (API)</i>	<i>String</i>	Método HTTP demandado por um determinado <i>endpoint</i> (recurso) de uma API externa para ser utilizada.
<i>URI</i>	<i>String</i>	Identificador Uniforme de Recurso (do inglês, <i>Uniform Resource Identifier</i>) que define o <i>endpoint</i> de uma API externa para ser utilizado.
<i>Source</i>	<i>Schema</i>	Nome de uma API que será fonte de dados para uma determinada <i>Query</i> ou <i>Mutation</i> .
<i>Name (Settings)</i>	<i>String</i>	Nome da aplicação BFF que está sendo construída e gerenciada.
<i>Description (Settings)</i>	<i>String</i>	Descrição da aplicação BFF que está sendo construída e gerenciada.
<i>API Path</i>	<i>String</i>	Rota (<i>endpoint</i>) da aplicação onde é exposta a API GraphQL.
<i>IDE Path</i>	<i>String</i>	Rota (<i>endpoint</i>) da aplicação onde é exposta a IDE GraphQL.
<i>Host</i>	<i>String</i>	Endereço IP ou URI da aplicação BFF que está sendo construída.
<i>Port</i>	<i>Number</i>	Porta do servidor na qual é exposta a aplicação BFF.
<i>Env</i>	<i>String</i>	Ambiente de execução atual da aplicação (<i>development</i> , <i>staging</i> ou <i>production</i>).
<i>Extra Source</i>	<i>Schema</i>	Nome de uma API utilizada para obter um determinado atributo.

<i>Response Path</i>	<i>String</i>	Sequência de chaves da estrutura de resposta a ser percorrida.
CORS	Lista de <i>Strings</i>	Lista de domínios externos que possuem autorização para se comunicar com a aplicação BFF.
<i>Query Variables</i>	JSON	Variáveis que podem ser definidas no GraphQL para serem utilizadas em qualquer <i>Query</i> e <i>Mutation</i> .

Tabela 16. Glossário do sistema.

Por sua vez, o modelo de dados da aplicação armazena apenas alguns desses campos, usando os mesmos nomes das interfaces de usuário e, conseqüentemente, do glossário. O formato desses dados, no entanto, são representados em notação EDN dentro de um arquivo de especificações dentro do diretório do sistema. Por se tratar de um arquivo de texto, não há relacionamentos e formatos complexos de dados. A organização do armazenamento é baseada em estruturas simples e composições para formar estruturas mais complexas. O gerenciamento dos dados é realizado pelo sistema ou por alterações diretamente no arquivo. A Figura 32 representa a organização e composição dos dados dentro do arquivo.

