Documentação de Projeto

para o sistema

GraphTest

Versão 1.4

Projeto de sistema elaborado pelo(s) aluno(s) Lucas Cabral Soares e Maria Eduarda Amaral Muniz e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo dos professores Danilo de Quadros Maia Filho, Leonardo Vilela Cardoso, Raphael Ramos Dias Costa, orientação acadêmica do professor Cleiton Silva Tavares e orientação de TCC II do professor (a ser definido no próximo semestre).

25/09/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Lucas Cabral Soares e Maria Eduarda Amaral Muniz	28/09/2025	• Criação do documento; • Adicionando a Seção 1. Introdução; • Adicionando a Seção 2.3. Modelo de Casos de Uso e Histórias de Usuário; • Adicionando a Seção 2.3.1. Diagrama de Casos de Uso; • Adicionando a Seção 2.3.2. Histórias de Usuários • Adicionando a Seção 4. Projeto de Interface com Usuário • Adicionando a Seção 4.1 Esboço das Interfaces Comuns a Todos os Atores	1.0
Lucas Cabral Soares e Maria Eduarda Amaral	12/10/2025	• Criar diagrama de domínio • Criação do diagrama de sequência do sistema • Criação do diagrama de sequência • Criação do diagrama de comunicação (Fluxo GFC)	1.1
Lucas Cabral Soares	18/10/2025	• Atualizar Tabela de Conteúdo adicionando hiperlinks • Ajustar páginas do documento	1.2
Maria Eduarda Amaral	01/11/2025	• Adicionar diagramas de arquitetura e estados	1.3
Lucas Cabral Soares	02/11/2025	• Fazer diagrama de componente • Fazer diagrama de Implantação	1.4
Maria Eduarda Amaral	02/11/2025	• Adicionar DER e Glossário	1.4

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o Sistema de Geração e Análise de Grafos de Teste. A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema. Tal especificação acompanha este documento. Anexo a este documento também se encontra o Glossário.

O presente documento descreve o projeto do Sistema de Geração e Análise de Grafos de Teste, desenvolvido como parte do Trabalho de Conclusão de Curso (TCC) do curso de Engenharia de Software da PUC Minas. O sistema tem como objetivo fornecer uma plataforma integrada para a criação, visualização e análise de artefatos de teste de software, abrangendo tanto a perspectiva estrutural (Grafo de Fluxo de Controle – GFC) quanto a funcional (Grafo de Causa e Efeito – GCE).

O sistema foi concebido para atender às necessidades de estudantes, professores e profissionais da área de Tecnologia da Informação (TI), oferecendo um ambiente interativo que permite colar ou importar trechos de código, gerar automaticamente o GFC, aplicar métricas de cobertura estrutural e derivar, a partir de modelos de causa e efeito, tabelas de decisão e casos de teste funcionais. A plataforma também possibilita a validação lógica de modelos, o cálculo de métricas e a exportação de artefatos em diferentes formatos, promovendo o uso educacional e profissional em contextos de ensino, pesquisa e prática de engenharia de software.

Este documento reúne os principais modelos de requisitos, projeto, dados e interface que compõem o sistema. Inclui diagramas de casos de uso, de classes, de sequência e de componentes, bem como os contratos de operações correspondentes, que detalham o comportamento esperado de cada funcionalidade. O conteúdo aqui descrito serve como base técnica para a implementação do sistema, garantindo rastreabilidade entre os requisitos elicitados e os artefatos de projeto gerados.

2. Modelos de Usuário e Requisitos

Esta seção tem como finalidade descrever o perfil do usuário e os requisitos funcionais e não funcionais que orientam o desenvolvimento do sistema. Nela são apresentados os atores que interagem com a plataforma, os modelos de usuário desenvolvidos com base em personas e os casos de uso que estruturam o comportamento esperado do sistema.

A seção inicia com a descrição do ator principal, responsável por representar todos os perfis que utilizam o sistema. Em seguida, são apresentados os modelos de usuário, que detalham características, motivações e objetivos por meio de personas representativas. Posteriormente, são descritos o modelo de casos de uso e as histórias de usuário, que expressam as funcionalidades identificadas a partir das necessidades elicitadas durante a análise de requisitos.

2.1 Descrição de Atores

Esta seção apresenta os atores que interagem diretamente com o sistema, descrevendo seus papéis, objetivos e principais ações dentro da plataforma. O propósito é identificar quem utiliza o sistema e de que forma essas interações ocorrem, servindo de base para a definição dos casos de uso.

No contexto do Sistema de Geração e Análise de Grafos de Teste, todos os perfis de usuários, como estudantes, professores e profissionais da área, foram unificados sob o ator Usuário, que representa qualquer indivíduo que utiliza a ferramenta para gerar, visualizar e analisar grafos de teste, bem como realizar atividades de cobertura e modelagem de casos de teste.

Usuário: representa o indivíduo que interage diretamente com o sistema, abrangendo diferentes perfis, como estudantes, professores e profissionais da área de Engenharia de Software. O usuário utiliza a plataforma para gerar, visualizar e analisar grafos de teste, tanto sob a perspectiva estrutural (Grafo de Fluxo de Controle – GFC) quanto funcional (Grafo de Causa e Efeito – GCE).

Entre suas principais ações estão:

- colar ou importar trechos de código para gerar o GFC;
- visualizar métricas de cobertura de comandos, decisões, condições e caminhos;
- modelar relações de causa e efeito para derivar tabelas de decisão e casos de teste funcionais;
- validar modelos e analisar inconsistências;
- exportar imagens e relatórios em diferentes formatos para uso acadêmico ou integração com outras ferramentas.

Esse ator concentra todas as interações externas relevantes do sistema, representando o comportamento de qualquer pessoa que utilize o sistema para fins de ensino, análise ou apoio ao processo de teste de software.

2.2 Modelos de Usuários

Esta subseção tem como objetivo descrever os modelos de usuários desenvolvidos por meio da implementação de personas. Para a construção das personas, foram considerados os diferentes perfis que utilizarão a plataforma: estudantes de Engenharia de Software, professores universitários de teste de software e engenheiros de software profissionais.

A Tabela 1 descreve a persona do usuário Julia Alves Silva, uma estudante de Engenharia de Software. Nela é possível perceber que, apesar de interessada no aprendizado, ela encontra dificuldades em compreender os conceitos teóricos de grafos quando apresentados apenas em aula

expositiva. Por isso, deseja contar com uma ferramenta interativa que permita visualizar e manipular os grafos de forma prática, associando-os diretamente à derivação de casos de teste.

Julia Alves Silva	
Descrição	Julia tem 21 anos, cursa o quinto período de Engenharia de Software em Belo Horizonte e está atualmente matriculada na disciplina de Teste de Software. Apesar de interessada pela área, sente dificuldade em compreender conceitos abstratos de cobertura estrutural apenas a partir de slides. Ela vê na plataforma uma forma de praticar a geração de grafos de fluxo de controle e de causa-efeito, relacionando-os com casos de teste concretos.
Dores	• Dificuldade em visualizar conceitos abstratos apenas em teoria. • Falta de ferramentas práticas que auxiliem no aprendizado em sala.
Objetivos	• Praticar a geração de grafos de fluxo de controle a partir de pequenos trechos de código. • Entender a derivação de casos de teste funcionais por meio de grafos de causa-efeito.

Tabela 1. Persona Julia Alves Silva

A Tabela 2 descreve a persona do usuário Carlos Eduardo Menezes, um engenheiro de software profissional. Sua descrição mostra que ele tem experiência prática com testes, mas frequentemente precisa analisar grandes volumes de código e validar a qualidade dos testes existentes. Ele sente como dor a fragmentação entre ferramentas de cobertura estrutural e funcional, o que dificulta a visão integrada da qualidade de um sistema.

Carlos Eduardo	
Descrição	Carlos tem 34 anos, atua como engenheiro de software em uma empresa de desenvolvimento de sistemas corporativos. Trabalha diariamente com testes unitários e de integração, utilizando ferramentas de cobertura como JaCoCo. No entanto, enfrenta dificuldade para conectar métricas de cobertura estrutural com derivação de testes funcionais, precisando recorrer a diferentes ferramentas e relatórios manuais.

Dores	• Necessidade de alternar entre ferramentas para avaliar cobertura estrutural e funcional. • Dificuldade em identificar rapidamente lacunas nos testes implementados.
Objetivos	• Importar código do projeto e gerar automaticamente o grafo de fluxo de controle. • Avaliar cobertura de testes já existentes e identificar pontos não exercitados. • Derivar casos de teste funcionais a partir de requisitos.

Tabela 2. Persona Carlos Eduardo

A Tabela 3 descreve a persona do usuário Mariana Rocha Almeida, professora universitária de Teste de Software. Sua rotina exige o preparo de aulas, exercícios e exemplos práticos. Ela sente como dor a ausência de ferramentas didáticas que integrem conceitos teóricos e visuais, capazes de apoiar o ensino e a avaliação dos estudantes em disciplinas de teste

Mariana Rocha Almeida	
Descrição	Mariana tem 42 anos, é doutora em Ciência da Computação e leciona disciplinas de Qualidade e Teste de Software em uma universidade. Costuma preparar exercícios com base em exemplos de código simples e em especificações de sistemas. Entretanto, a construção manual de grafos e tabelas de decisão é trabalhosa e nem sempre engaja os alunos.
Dores	• Esforço elevado para criar exemplos manuais de grafos e tabelas de decisão. • Falta de ferramentas que ilustram de forma clara e didática o processo de geração de casos de teste.
Objetivos	• Utilizar a plataforma em sala de aula para demonstrar a geração de grafos e casos de teste. • Explorar visualmente a relação entre requisitos, fluxos de controle e cobertura estrutural/funcional.

Tabela 3. Persona Mariana Rocha Almeida

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta subseção tem como objetivo descrever os casos de uso e histórias de usuário previstos para o projeto. Para isso é apresentado um diagrama de casos de uso onde todos são listados. Dessa mesma forma, também são apresentadas as histórias de usuário relacionadas às funcionalidades previstas para o sistema a ser desenvolvido.