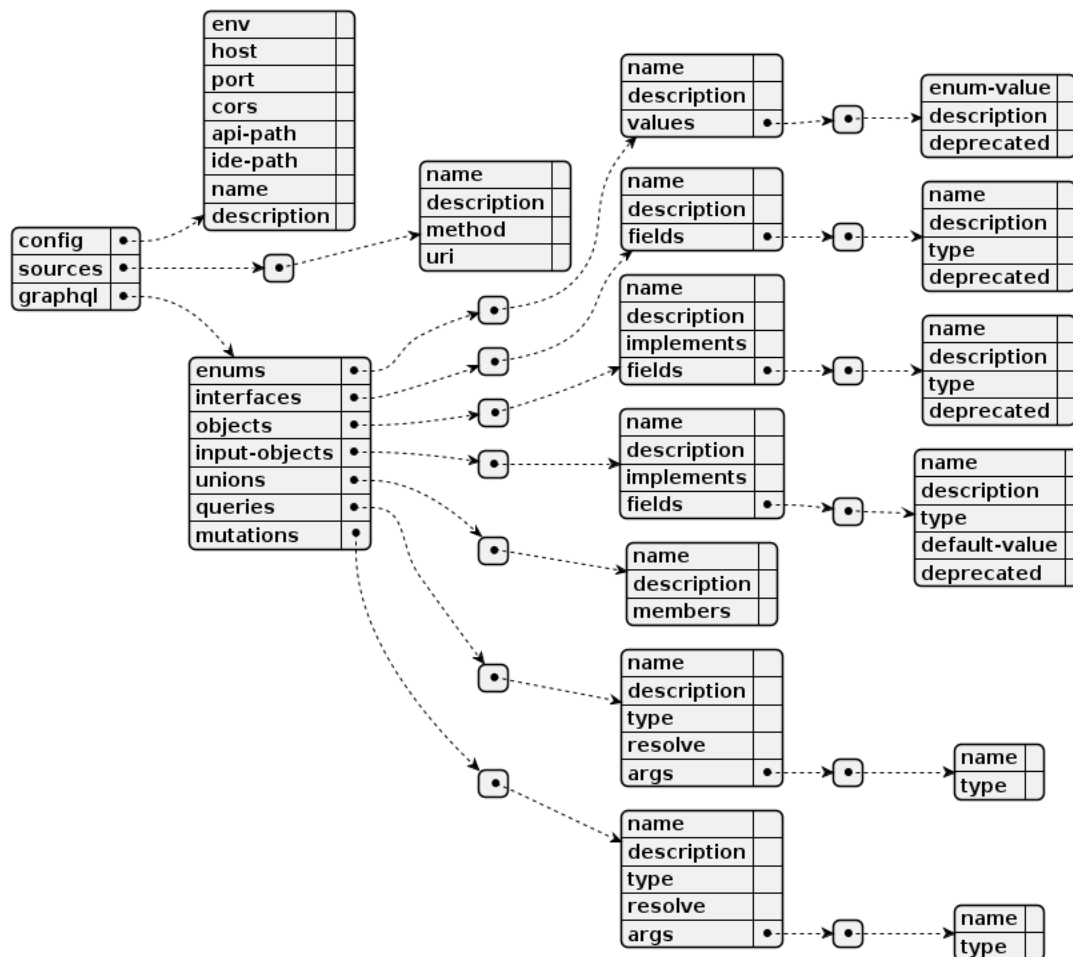


Figura 32. Modelo de dados do sistema.

6. Casos de Teste

Nesta seção, são apresentados os casos de testes definidos para o sistema. Os casos de teste de aceitação, que validam o cumprimento das necessidades levantadas inicialmente, são apresentados na Seção 6.1. Já os casos de teste de integração, que validam o comportamento do sistema ao integrar seus diferentes componentes, são apresentados na Seção 6.2. Para cada caso de teste, é dada uma pré-condição, que é o estado do sistema antes da execução do teste; ações, que são as interações feitas pelo usuário durante a execução e, por fim, o resultado esperado, que é o estado final do sistema após a execução do teste.

6.1 Testes de Aceitação

Os casos de teste de aceitação são definidos a partir das necessidades levantadas no documento de visão. As necessidades deram origem aos requisitos funcionais do sistema. Assim, para cada necessidade, são definidos três casos de teste de aceitação, a fim de validar o cumprimento de cada uma delas. A lista de necessidades do projeto, numerada de 1 a 5, é apresentada a seguir.

1. Integração entre diversos microsserviços e diferentes aplicações clientes;
2. Controle e flexibilidade sobre os dados das requisições;
3. Automação para criação de Web APIs;
4. Facilitação para implementação de GraphQL;
5. Facilitação para implementação de uma aplicação BFF.

As tabelas a seguir detalham os casos de teste para a necessidade 1. É definido um caso de teste para a especificação de uma API externa, contemplado na Tabela 17; um caso de teste para a alteração de uma API já cadastrada, contemplado na Tabela 18 e, por fim, um caso de teste para o mapeamento de uma requisição REST para GraphQL, contemplado na Tabela 19.

Identificador	T1A1
Caso de teste	Definir uma API externa
Pré-condições	<ul style="list-style-type: none">● Sistema em execução;● Interface de usuário aberta na aba de APIs.
Ações	<ul style="list-style-type: none">● Clicar no botão “<i>Create new</i>”;● Preencher informações sobre a API na interface do sistema;● Clicar no botão “<i>Save</i>”.
Resultados esperados	<ul style="list-style-type: none">● Mensagem de sucesso;● Registro armazenado no arquivo EDN.

Tabela 17. Teste de aceitação 1 da necessidade 1.