2.3.1 Diagrama de Casos de Uso

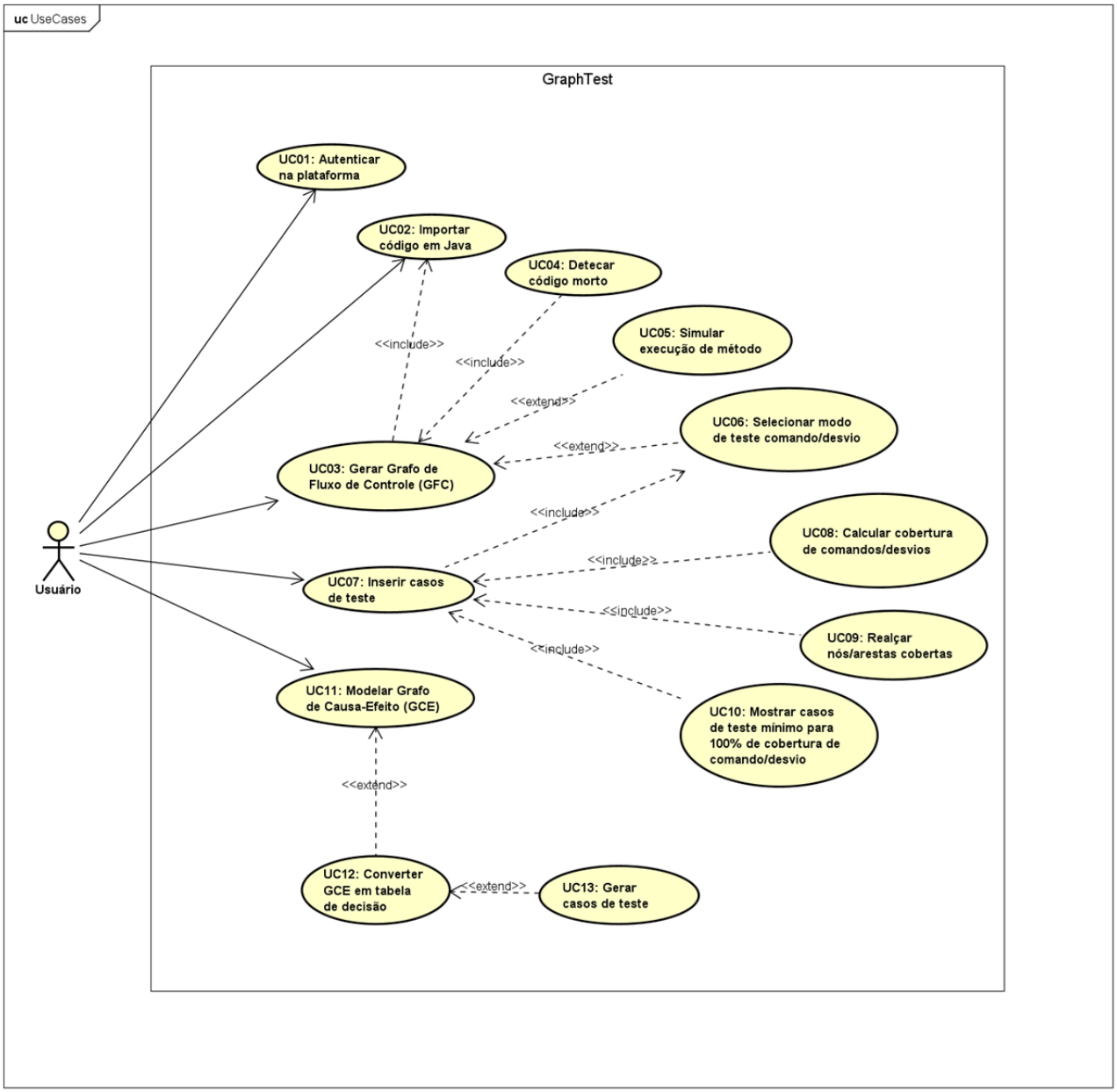

Figura 1. Diagrama de Caso de Uso

2.3.2 Histórias de Usuário

As histórias de usuários apresentadas nesta seção representam os artefatos que orientam o desenvolvimento do sistema. As histórias de usuários se relacionam com os casos de uso, onde para cada caso de uso tem-se uma história de usuário. Desse modo, relacionam-se da seguinte forma: para cada caso de uso (UC, do inglês *Use Case*) tem-se uma história de usuário (US, do inglês *User Story*), sendo UC01 relacionado ao US01, UC02 relacionado ao US02 e assim por diante. Portanto, as histórias de usuário identificadas para o sistema são:

US01. Como usuário, desejo me autenticar utilizando meu e-mail e senha para usar as funcionalidades do sistema.

US02. Como usuário, desejo importar um código-fonte em Java para analisar seu fluxo e gerar o grafo correspondente.

US03. Como usuário, desejo gerar o Grafo de Fluxo de Controle (GFC) a partir de um método do código importado para visualizar o seu comportamento estrutural.

US04. Como usuário, desejo detectar automaticamente trechos de código morto para identificar nós inalcançáveis e otimizar o código analisado.

US05. Como usuário, desejo simular a execução de um método para acompanhar visualmente o fluxo de execução passo a passo no grafo.

US06. Como usuário, desejo selecionar o modo de teste (comando ou desvio) para avaliar o tipo de cobertura desejada durante o teste estrutural.

US07. Como usuário, desejo inserir casos de teste para verificar a cobertura de comandos e desvios sobre o grafo de fluxo de controle.

US08. Como usuário, desejo calcular automaticamente a cobertura de comandos e desvios para avaliar quantitativamente a eficácia dos testes inseridos.

US09. Como usuário, desejo visualizar nós e arestas cobertas em destaque para identificar graficamente as partes do código exercitadas pelos testes.

US10. Como usuário, desejo visualizar os casos de teste mínimos que garantem 100% de cobertura para otimizar o conjunto de testes executados.

US11. Como usuário, desejo modelar um Grafo de Causa-Efeito (GCE) definindo causas, efeitos, operadores e restrições para representar regras funcionais do sistema sob teste.

US12. Como usuário, desejo converter o GCE em uma tabela de decisão para visualizar as combinações lógicas entre causas e efeitos de forma estruturada.

US13. Como usuário, desejo gerar automaticamente casos de teste funcionais a partir da tabela de decisão para garantir que todas as combinações lógicas sejam validadas durante o teste.

2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Esta seção apresenta os Diagramas de Sequência do Sistema (DSS), que tem como finalidade representar a interação entre o ator principal e o sistema durante a execução de cada caso de uso. Diferentemente dos diagramas de sequência detalhados com múltiplos objetos, aqui o foco está na visão de alto nível do comportamento do sistema, mostrando como o usuário (ator) comunica-se diretamente com o software por meio de suas principais operações.

Cada Diagrama de Sequência do Sistema (DSS) descreve o fluxo de mensagens entre o ator "Usuário" e o "Sistema DSS", representando as solicitações realizadas pelo usuário e as respostas processadas pelo sistema. As mensagens são dispostas em ordem cronológica, evidenciando o caminho lógico seguido para a execução de cada funcionalidade. Essa representação permite compreender como o sistema se comporta externamente frente às ações do usuário, sem detalhar a decomposição interna dos módulos.

Associado a cada diagrama, apresenta-se um Contrato de Operações, que define formalmente a operação realizada pelo sistema — incluindo pré-condições, pós-condições, entradas, saídas e exceções. Esses contratos descrevem, de maneira estruturada, as responsabilidades do sistema e os efeitos esperados após a execução de cada operação, garantindo clareza no comportamento funcional e nos limites de atuação do software.

O diagrama da Figura 2 representa a interação entre o ator Usuário e o Sistema DSS durante o processo de autenticação na plataforma, correspondente ao caso de uso UC01 – Autenticar na plataforma. O fluxo tem início quando o usuário insere suas credenciais de acesso (usuário e senha) e solicita a autenticação por meio do comando realizaLogin(). O sistema, então, valida as credenciais fornecidas, verificando sua correspondência com os dados armazenados. A partir dessa verificação, o diagrama apresenta um bloco de decisão (alt) com dois caminhos alternativos. No primeiro, quando as credenciais são válidas, o sistema executa as operações iniciaSessao() e carregaDashboard(), concedendo acesso ao ambiente principal da aplicação. No segundo caminho, quando as credenciais são inválidas, o sistema executa mostrarMensagemDeErro(), informando ao usuário que o processo de login não foi concluído com sucesso.

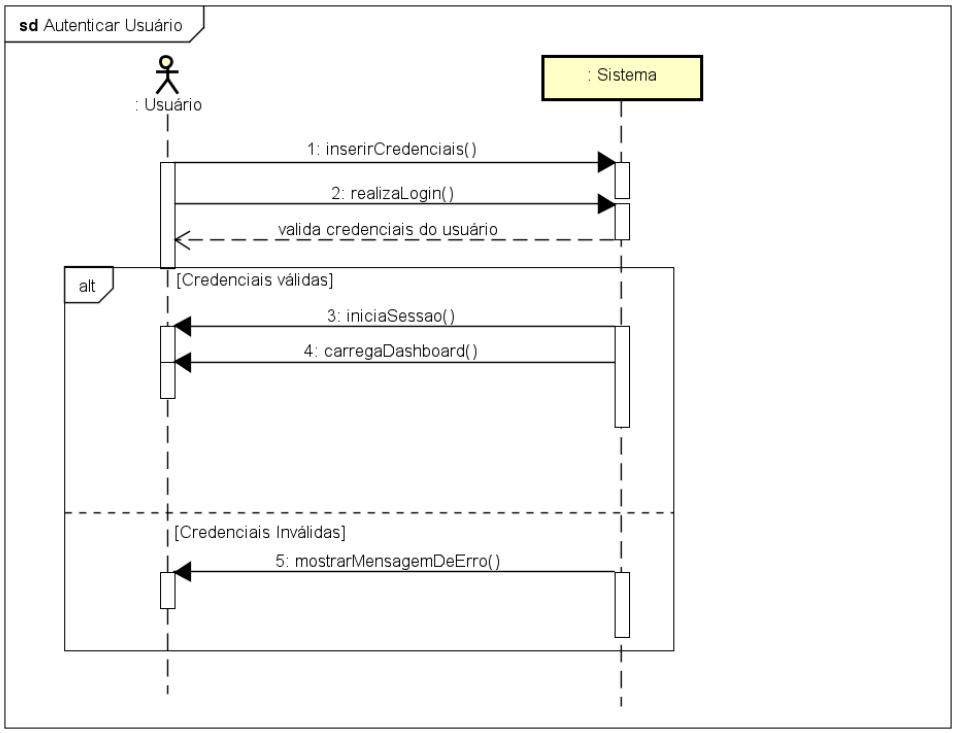

Figura 2. DSS: Autenticar Usuário

Contrato	Autenticar Usuário
Operação	inserirCredenciais()
Referências cruzadas	UC1: Autenticar na plataforma
Pré-condições	O usuário precisa estar cadastrado no sistema
Pós-condições	O usuário precisa estar autenticado

O diagrama da figura 3 representa a interação entre o ator *Usuário* e o *Sistema DSS* no processo de geração do Grafo de Fluxo de Controle (GFC) a partir de um trecho de código colado no editor. O fluxo inicia quando o usuário abre o editor, insere o código e aciona o comando para gerar o GFC. O sistema então realiza etapas internas de normalização do texto, detecção da linguagem e validação de conteúdo. Caso o editor esteja vazio, é exibida uma mensagem de erro e o processo é interrompido. Se o código for válido, o sistema executa o parser para gerar a Árvore Sintática (AST), tratando eventuais erros de análise. Em seguida, o GFC é construído com base na AST, mapeando cada nó e aresta às respectivas linhas do código-fonte. Por fim, o grafo é salvo como artefato e renderizado na interface, sendo apresentado ao usuário com a visualização interativa e o vínculo direto entre o código e a estrutura do grafo.

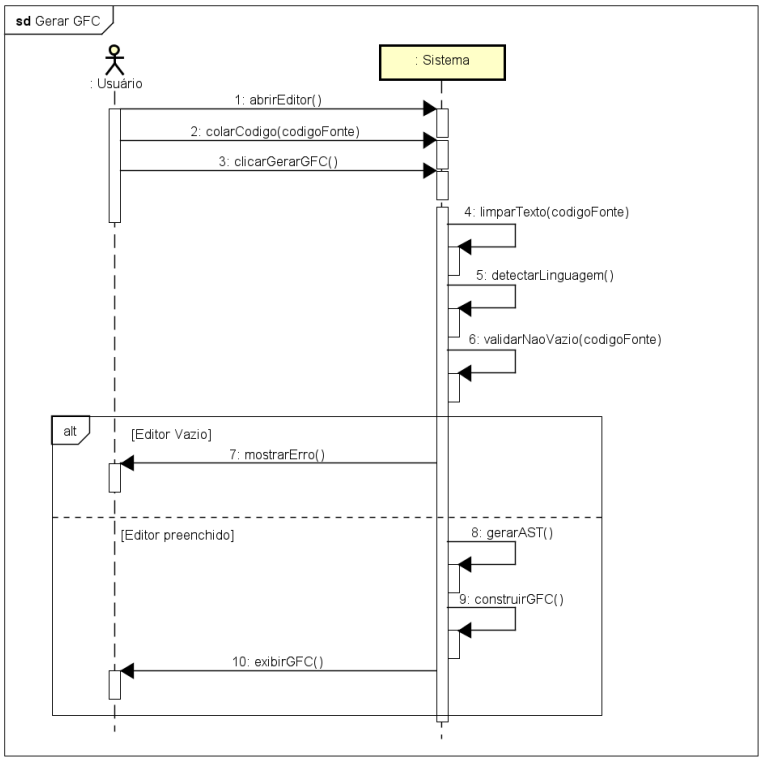

Figura 3. DSS: Gerar GFC

Contrato	Gerar GFC
Operação	GerarGFC(codigoFonte)
Referências cruzadas	UC2, UC3, UC4, UC5
Pré-condições	O editor precisa estar aberto e o código válido colado
Pós-condições	GFC gerado, salvo e exibido com mapeamento entre nós e linhas do código

A figura 4 ilustra a interação entre o Usuário e o Sistema DSS no processo de cálculo da cobertura estrutural do Grafo de Fluxo de Controle (GFC) e geração do relatório de elementos não cobertos. Inicialmente, o usuário acessa o painel de cobertura e seleciona o GFC que deseja analisar. Em seguida, solicita o cálculo das métricas, podendo opcionalmente importar resultados de execução de testes a partir de arquivos ou integrações externas. O sistema valida os dados importados e, caso sejam inválidos, exibe uma mensagem de erro e interrompe a execução. Se os dados forem válidos, o sistema carrega as execuções vinculadas ao GFC e verifica se há informações disponíveis. Com dados válidos, o sistema inicia o cálculo das métricas estruturais, computando cobertura de nós, arestas, decisões e, opcionalmente, condições (MC/DC) e caminhos. Após consolidar os indicadores, o sistema salva os resultados, aplica o heatmap sobre o GFC e exibe o painel de métricas com a visualização correspondente. Posteriormente, o usuário pode solicitar a geração do relatório de elementos não cobertos. O sistema consulta os nós e arestas sem cobertura, gera o relatório detalhado com os itens não exercitados, persiste o documento e apresenta a visualização ao usuário.

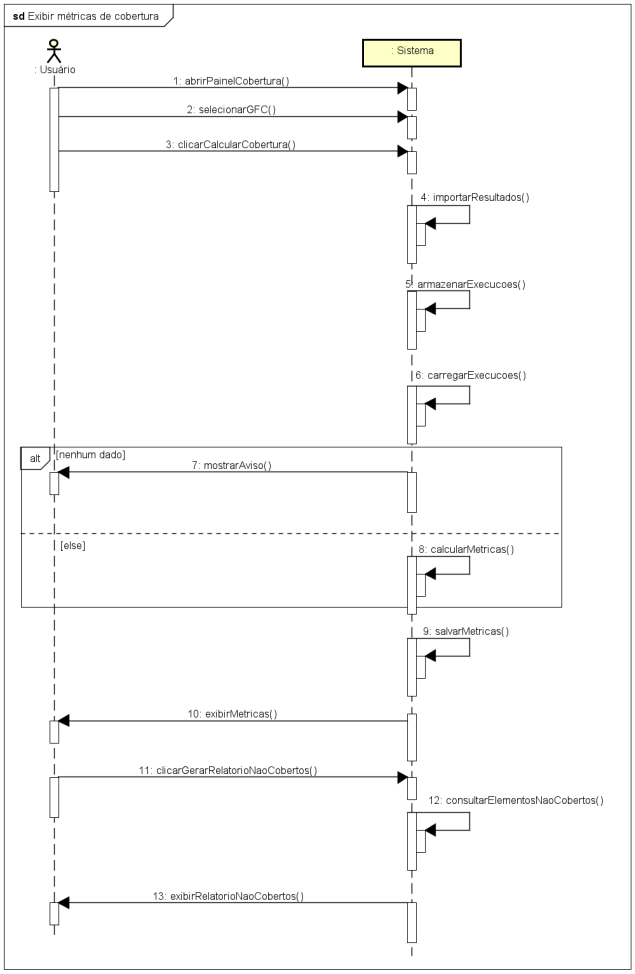

Figura 4. DSS: Exibir métricas de cobertura

Contrato	Exibir métricas de cobertura
Operação	CalcularCobertura() e GerarRelatorioNaoCobertos()
Referências cruzadas	UC10, UC9, UC7
Pré-condições	O editor precisa estar aberto e o código válido colado
Pós-condições	GFC gerado, salvo e exibido com mapeamento entre nós e linhas do código

A figura 5 apresenta a sequência de interações entre o Usuário e o Sistema DSS durante o processo de criação e edição do Grafo de Causa e Efeito (GCE). O fluxo inicia com a abertura do editor funcional, na qual o usuário pode optar por criar um novo grafo ou carregar um existente. Em seguida, o usuário adiciona causas e efeitos, estabelecendo relações entre eles por meio de ligações lógicas. Após definir a estrutura inicial, o usuário aplica operadores lógicos (AND, OR, NOT) e restrições (E, I, O, R, M) para refinar o comportamento do modelo. O sistema atualiza o resumo de regras e re-renderiza o grafo a cada modificação. Quando o usuário solicita a validação, o sistema executa verificações de consistência, detectando causas sem efeito, efeitos inalcançáveis ou conflitos entre restrições. Os resultados são exibidos em um relatório de validação, permitindo correções no modelo. Por fim, o usuário salva o grafo validado, e o sistema confirma a persistência do artefato.

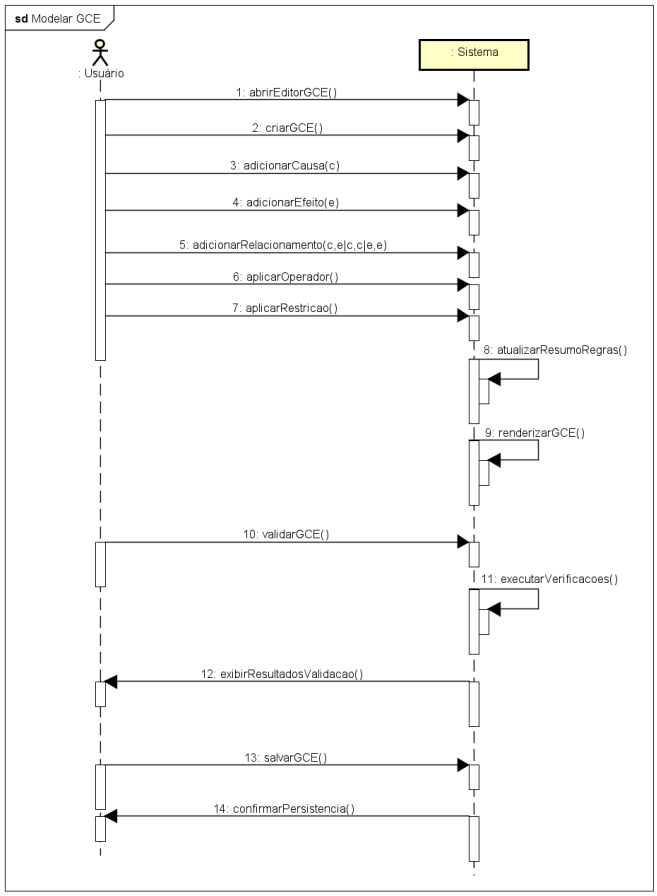

Figura 5. DSS: Modelar GCE

Contrato	GCE (modelagem/operadores/validação)
Operação	criarGCE(), adicionarCausa(c), adicionarEfeito(e), adicionarRelacionamento(c,e|c,c|e,e), aplicarOperador(), aplicarRestricao(), validarGCE(), salvarGCE()
Referências cruzadas	UC11, UC15
Pré-condições	O editor GCE precisa estar aberto

Pós-condições	GCE persistido, regras aplicadas, relatório de validação disponível

A figura 6 representa a interação entre o Usuário e o Sistema DSS durante o processo de derivação da Tabela de Decisão a partir de um Grafo de Causa e Efeito (GCE). O fluxo inicia quando o usuário abre o GCE previamente modelado e seleciona a opção de gerar a Tabela de Decisão. O sistema então interpreta o grafo, identificando todas as combinações possíveis de causas e efeitos (mintermos). Em seguida, aplica uma etapa de minimização, eliminando redundâncias lógicas e simplificando as combinações de condições. Após a minimização, o sistema constrói a tabela, organizando as condições (causas) nas colunas e as ações (efeitos) nas linhas correspondentes. Cada linha representa uma regra derivada do GCE, descrevendo os cenários de ativação das ações de acordo com as combinações de causas. Ao final, a Tabela de Decisão é persistida no repositório e exibida ao usuário, possibilitando revisão, exportação ou posterior uso na geração de casos de teste funcionais.

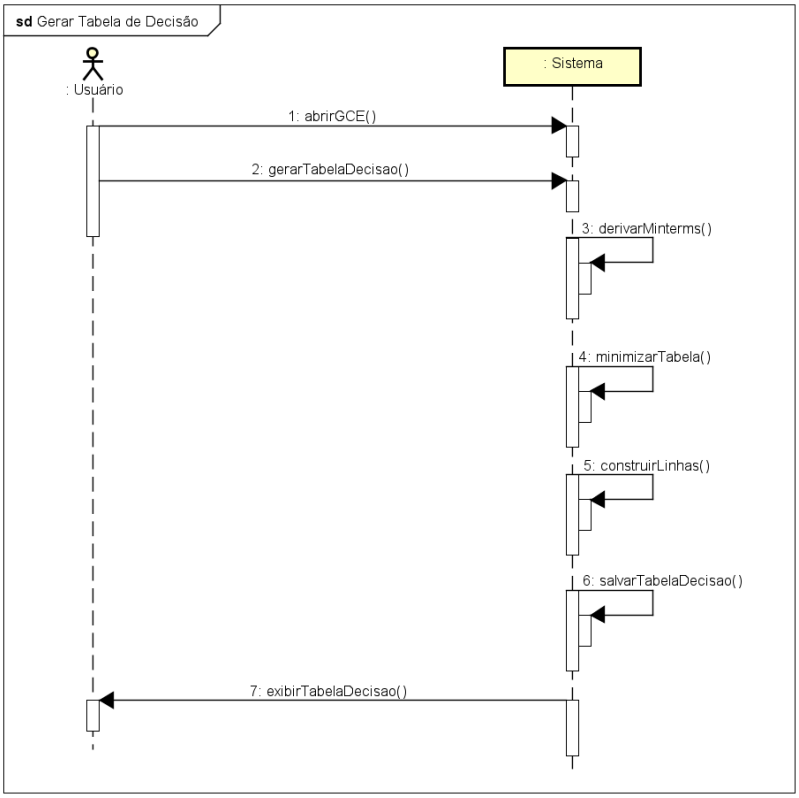

Figura 6. DSS: Gerar tabela de decisão

Contrato	Tabela de Decisão
Operação	GerarTabelaDecisao()
Referências cruzadas	UC13
Pré-condições	O GCE precisa estar pronto e validado
Pós-condições	Tabela persistida, traço causas/efeitos → condições/ações

A figura 7 ilustra a interação entre o Usuário e o Sistema DSS durante o processo de geração de casos de teste funcionais a partir da Tabela de Decisão. O fluxo tem início quando o usuário acessa a Tabela de Decisão previamente gerada e seleciona a opção de criar casos de teste. O sistema interpreta cada linha da tabela que representa uma regra composta por combinações de condições e ações e transforma essas informações em casos de teste individuais. Durante essa etapa, o sistema instancia os casos com base em um template definido, atribuindo a cada caso os parâmetros de entrada (condições verdadeiras/falsas), as ações esperadas e as saídas correspondentes. Em seguida, o conjunto de casos gerados é consolidado em uma suíte de testes funcionais. Após a geração, o sistema persiste a suíte de testes e exibe-a ao usuário, permitindo visualização, edição e exportação.