Identificador	T2A1
Caso de teste	Alterar uma API cadastrada
Pré-condições	<ul style="list-style-type: none"> • Sistema em execução; • Interface de usuário aberta na aba de APIs; • Pelo menos uma API já cadastrada.
Ações	<ul style="list-style-type: none"> • Clicar em uma API cadastrada; • Editar informações da API; • Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> • Mensagem de sucesso; • Registro alterado no arquivo EDN.

Tabela 18. Teste de aceitação 2 da necessidade 1.

Identificador	T3A1
Caso de teste	Mapear uma requisição GraphQL para REST
Pré-condições	<ul style="list-style-type: none"> • Sistema em execução; • GraphiQL aberto no navegador; • API e <i>Query</i> previamente definidas.
Ações	<ul style="list-style-type: none"> • Escrever uma consulta válida na interface; • Executar a requisição.
Resultados esperados	<ul style="list-style-type: none"> • Requisição REST disparada pelo sistema; • Resposta da requisição exibida na tela.

Tabela 19. Teste de aceitação 3 da necessidade 1.

As tabelas a seguir detalham os casos de teste para a necessidade 2. É definido um caso de teste para a especificação de um *Enum*, contemplado na Tabela 20; um caso de teste para a especificação de um *Input*, contemplado na Tabela 21 e, por fim, um caso de teste para a integração de um *Enum* com um *Type*, contemplado na Tabela 22.

Identificador	T1A2
Caso de teste	Definir um <i>Enum</i>
Pré-condições	<ul style="list-style-type: none"> • Sistema em execução; • Interface de usuário aberta na aba de <i>Schemas</i>.
Ações	<ul style="list-style-type: none"> • Clicar no botão “Create new”; • Selecionar “Enum” no campo “Kind”.

	<ul style="list-style-type: none"> ● Preencher informações sobre o <i>Enum</i> na interface do sistema; ● Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> ● Mensagem de sucesso; ● Registro armazenado no arquivo EDN.

Tabela 20. Teste de aceitação 1 da necessidade 2.

Identificador	T2A2
Caso de teste	Definir um <i>Input</i>
Pré-condições	<ul style="list-style-type: none"> ● Sistema em execução; ● Interface de usuário aberta na aba de <i>Schemas</i>.
Ações	<ul style="list-style-type: none"> ● Clicar no botão “Create new”; ● Selecionar “<i>Input</i>” no campo “<i>Kind</i>”; ● Preencher informações sobre o <i>Input</i> na interface do sistema; ● Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> ● Mensagem de sucesso; ● Registro armazenado no arquivo EDN.

Tabela 21. Teste de aceitação 2 da necessidade 2.

Identificador	T3A2
Caso de teste	Relacionar um <i>Enum</i> com um <i>Type</i>
Pré-condições	<ul style="list-style-type: none"> ● Pelo menos um <i>Type</i> previamente definido; ● Pelo menos um <i>Enum</i> previamente definido; ● Sistema em execução; ● Interface de usuário aberta na aba de <i>Types</i>.
Ações	<ul style="list-style-type: none"> ● Clicar no botão “Add” na seção de <i>Fields</i>; ● Preencher as informações do <i>Field</i>; ● Selecionar o <i>Enum</i> previamente definido no campo “<i>Type</i>”; ● Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> ● Mensagem de sucesso; ● Registro atualizado no arquivo EDN; ● Novo campo atribuído ao <i>Type</i> com o tipo e os valores do <i>Enum</i>.

Tabela 22. Teste de aceitação 3 da necessidade 2.

As tabelas a seguir detalham os casos de teste para a necessidade 3. É definido um caso de teste para a realização de requisições com base no arquivo EDN, contemplado na Tabela 23; um caso de teste para o acesso à interface do GraphiQL, contemplado na Tabela 24 e, por fim, um caso de teste para a integração de uma requisição GraphQL com uma requisição REST, contemplado na Tabela 25.