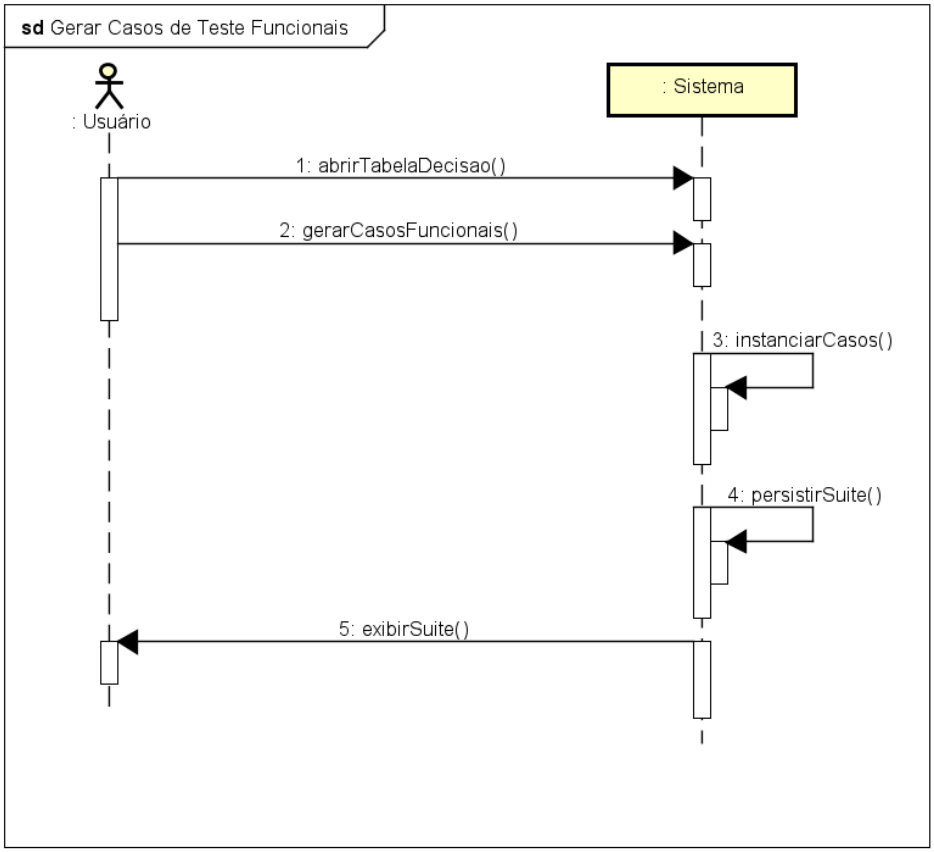

Figura 7. DSS: Gerar casos de teste funcionais

Contrato	Casos de Testes Funcionais
Operação	GerarCasosFuncionais()
Referências cruzadas	UC14
Pré-condições	A tabela de decisão precisa estar gerada
Pós-condições	Suite de testes persistida, traço linha→caso

A figura 8 descreve a interação entre o Usuário e o Sistema DSS no processo de exportação de artefatos gerados pela aplicação, como o Grafo de Causa e Efeito (GCE), a Tabela de Decisão, os Casos de Teste Funcionais ou o GFC. O fluxo inicia quando o usuário acessa a funcionalidade de exportação a partir da visualização de um artefato existente. Em seguida, o usuário seleciona o formato desejado para a saída — imagem (PNG ou SVG) ou dados estruturados (CSV ou JSON). O sistema identifica o tipo do artefato, converte internamente o conteúdo para o formato selecionado e gera o arquivo correspondente. Após a conversão, o sistema prepara o arquivo para download e notifica o usuário sobre a disponibilidade da exportação.

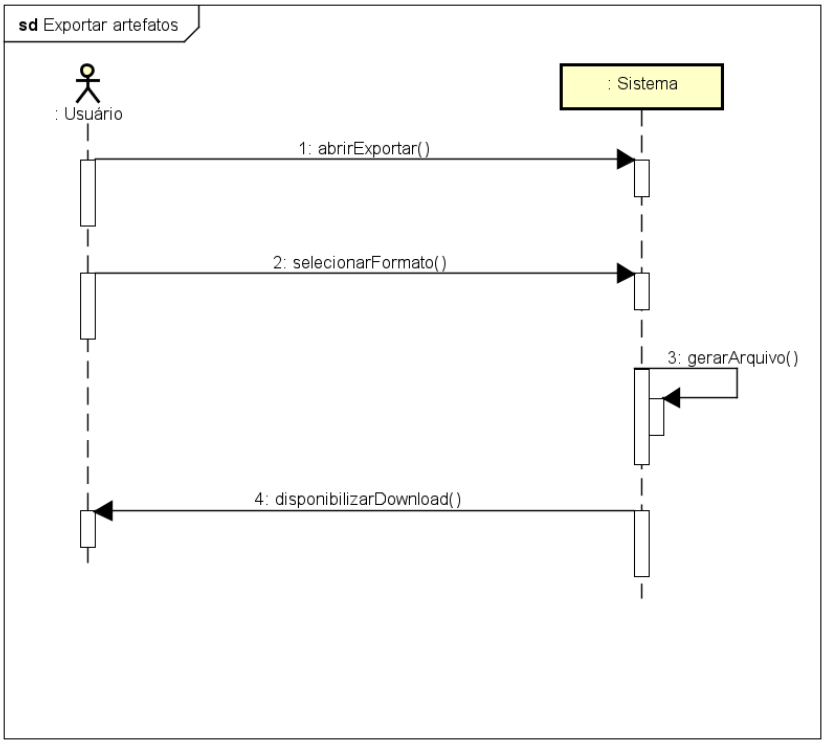

Figura 8. DSS: Exportar artefatos

Contrato	Exportar artefatos
Operação	gerarArquivo(), selecionarFormato()
Referências cruzadas	UC6, UC12
Pré-condições	O artefato precisa estar carregado
Pós-condições	Arquivo gerado e acessível

3. Modelos de Projeto

3.1 Modelo de Domínio

Nesta seção, apresenta-se o Modelo de Domínio, utilizando a notação da Linguagem de Modelagem Unificada (UML, do inglês *Unified Modeling Language*). Nesse caso, o modelo é ilustrado como um conjunto de diagramas de classes sem definição de operações ou assinaturas de métodos (LARMAN, 2005).

A Figura 9 apresenta o diagrama de domínio do sistema de Geração e Análise de Grafos de Teste, o qual contém as principais classes *Project, CodeArtifact, ControlFlowGraph, CauseEffectGraph, DecisionTable, FunctionalTestCase, Result, ConsolidatedReport, CoverageMetric, CoverageReport, MetricsDashboard, AbstractSyntaxTree, ExecutionData,* e *User*, responsáveis por representar os elementos estruturais, funcionais e analíticos da plataforma. As classes *Project* e *CodeArtifact* armazenam os artefatos de código que dão origem aos grafos de fluxo de controle (*ControlFlowGraph*) e aos grafos de causa e efeito (*CauseEffectGraph*), utilizados para geração de métricas e casos de teste funcionais (*FunctionalTestCase*), cujas execuções produzem resultados (*Result*). As classes *CoverageMetric, CoverageReport* e *ConsolidatedReport* permitem o cálculo e consolidação das métricas estruturais e funcionais, enquanto *MetricsDashboard* possibilita a visualização integrada dessas informações. Por fim, as classes auxiliares *AbstractSyntaxTree*, *ExecutionData* e *ImageFormat* dão suporte à análise do código-fonte, execução de testes e exportação de resultados. Todas essas classes são tratadas na plataforma durante a execução do sistema, sendo responsáveis pelo armazenamento, processamento e análise das informações referentes às funcionalidades do usuário e à geração automatizada de grafos e métricas de teste.

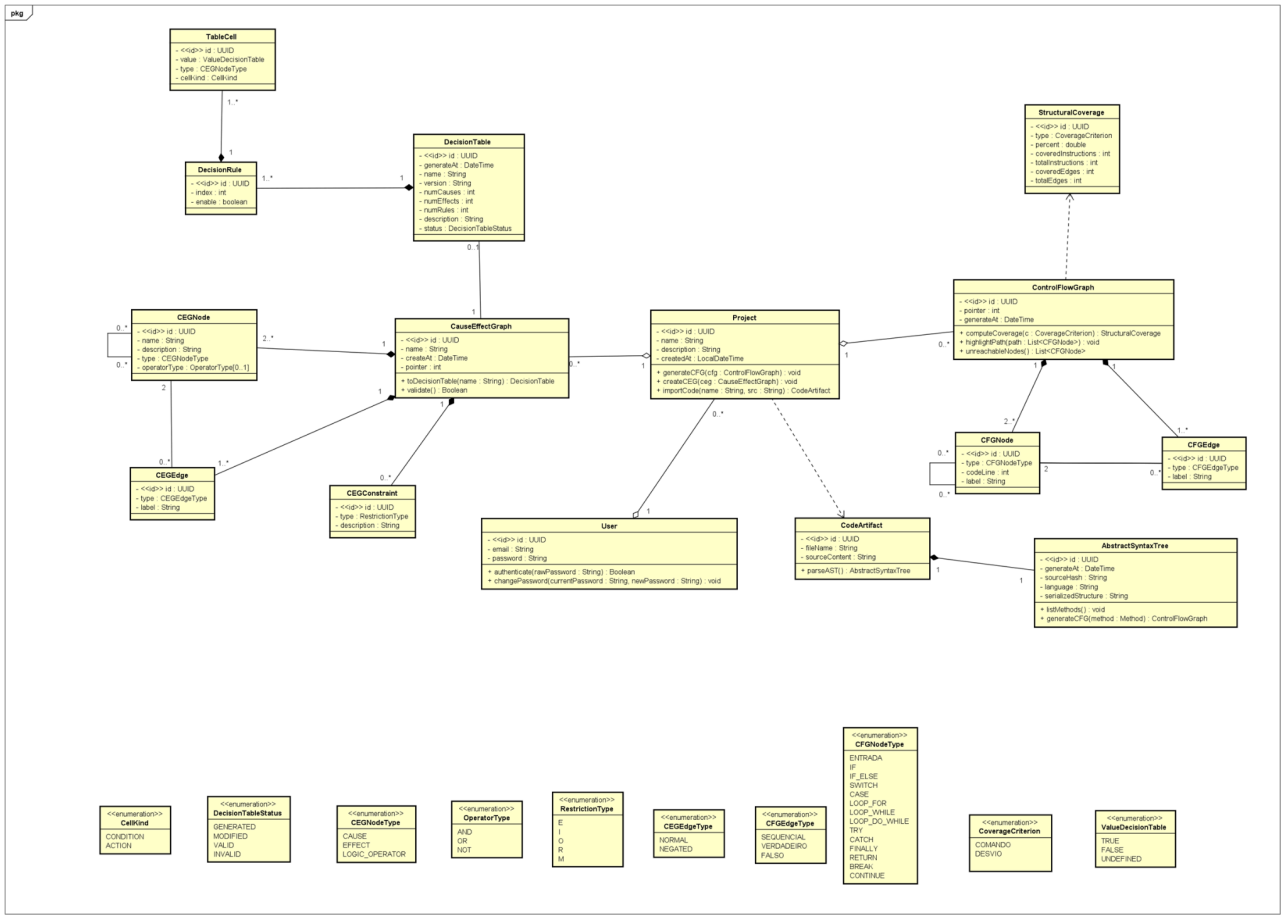

Figura 9. Modelo de Domínio

3.2 Diagramas de Sequência

Esta seção apresenta os Diagramas de Sequência, que descrevem o comportamento dinâmico do sistema, representando a ordem temporal das interações entre os atores externos, as fronteiras (interfaces), os controles (módulos de processamento) e as entidades (dados persistidos). Cada diagrama ilustra como as mensagens são trocadas entre esses elementos para realizar um determinado fluxo funcional, evidenciando o encadeamento lógico das operações dentro do sistema.

No contexto deste projeto, os diagramas de sequência foram elaborados de forma a detalhar os principais fluxos de execução das funcionalidades da plataforma, contemplando tanto as ações estruturais (relacionadas ao Grafo de Fluxo de Controle – GFC) quanto as funcionais (relacionadas ao Grafo de Causa e Efeito – GCE), além dos processos complementares, como a geração de tabelas de decisão, a criação de casos de teste e a exportação de artefatos.

Cada diagrama apresenta os módulos internos do sistema (como editores, motores de análise e repositórios) e suas interações com serviços externos, tais como conectores de repositórios de código, validadores lógicos e bibliotecas de minimização. Também são destacadas as fronteiras de

comunicação entre o software e os atores externos, mostrando claramente onde as mensagens ultrapassam os limites do sistema.

A figura 10 representa as interações entre o Usuário, as fronteiras externas e os módulos internos do sistema no processo de geração, análise e visualização do Grafo de Fluxo de Controle (GFC). Ele cobre os casos de uso UC2 – Fornecer código fonte, UC4 – Importar código de repositório, UC10 – Calcular métricas de cobertura estrutural, UC9 – Relatório de elementos não cobertos, UC8 – Mapear casos de teste aos elementos do GFC e UC5 – Gerar Grafo de Fluxo de Controle.

O fluxo inicia quando o Usuário acessa o módulo de GFC por meio da interface web (UI/WebClient). Nesse momento, há duas alternativas: o usuário pode colar um trecho de código no editor ou importar arquivos diretamente de um repositório remoto, utilizando o conector externo VCSConnector(API). Em ambos os casos, o código é processado pelo EditorDeCodigo e encaminhado ao módulo AnaliseDeCodigo, responsável por gerar a Árvore Sintática Abstrata (AST).

A geração da AST é realizada por um serviço externo (ParserService), configurado como uma fronteira de integração. O resultado dessa análise é devolvido ao sistema, que então constrói o GFC e o persiste no RepositorioCFG(DB). Em seguida, o grafo é renderizado por meio do motor gráfico externo (RenderizadorGraficos), sendo exibido visualmente na interface para o usuário (UC5).

Posteriormente, o usuário pode solicitar o cálculo das métricas de cobertura estrutural. O sistema acessa o CoverageImporter(API) para importar resultados de execução de testes, processa os dados e aplica um heatmap sobre o grafo, destacando as partes do código cobertas e não cobertas pelos testes. A partir dessas métricas, o usuário pode ainda gerar o relatório de elementos não cobertos e realizar o mapeamento de casos de teste aos elementos do GFC, o que permite identificar visualmente quais partes do código são exercitadas por cada caso de teste.

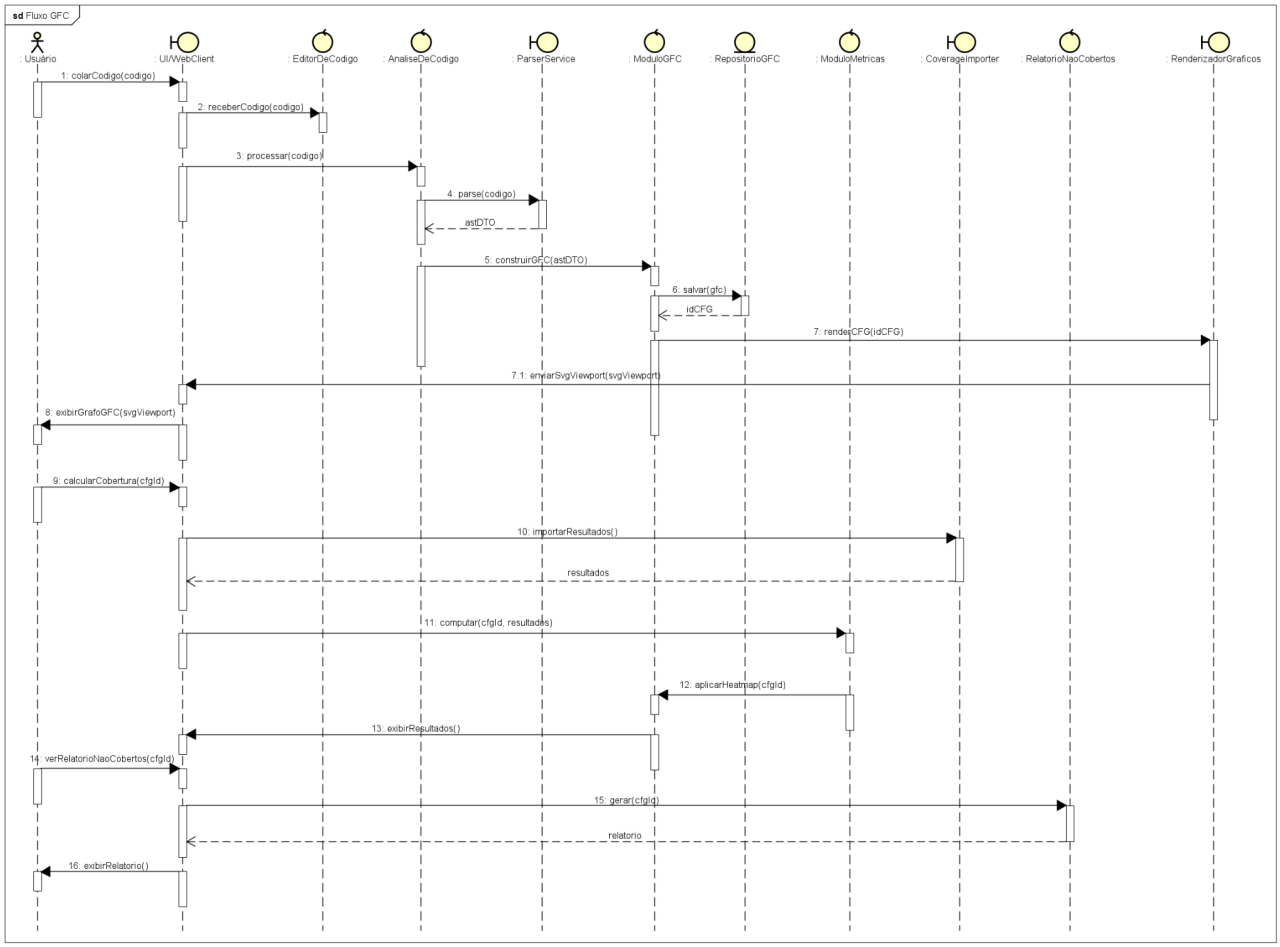

Figura 10. Diagrama de Sequência: Fluxo GFC

A figura 11 apresenta as interações entre o Usuário, as fronteiras externas e os módulos internos do sistema durante o processo de modelagem, aplicação de operadores e validação do Grafo de Causa e Efeito (GCE). Ele cobre os casos de uso UC11 – Modelar GCE e UC15 – Validar consistência do GCE.

O fluxo inicia quando o Usuário acessa o editor de GCE por meio da interface web (UI/WebClient), podendo criar um novo modelo ou abrir um grafo já existente. Dentro do editor, o usuário adiciona causas e efeitos, estabelece conexões entre eles e aplica operadores lógicos (AND, OR, NOT) e restrições do tipo E, I, O, R e M. Essas interações são processadas pelo EditorGCE, que mantém o estado do grafo e envia as alterações ao MotorDeRegra responsável por recalcular as dependências e atualizar o resumo lógico apresentado na interface.

Quando o usuário solicita a validação do modelo, o UI/WebClient encaminha o grafo ao serviço externo Solver/Validador(API), representado como uma fronteira de integração. Esse serviço executa verificações de consistência para detectar erros como causas não utilizadas, efeitos inalcançáveis e conflitos de restrições. O relatório de validação retornado é repassado ao

EditorGCE, que o formata e envia de volta à interface, permitindo que o usuário visualize as inconsistências diretamente no grafo e realize ajustes no modelo.

Por fim, o EditorGCE salva o estado atualizado do grafo no RepositorioArtefatos(DB), assegurando que as modificações e resultados de validação fiquem armazenados para uso posterior.

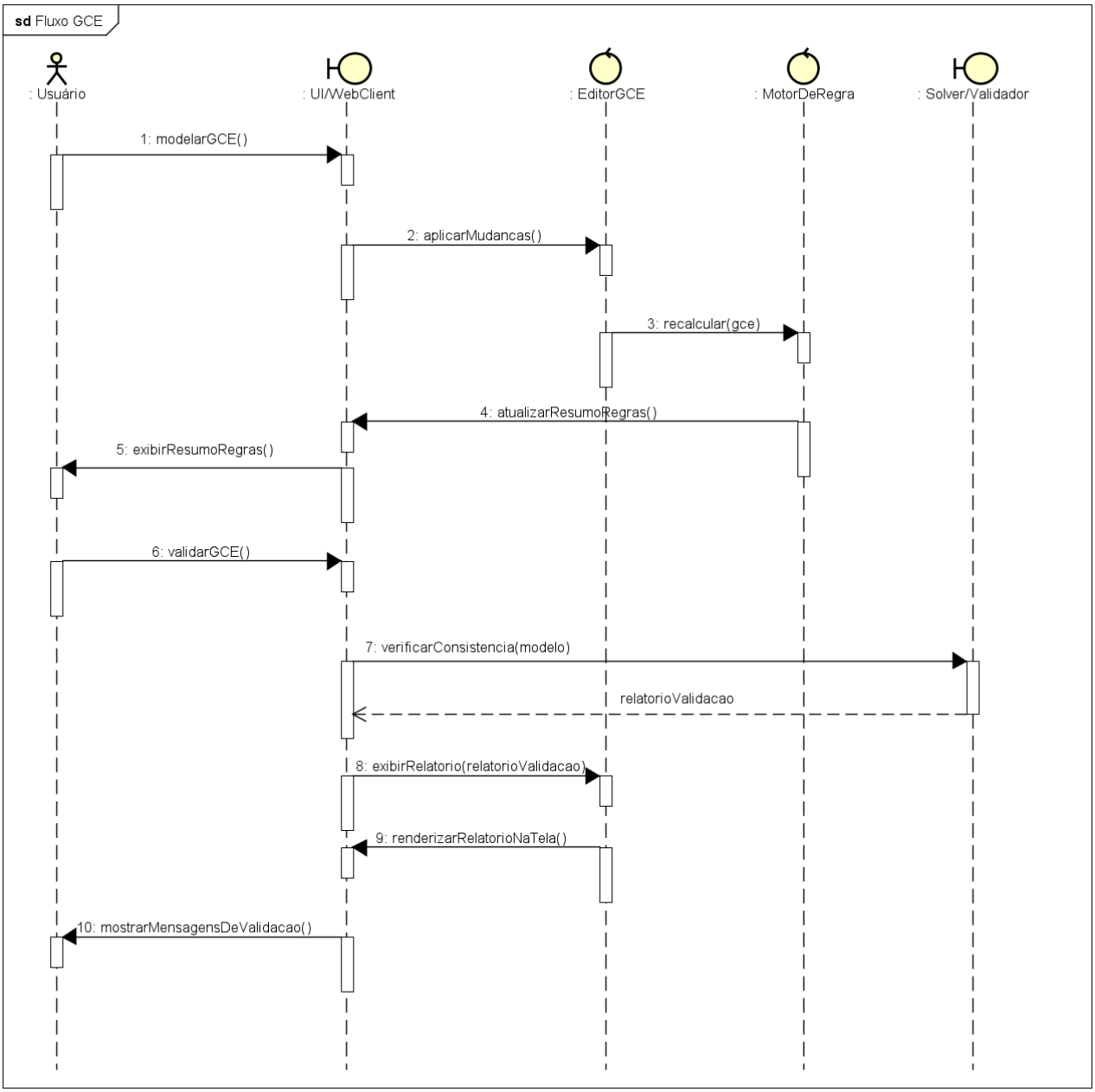

Figura 11. Diagrama de Sequência: Fluxo GCE

A Figura 12 descreve as interações entre o Usuário, as fronteiras externas e os módulos internos do sistema durante as etapas de derivação da Tabela de Decisão a partir do GCE e geração dos Casos

de Teste Funcionais. Ele cobre os casos de uso UC13 – Converter GCE em Tabela de Decisão e UC14 – Gerar casos de teste funcionais.