Identificador	T1A3
Caso de teste	Habilitar requisições pelo arquivo EDN
Pré-condições	<ul style="list-style-type: none">• Pelo menos uma API armazenada no arquivo;• Pelo menos um <i>Type</i> armazenado no arquivo;• Pelo menos uma <i>Query</i> ou <i>Mutation</i> válida armazenada no arquivo.
Ações	<ol style="list-style-type: none">1. Executar o sistema;2. Abrir a interface do GraphiQL no navegador;3. Executar uma requisição.
Resultados esperados	<ul style="list-style-type: none">• Aplicação iniciada sem erros de especificações (Após ação 1);• Requisição realizada com sucesso (Após ações 2 e 3).

Tabela 23. Teste de aceitação 1 da necessidade 3.

Identificador	T2A3
Caso de teste	Acessar a página do GraphiQL
Pré-condições	Sistema em execução.
Ações	Acessar a rota definida para o GraphiQL no navegador.
Resultados esperados	Exibição da interface gráfica do GraphiQL.

Tabela 24. Teste de aceitação 2 da necessidade 3.

Identificador	T3A3
Caso de teste	Relacionar uma requisição GraphQL com uma REST
Pré-condições	<ul style="list-style-type: none">• Pelo menos uma API previamente definida;• Pelo menos uma <i>Query</i> previamente definida;• Sistema em execução;• Interface de usuário aberta na aba de <i>Queries</i>.
Ações	<ul style="list-style-type: none">• Clicar em uma <i>Query</i> cadastrada;• Selecionar uma API previamente definida no campo “<i>Source</i>”;

	<ul style="list-style-type: none"> ● Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> ● Mensagem de sucesso; ● Registro atualizado no arquivo EDN; ● API atribuída como fonte de dados da <i>Query</i>.

Tabela 25. Teste de aceitação 3 da necessidade 3.

As tabelas a seguir detalham os casos de teste para a necessidade 4. É definido um caso de teste para a especificação de um *Type GraphQL*, contemplado na Tabela 26; um caso de teste para a especificação de uma *Query GraphQL*, contemplado na Tabela 27 e, por fim, um caso de teste para a especificação de uma *Mutation GraphQL*, contemplado na Tabela 28.

Identificador	T1A4
Caso de teste	Definir um <i>Type GraphQL</i>
Pré-condições	<ul style="list-style-type: none"> ● Sistema em execução; ● Interface de usuário aberta na aba de <i>Schemas</i>.
Ações	<ul style="list-style-type: none"> ● Clicar no botão “<i>Create new</i>”; ● Selecionar “<i>Type</i>” no campo “<i>Kind</i>”; ● Preencher informações sobre o <i>Type</i> na interface do sistema; ● Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> ● Mensagem de sucesso; ● Registro armazenado no arquivo EDN.

Tabela 26. Teste de aceitação 1 da necessidade 4.

Identificador	T2A4
Caso de teste	Definir uma <i>Query GraphQL</i>
Pré-condições	<ul style="list-style-type: none"> ● Sistema em execução; ● Interface de usuário aberta na aba de <i>Queries</i>.
Ações	<ul style="list-style-type: none"> ● Clicar no botão “<i>Create new</i>”; ● Preencher informações sobre a <i>Query</i> na interface do sistema; ● Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none"> ● Mensagem de sucesso; ● Registro armazenado no arquivo EDN.

Tabela 27. Teste de aceitação 2 da necessidade 4.

Identificador	T3A4
Caso de teste	Definir uma <i>Mutation</i> GraphQL
Pré-condições	<ul style="list-style-type: none"> • Sistema em execução; • Interface de usuário aberta na aba de <i>Mutations</i>.
Ações	<ul style="list-style-type: none"> • Clicar no botão “<i>Create new</i>”; • Preencher informações sobre a <i>Mutation</i> na interface do sistema; • Clicar no botão “<i>Save</i>”.
Resultados esperados	<ul style="list-style-type: none"> • Mensagem de sucesso; • Registro armazenado no arquivo EDN.

Tabela 28. Teste de aceitação 3 da necessidade 4.

As tabelas a seguir detalham os casos de teste para a necessidade 5. É definido um caso de teste para o acesso à interface de usuário, contemplado na Tabela 29; um caso de teste para a visualização das especificações na interface, contemplado na Tabela 30 e, por fim, um caso de teste para a persistência das informações no arquivo EDN, contemplado na Tabela 31.