O fluxo inicia quando o Usuário, após modelar e validar o GCE, solicita a geração da Tabela de Decisão por meio da interface (UI/WebClient). O módulo DerivadorTabela interpreta o grafo, identifica as combinações lógicas possíveis de causas e efeitos (mintermos) e envia esses dados para a biblioteca externa MinimizationLib(API) — uma fronteira responsável por realizar a minimização lógica das combinações, removendo redundâncias e simplificando o conjunto de regras. O resultado é devolvido ao DerivadorTabela, que constrói a tabela final a partir do conjunto minimizado e a salva no RepositorioArtefatos(DB). Em seguida, a tabela é carregada e exibida ao usuário na interface, permitindo sua análise e edição visual.

Na sequência, o usuário pode acionar a funcionalidade de geração de casos de teste funcionais. O UI/WebClient solicita ao módulo GeradorDeTestes que instancie casos com base na Tabela de Decisão e em um template selecionado. O módulo processa cada linha da tabela, definindo as condições de entrada e as ações esperadas, e constrói a SuiteDeTestes, que é então armazenada no RepositorioArtefatos(DB). Após a persistência, a suíte é carregada e apresentada ao usuário, contendo os casos de teste devidamente derivados das regras funcionais.

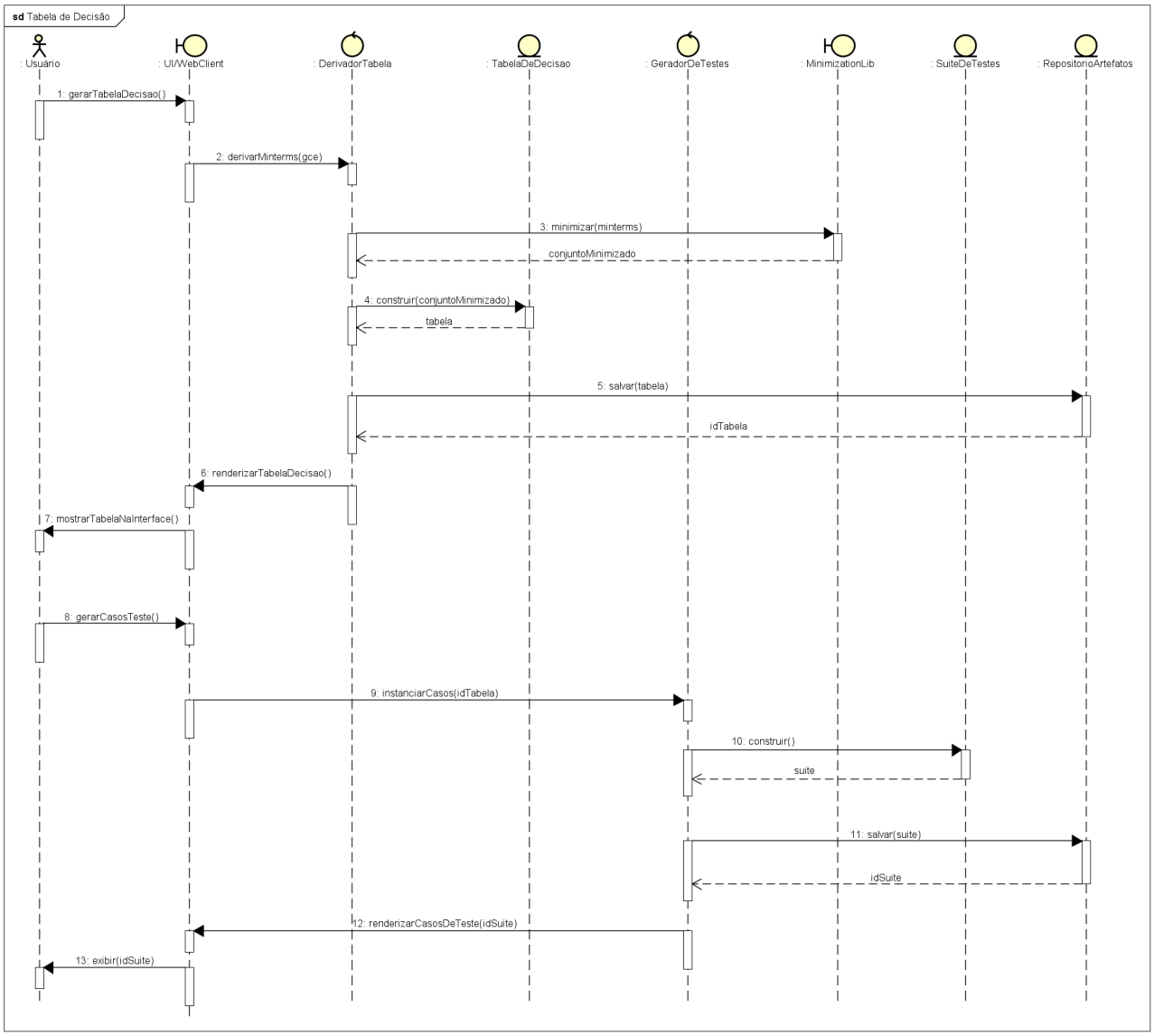

Figura 12. Diagrama de Sequência: Tabela de Decisão

A figura 13 apresenta o processo de exportação de artefatos do sistema, ilustrando a interação entre o Usuário, a interface web e o serviço externo de exportação. Ele cobre os casos de uso UC12 e UC6, que se referem a exportar artefatos (PNG, SVG).

O fluxo inicia quando o Usuário, a partir do painel do sistema, seleciona a opção de exportar um artefato já gerado — como um Grafo de Fluxo de Controle (GFC), um Grafo de Causa e Efeito (GCE), uma Tabela de Decisão ou uma Suíte de Testes. Pela interface (UI/WebClient), o usuário define o formato de saída desejado (por exemplo, PNG ou SVG para imagens, CSV ou JSON para dados estruturados) e confirma a solicitação de exportação.

A interface encaminha o pedido ao serviço externo ExportService(API), que atua como uma fronteira de integração responsável por converter o artefato selecionado para o formato especificado.

Esse serviço processa o conteúdo, gera o arquivo correspondente e retorna ao sistema uma URL de download.

Por fim, a UI/WebClient disponibiliza o arquivo ao usuário, permitindo que ele realize o download diretamente pela interface.

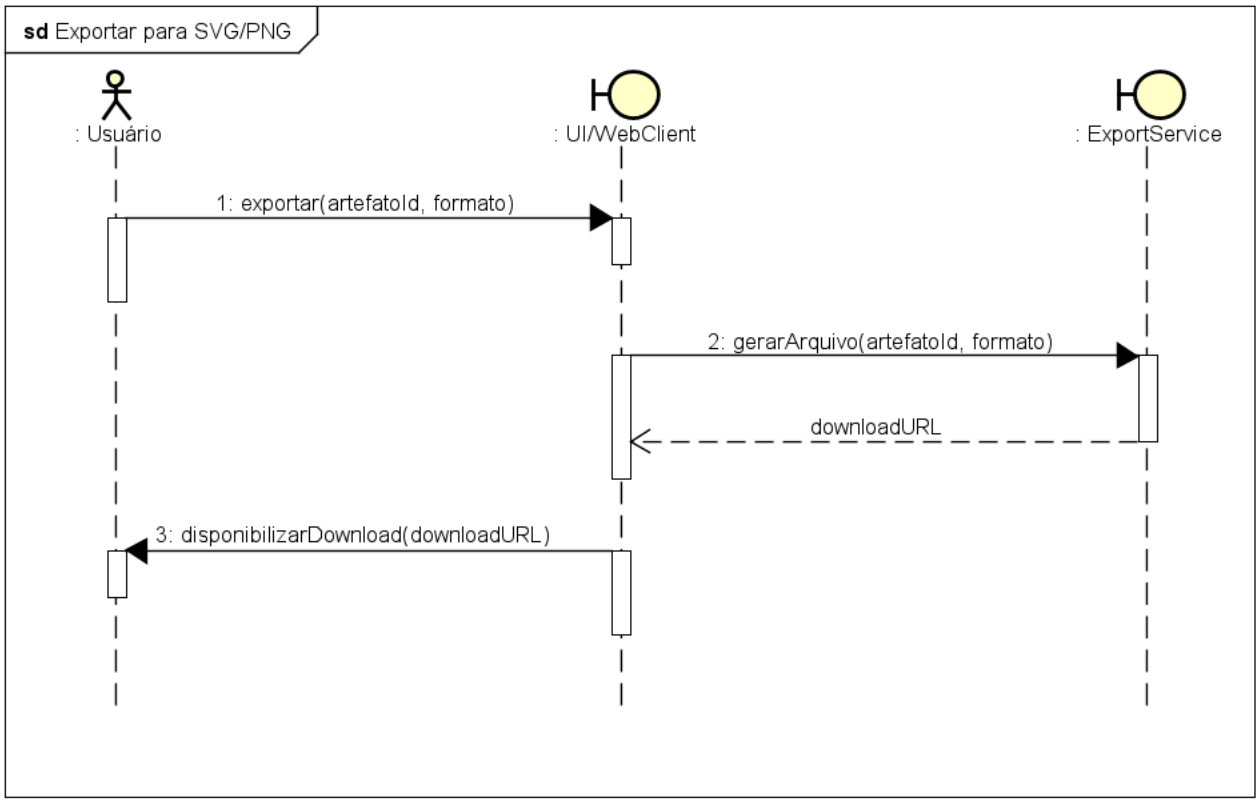

Figura 13. Diagrama de Sequência: Exportar artefatos

3.3 Diagramas de Comunicação

O diagrama de comunicação apresentado na Figura 14 representa as interações entre os objetos responsáveis pela execução do caso de uso UC5, bem como os casos complementares UC2, UC10, UC7 e UC9. Assim como no diagrama de sequência correspondente, o fluxo tem início com a ação do usuário ao colar o código-fonte na interface web e prossegue pelas trocas de mensagens entre os módulos de análise, parsing, geração e renderização do grafo, culminando na exibição do grafo de fluxo de controle e nos cálculos de cobertura realizados pelo módulo de métricas. O diagrama evidencia as relações estruturais entre as classes participantes do processo, destacando a comunicação entre os componentes *UIWebClient*, AnaliseDeCodigo, *ParserService*, ModuloGFC, ModuloMetricas e RenderizadorGraficos, responsáveis pela orquestração completa da geração, cálculo e exibição das métricas do GFC.

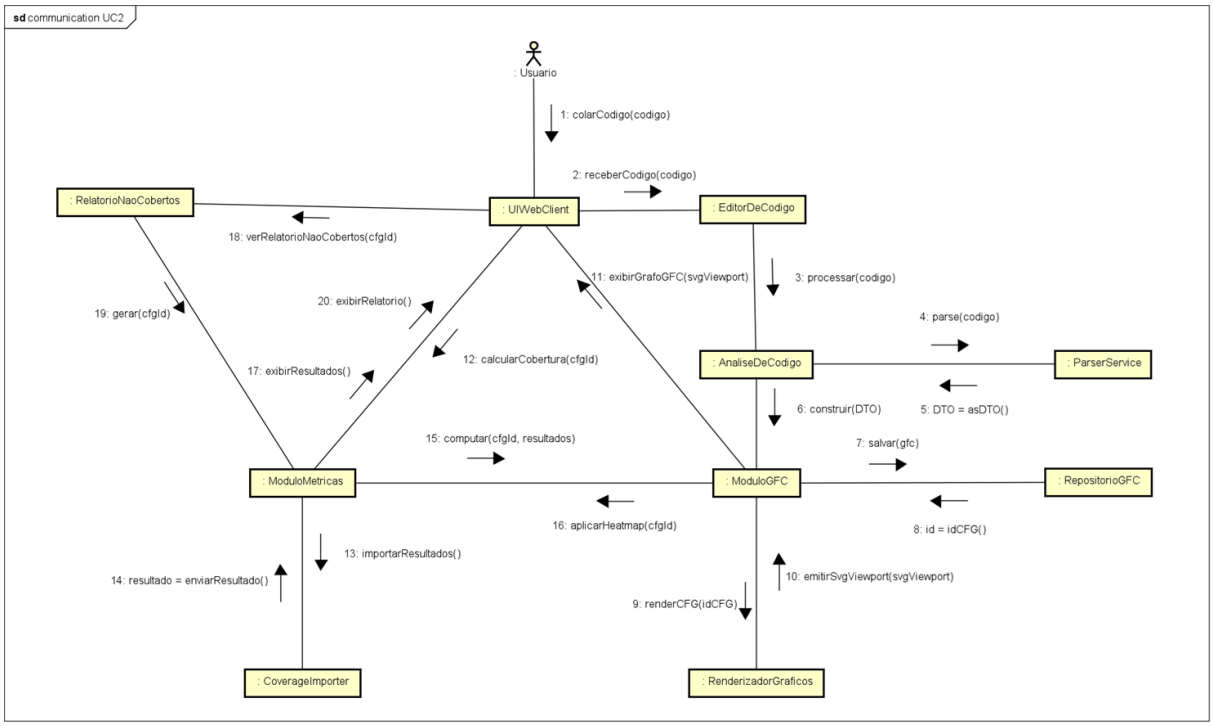

Figura 14. Diagrama de Comunicação: Fluxo GFC

3.4 Arquitetura

A Figura 15 representa a organização estrutural dos principais pacotes e subsistemas da aplicação, evidenciando as camadas de responsabilidade e as dependências entre elas. A arquitetura segue um modelo modular em camadas, que favorece a separação de preocupações, a reutilização de componentes e a manutenção evolutiva do sistema.

Na camada superior, o subsistema UI (*User Interface*) compreende os elementos responsáveis pela interação com o usuário, incluindo as *views* e components que realizam a comunicação direta com o usuário final. Essa camada se comunica unicamente com o subsistema AppGateway, que atua como a fronteira entre a interface e o núcleo da aplicação.

O subsistema AppGateway centraliza os pontos de entrada do sistema, sendo responsável pelo controle de acesso, roteamento de requisições e gerenciamento de propriedades globais. Ele contém módulos como Controllers, que expõem os serviços ao front-end; Auth, responsável por autenticação e autorização; e Properties, que mantém configurações de ambiente e parâmetros de execução.

A camada intermediária Core implementa as regras de negócio do GraphTest. Ela é composta por dois módulos principais: Services, que encapsulam a lógica funcional da aplicação (como geração de grafos, cálculos de métricas e manipulação de artefatos), e Helpers, que fornecem funções auxiliares e utilitárias para suporte às operações dos serviços e controladores. O Core é o coração da aplicação e mantém dependência direta das camadas Domain e Infrastructure.

O subsistema Domain modela os conceitos centrais do domínio da aplicação, assegurando a consistência semântica dos artefatos manipulados. Ele é composto pelos pacotes Models, DTO, Constants, Enums e Utils, que representam, respectivamente, as entidades do sistema, objetos de transferência de dados, constantes de domínio, enumerações e utilitários de manipulação. O Domain é utilizado pelo Core como base para as operações e pelo Infrastructure para persistência e integração.

Por fim, a camada Infrastructure fornece os recursos de suporte técnico e integração externa, contendo os módulos API (para comunicação com serviços externos) e Database (para persistência de dados). Essa camada abstrai detalhes de implementação de acesso a dados, permitindo que o restante do sistema opere de forma independente de tecnologias específicas.

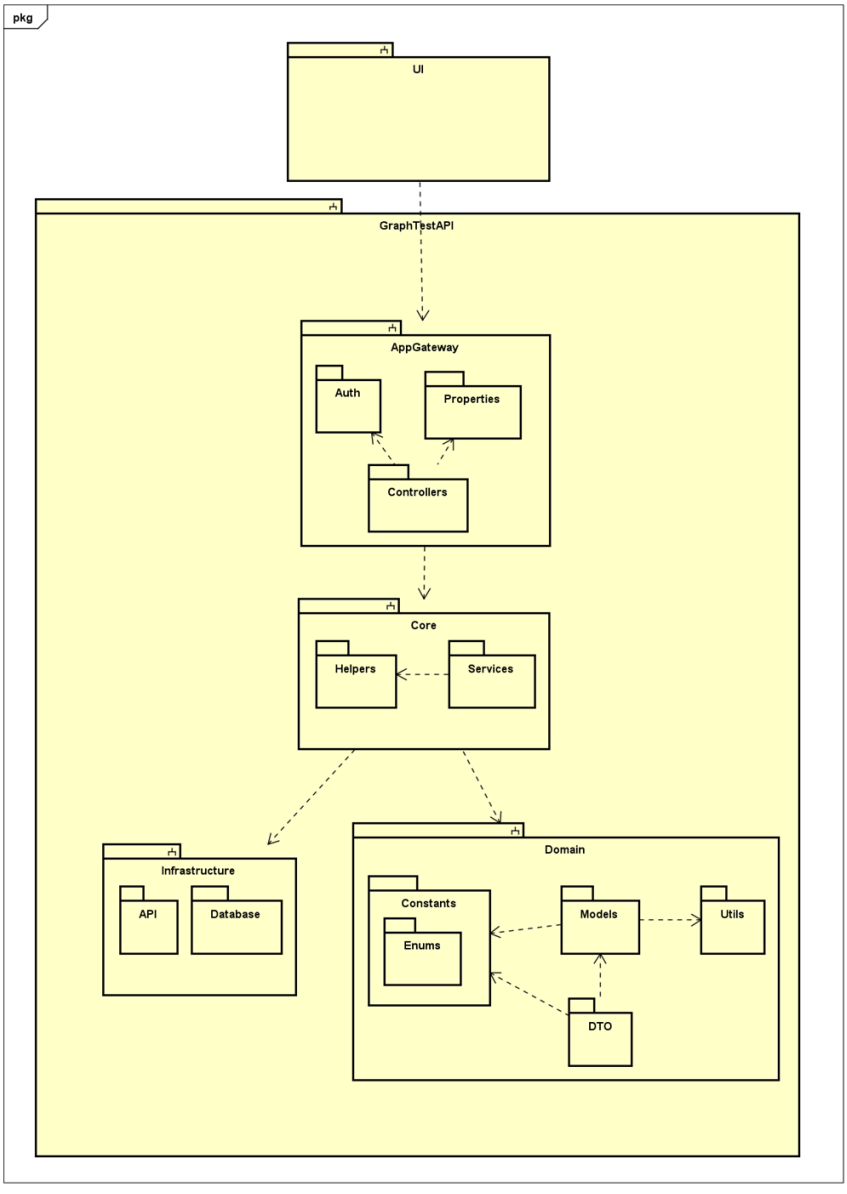

Figura 15. Diagrama de Arquitetura

3.5 Diagramas de Estados

A Figura 16, Diagrama de Estados do Ciclo de Vida de um GFC (Grafo de Fluxo de Controle) representa o comportamento dinâmico do artefato ao longo de sua criação, análise e cálculo de cobertura. O diagrama descreve as transições de estado desde a disponibilização do código-fonte até a obtenção do relatório final de cobertura, evidenciando as etapas internas do sistema e as ações executadas em cada momento.

O ciclo inicia no estado Código Disponível, em que o sistema registra a fonte do código inserido pelo usuário (entry/registrarFonte()). A partir desse ponto, o usuário pode acionar o comando solicitar gerar GFC, o que leva o sistema ao estado Parsing, no qual o código é analisado sintaticamente (do/parseToAST()). Caso a análise resulte em erro, a transição ocorre para o estado Exibindo Relatório de Erro, onde são apresentados os diagnósticos e mensagens de falha (entry/exibirDiagnósticos()). O usuário pode então corrigir o código, retornando ao estado Código Corrigido e reiniciando o processo.

Quando o parsing é bem-sucedido ([sucesso]), o sistema passa ao estado Gerando GFC, responsável pela construção efetiva do grafo de fluxo de controle (do/construirCFG()) e sua persistência (exit/persistirGFC()). Em seguida, o artefato entra no estado GFC Gerado, onde é renderizado graficamente na interface (entry/renderizarGrafo()), tornando-se visualizável pelo usuário.

A partir desse ponto, o diagrama apresenta uma bifurcação condicional: caso existam resultados de cobertura disponíveis ([cobertura disponível]), o sistema avança para o estado Calcular Cobertura, no qual são computadas as métricas de cobertura estrutural (do/computarMetricasDeCobertura()) e o relatório correspondente é persistido (exit/persistirRelatorio()). Se não houver dados de cobertura ([cobertura indisponível]), o sistema entra no estado Aguardando Cobertura, exibindo uma mensagem informativa (entry/exibirMensagem("Sem dados de cobertura disponíveis")) e permanecendo em espera até que os resultados sejam importados (exit/importarResultados()).

Quando o processo de cálculo é concluído, o sistema alcança o estado Cobertura Calculada, responsável pela apresentação do relatório final de cobertura (do/exibirRelatorioDeCobertura()), encerrando o ciclo de vida do GFC.