Identificador	T1A5
Caso de teste	Acessar a interface de usuário
Pré-condições	Sistema em execução.
Ações	Acessar a rota definida para a interface web no navegador.
Resultados esperados	Exibição da interface de gerenciamento do sistema.

Tabela 29. Teste de aceitação 1 da necessidade 5.

Identificador	T2A5
Caso de teste	Visualizar especificações na interface gráfica
Pré-condições	<ul style="list-style-type: none"> • Sistema em execução; • Especificações previamente definidas no arquivo.
Ações	<ul style="list-style-type: none"> • Acessar a página de gerenciamento no navegador; • Clicar e navegar entre as abas da interface.
Resultados esperados	<ul style="list-style-type: none"> • Interface de usuário exibida no navegador; • Informações do arquivo exibidas na interface web.

Tabela 30. Teste de aceitação 2 da necessidade 5.

Identificador	T3A5
Caso de teste	Salvar informações da interface no arquivo EDN
Pré-condições	<ul style="list-style-type: none">• Sistema em execução;• Interface de usuário aberta na aba de “Settings”.
Ações	<ul style="list-style-type: none">• Editar o valor de qualquer campo na tela;• Clicar no botão “Save”.
Resultados esperados	<ul style="list-style-type: none">• Mensagem de sucesso;• Alteração salva no arquivo EDN.

Tabela 31. Teste de aceitação 3 da necessidade 5.

6.2 Testes de Integração

Os casos de teste de integração do sistema são definidos a fim de testar o funcionamento dos componentes internos na interação com o ambiente externo. O objetivo é testar apenas as funcionalidades do sistema que envolvem operações de entrada/saída.

Para a interação com o ambiente externo por meio de requisições, são definidos dois casos de teste. Há um caso de teste para o recebimento de requisições GraphQL, contemplado na Tabela 32, e outro para a realização de requisições REST, contemplado na Tabela 33.

Identificador	TI1
Caso de teste	Receber requisições GraphQL
Pré-condições	<i>Queries</i> ou <i>Mutations</i> previamente definidas.
Ações	Enviar uma requisição GraphQL de uma aplicação externa para o <i>endpoint</i> da aplicação BFF.
Resultados esperados	Retornar o resultado de sucesso ou erro da requisição.

Tabela 32. Teste de integração para o recebimento de requisições.

Identificador	TI2
Caso de teste	Realizar requisições REST
Pré-condições	APIs previamente definidas.
Ações	Enviar uma requisição REST da aplicação BFF para uma aplicação externa.

Resultados esperados	Receber resultado de sucesso ou erro da requisição.
-----------------------------	---

Tabela 33. Teste de integração para a realização de requisições.

Para a interação com o ambiente externo por meio de arquivos, são definidos dois casos de teste. Há um caso de teste para a leitura do arquivo EDN, contemplado na Tabela 34, e outro para a escrita no arquivo EDN, contemplado na Tabela 35.

Identificador	TI3
Caso de teste	Ler e interpretar especificações do arquivo EDN
Pré-condições	Um arquivo EDN de especificações dentro do repositório do projeto.
Ações	<ul style="list-style-type: none"> • Ler o conteúdo do arquivo; • Carregar os dados dentro do sistema; • Interpretar as especificações e validá-las.
Resultados esperados	Funcionalidades e recursos do sistema definidos e habilitados pelas especificações.

Tabela 34. Teste de integração para a leitura de arquivos.

Identificador	TI4
Caso de teste	Manipular e escrever especificações no arquivo EDN
Pré-condições	Um arquivo EDN de especificações dentro do repositório do projeto.
Ações	<ul style="list-style-type: none"> • Modificar as especificações dentro do sistema; • Formatar e serializar os dados; • Escrever o conteúdo no arquivo.
Resultados esperados	Arquivo salvo contendo especificações válidas para o sistema.

Tabela 35. Teste de integração para a escrita em arquivos.

7. Cronograma e Processo de Implementação

Nesta seção, é apresentado o processo adotado para a execução do projeto. A implementação do sistema é organizada em *Milestones*, entre Agosto a Novembro de 2022, seguindo uma adaptação da metodologia ágil *Scrum*. Dessa maneira, o sistema é desenvolvido de forma iterativa e incremental. Para cada *Milestone*, é planejada a execução de um conjunto de tarefas, sendo elas destinadas à implementação de alguma parte do sistema, realização de testes manuais e/ou automatizados, ou à criação da documentação do sistema. A seguir, é apresentada a lista das tarefas a serem executadas, numeradas de 1 a 16, indicando a ordem de execução delas.

1. Fazer a configuração inicial do projeto e do repositório;
2. Implementar componente *Storage*;
3. Implementar componente API (*Main*) com GraphQL;
4. Implementar componente *Resolver*;
5. Implementar componente *Diplomat*;
6. Implementar componente *Controller*;
7. Implementar componente *ManagerUI*;
8. Implementar tela de gerenciamento de *Schemas*;
9. Implementar tela de gerenciamento de APIs;
10. Implementar tela de gerenciamento de *Queries*;
11. Implementar tela de gerenciamento de *Mutations*;
12. Implementar tela de gerenciamento de *Settings*;
13. Gerar executáveis do sistema (App e Manager);
14. Empacotar os artefatos do sistema (App, Manager e EDN);
15. Implementar *script* de execução do módulo Manager;
16. Escrever manual de uso do sistema.

A lista de tarefas apresentada anteriormente é definida com base nos requisitos do sistema, projetos de interface e diagrama de componentes, dividindo a estrutura do sistema em partes que possam ser implementadas individualmente, para serem compostas à medida que as tarefas seguintes sejam concluídas. O cronograma referente ao segundo semestre letivo de 2022 é apresentado na Tabela 36, com a indicação de quais tarefas são executadas em cada *Milestone*.

2022					
Número da Tarefa	Milestone 1	Milestone 2	Milestone 3	Milestone 4	Milestone 5
	01/08 - 28/08	29/08 - 18/09	19/09 - 09/10	10/10 - 30/10	31/10 - 13/11
1	X				
2	X				
3	X				
4		X			
5		X			
6		X			
7		X			
8			X		

9			X		
10			X		
11			X		
12			X		
13				X	
14				X	
15				X	
16					X

Tabela 36. Cronograma de desenvolvimento do projeto.

Para gerenciar o desenvolvimento do sistema, é utilizado um quadro *kanban* no repositório do projeto pela funcionalidade *Projects* do GitHub. No quadro, há uma *issue* para cada uma das tarefas listadas anteriormente, alocadas em quatro colunas, sendo movidas entre elas à medida que são iniciadas e concluídas. O código produzido na execução das tarefas é enviado para o mesmo repositório, abrindo um *pull request* (PR) que referencia o *Milestone* correspondente antes de mesclar o código na *branch* principal. Dessa maneira, à medida que um conjunto de tarefas é finalizado, o *pull request* é mesclado e as *issues* são movidas para a última coluna do *kanban*. Em cada *branch* criada, há o código-fonte desenvolvido para as tarefas, com seus respectivos testes automatizados. Todos os testes são executados de forma automática dentro do PR, utilizando a funcionalidade de *Actions* do GitHub, para verificar se a alteração está em conformidade, antes de mesclar o novo código ao antigo.

Para a validação de versões intermediárias do sistema, são definidos dois momentos em que a combinação das tarefas concluídas gera uma versão possível de ser utilizada parcialmente. O primeiro momento é ao final do *Milestone 2*, em que já é possível utilizar as funcionalidades principais da aplicação BFF definindo as especificações diretamente no arquivo EDN. O segundo momento é ao final do *Milestone 3*, em que é possível utilizar o sistema pela interface de usuário para definir as especificações. Já ao final do *Milestone 4*, tendo o sistema completo, é possível utilizá-lo a partir dos executáveis e *scripts* criados.

Ao final do desenvolvimento do projeto, é escrito um manual de uso do sistema em *markdown*, contido no *README* do repositório, para guiar o processo de instalação e utilização. Essa documentação é composta por instruções, sugestões e exemplos em forma de textos, imagens e trechos de código. No repositório, também há o código-fonte da ferramenta, assim como uma pequena aplicação de exemplo criada durante o desenvolvimento do projeto para testes e demonstrações. Esse exemplo deve ajudar novos usuários a compreenderem a estrutura da ferramenta e de uma aplicação BFF construída a partir dela, podendo executá-las em seus próprios ambientes e explorarem o seu funcionamento.

8. Post Mortem

Nesta seção, são descritas algumas das experiências obtidas durante o desenvolvimento deste projeto. A Seção 8.1 contém um relato das experiências positivas. A Seção 8.2 contém um relato das experiências negativas. A Seção 8.3 apresenta algumas lições aprendidas referentes aos processos de ideação e execução deste projeto. Por fim, a Seção 8.4 contém o *link* para o repositório do projeto com o resultado.

8.1 Experiências Positivas

Dentre as experiências positivas, é possível destacar a assertividade do planejamento realizado antes da implementação e que possibilitou o cumprimento das necessidades definidas ao especificar as estruturas e os fluxos que o software deveria possuir. Outro ponto positivo é a linguagem de programação escolhida. Clojure é uma linguagem muito poderosa para a manipulação de dados e a sua flexibilidade para ser executada no ambiente Java (JVM) e JavaScript (Node.js) foi fundamental para uma construção eficiente dos módulos do sistema, permitindo, inclusive, o compartilhamento de código entre eles. Já em relação à gestão do processo de desenvolvimento, o fluxo de trabalho funcionou de forma organizada, utilizando o *kanban* para acompanhar as tarefas no próprio repositório e utilizando a automação do GitHub *Actions* para executar, simultaneamente, todos os testes automatizados para as plataformas Windows, Linux e macOS em todos os *commits*, possibilitando a identificação de erros e inconsistências no código e/ou problemas de compatibilidade entre as plataformas.

8.2 Experiências Negativas

Embora não tenham sido marcantes, ocorreram algumas experiências negativas durante o desenvolvimento do sistema. A principal delas ocorreu no início da implementação, devido à dificuldade em configurar um projeto Clojure multiplataforma (Java e JavaScript). Não há muitos tutoriais atualizados na Internet sobre como fazer a configuração em um mesmo projeto, ainda que tenham vários que abordem a configuração individual. Um tempo considerável foi tomado para esse estudo e diversas tentativas foram realizadas até, de fato, obter um sistema funcional a partir de uma mesma base de código para ambas as plataformas. Além disso, a comunidade Clojure é mais madura para o ecossistema Java do que para JavaScript, então algumas ferramentas e bibliotecas não funcionam para as duas plataformas e, em alguns casos, não há nem mesmo uma equivalente à altura para o ecossistema ClojureScript, principalmente em relação aos testes automatizados. Contudo, essas questões não foram impeditivas, mas dificultaram o processo de desenvolvimento e exigiram mais esforços do que o esperado para algumas tarefas.

8.3 Lições Aprendidas

Dentre as lições aprendidas, destaca-se a necessidade de planejar e estimar o tempo de estudo, pesquisa e experimentação durante o processo de desenvolvimento para cenários em que as tecnologias, ferramentas e/ou métodos não sejam de domínio de quem estiver desenvolvendo. Esse tempo é importante para garantir o sucesso, a qualidade e a consistência do projeto. Afinal, não é possível ignorá-lo, pois impacta diretamente no resultado e nos prazos de entrega. Durante o desenvolvimento do projeto, não ocorreram problemas irreversíveis com os prazos estipulados inicialmente, mas foi demandado um esforço extra para a execução de diversas tarefas que, embora estivessem bem especificadas pela documentação, necessitavam de um tempo de aprendizado antes de realizar a implementação. Por outro lado, essas situações poderiam ter sido mitigadas se a documentação do projeto definisse alguns aspectos de forma mais prática, listando referências (*links* úteis) para a realização de algumas tarefas ou, até mesmo, os artefatos que pudessem ser utilizados para cada uma delas, como bibliotecas de código, *frameworks*, provas de conceito ou exemplos.

8.4 Repositório do Projeto

O código-fonte desenvolvido durante a implementação do sistema proposto, assim como a sua respectiva documentação e manual de uso encontram-se disponíveis em um repositório do GitHub. O *link* para o repositório é:

<https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2022-1-tcci-5308100-dev-marlon-silva>