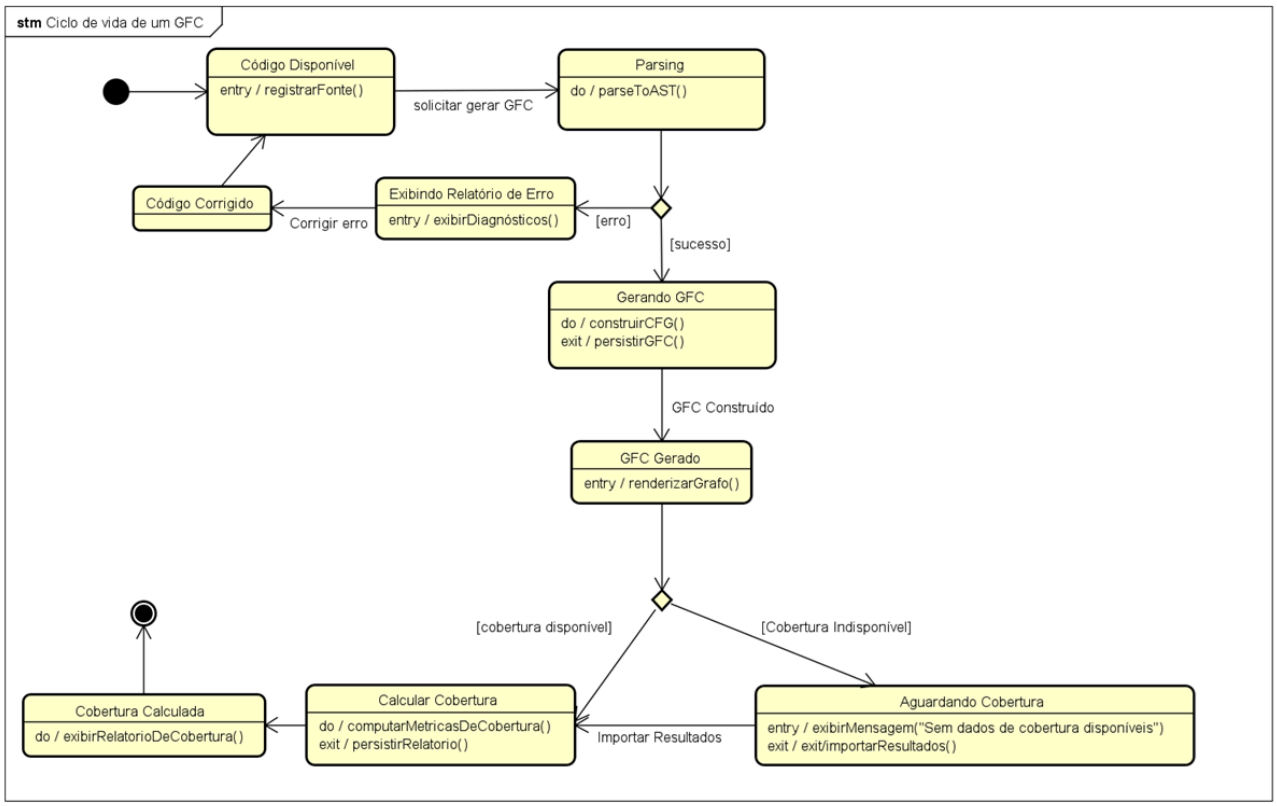

Figura 16. Diagrama de Estados do Ciclo de Vida do GFC

A Figura 17, Diagrama de Estados do Ciclo de Vida de um GCE (Grafo de Causa e Efeito) ilustra o comportamento dinâmico do artefato durante seu processo de criação, edição e validação. O diagrama descreve a sequência de estados que o modelo percorre desde sua inicialização até a persistência final, evidenciando as ações internas do sistema e as transições que resultam das interações do usuário ou da verificação automática de consistência lógica.

O ciclo inicia no estado Criado, momento em que o sistema inicializa a estrutura do modelo de GCE (entry/inicializarModelo()) e disponibiliza o editor gráfico ao usuário (do/exibirEditor()). Nesse estado, o modelo encontra-se vazio e apto para receber causas, efeitos e operadores lógicos, como AND, OR e NOT.

Ao adicionar elementos ao grafo, o sistema transita para o estado Em Edição, no qual o modelo passa a ser marcado como alterado (entry/marcarAlterado()) e o resumo das regras é atualizado continuamente conforme o usuário realiza modificações (do/atualizarResumoRegras()). Esse estado reflete o momento ativo de modelagem, em que o GCE é progressivamente construído e ajustado.

Quando o usuário solicita a verificação de consistência do modelo, ocorre a transição para o estado Validando. Nesse ponto, o sistema prepara o modelo para análise lógica (entry/prepararModeloParaSolver()) e executa os procedimentos de verificação (do/verificarConsistencia()), assegurando que todas as relações entre causas e efeitos estejam logicamente corretas e sem contradições.

O resultado dessa validação define a transição seguinte:

- Caso o modelo seja considerado consistente ([sim]), o sistema avança para o estado Salvo, onde o artefato é persistido em banco de dados (entry/persistirModelo()), encerrando o ciclo.

- Caso inconsistências sejam detectadas ([não]), o sistema transita para o estado Exibindo Relatório de Erro, apresentando ao usuário os diagnósticos e mensagens de falha (entry/exibirDiagnósticos()), permitindo a correção manual dos problemas e o retorno ao estado Em Edição.

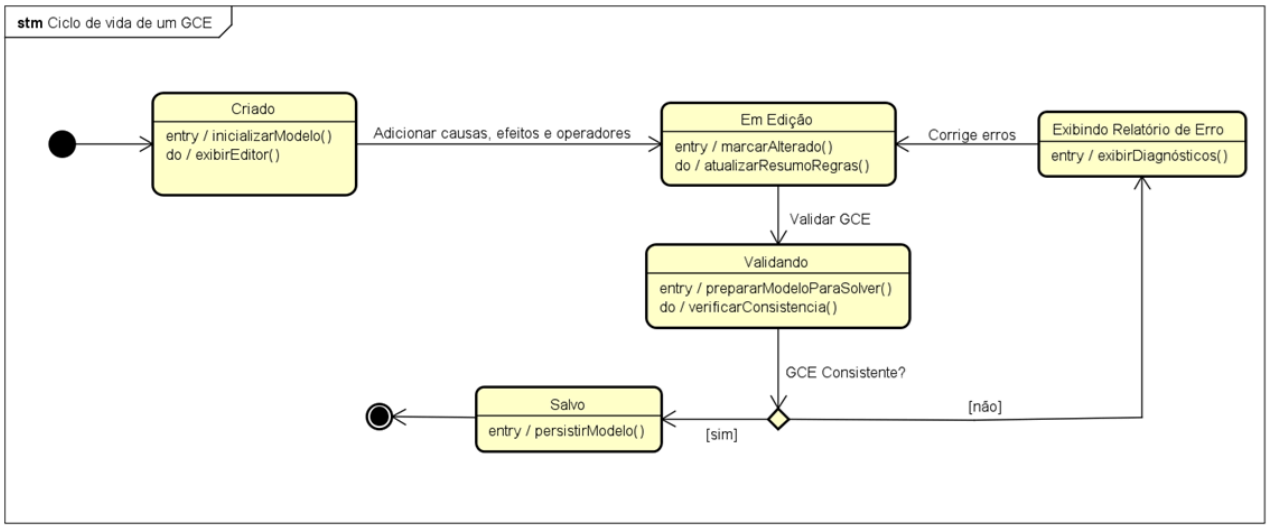

Figura 17. Diagrama de Estados do Ciclo de Vida do GCE

3.6 Diagrama de Componentes e Implantação.

Nesta seção são apresentados os diagramas de componentes e implantação da plataforma. O diagrama de componentes modela os principais componentes do sistema, apontando suas composições e dependências, bem como as interfaces que eles consomem e que eles proveem. Enquanto isso, o diagrama de implantação representa os recursos físicos e de software necessários para o sistema ser implantado adequadamente.

Na Figura 18 é apresentado o diagrama de componentes do sistema GraphTest. O principal componente representado é o Web Server (Spring Boot API), responsável por prover a API consumida pela aplicação *web*. Esse componente realiza o processamento do código-fonte *Java*, gera o Grafo de Fluxo de Controle e se comunica SGBD por meio do ORM, utilizando SQL para armazenamento de dados.

O Web App (Angular) representa o cliente do sistema, sendo responsável pela interação com o usuário. Ele consome os serviços expostos pelo Web Server e utiliza as bibliotecas CFG Viewer e CEG Editor, ambas baseadas no *Cytoscape.js*, para a visualização e edição dos grafos. Além disso, o módulo Import Module realiza a leitura de arquivos *.java* a partir do Sistema de Arquivos (Local .java / Saídas), enquanto o Export Module é responsável pela exportação das imagens dos grafos gerados.

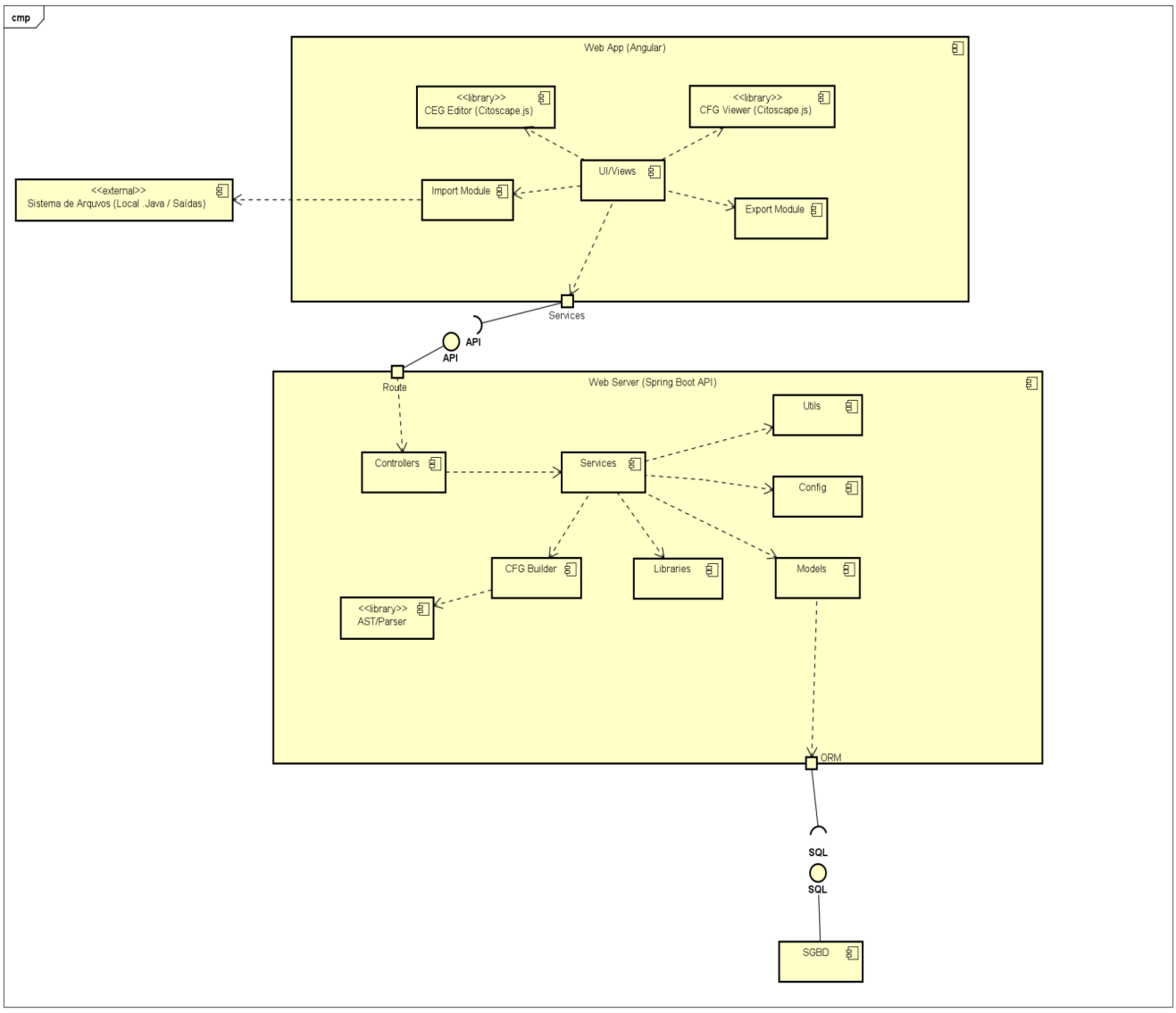

Figura 18. Diagrama de Componente

Na Figura 19 é apresentado o diagrama de implantação do sistema GraphTest. Nele são mostrados os nós de processamento utilizados. A aplicação *web* é acessada por meio de um *Web Browser* no *Client*, que obtém os arquivos estáticos do *Web Server – Angular App* via HTTPS e realiza requisições ao *Application Server – VM*, onde está implantada a *Backend Application (Spring Boot API)*. O *Application Server* comunica-se com o *Database Server* e o *File System* por meio do protocolo TCP/IP, responsáveis respectivamente pelo armazenamento de dados e pelos arquivos de entrada e saída da aplicação.

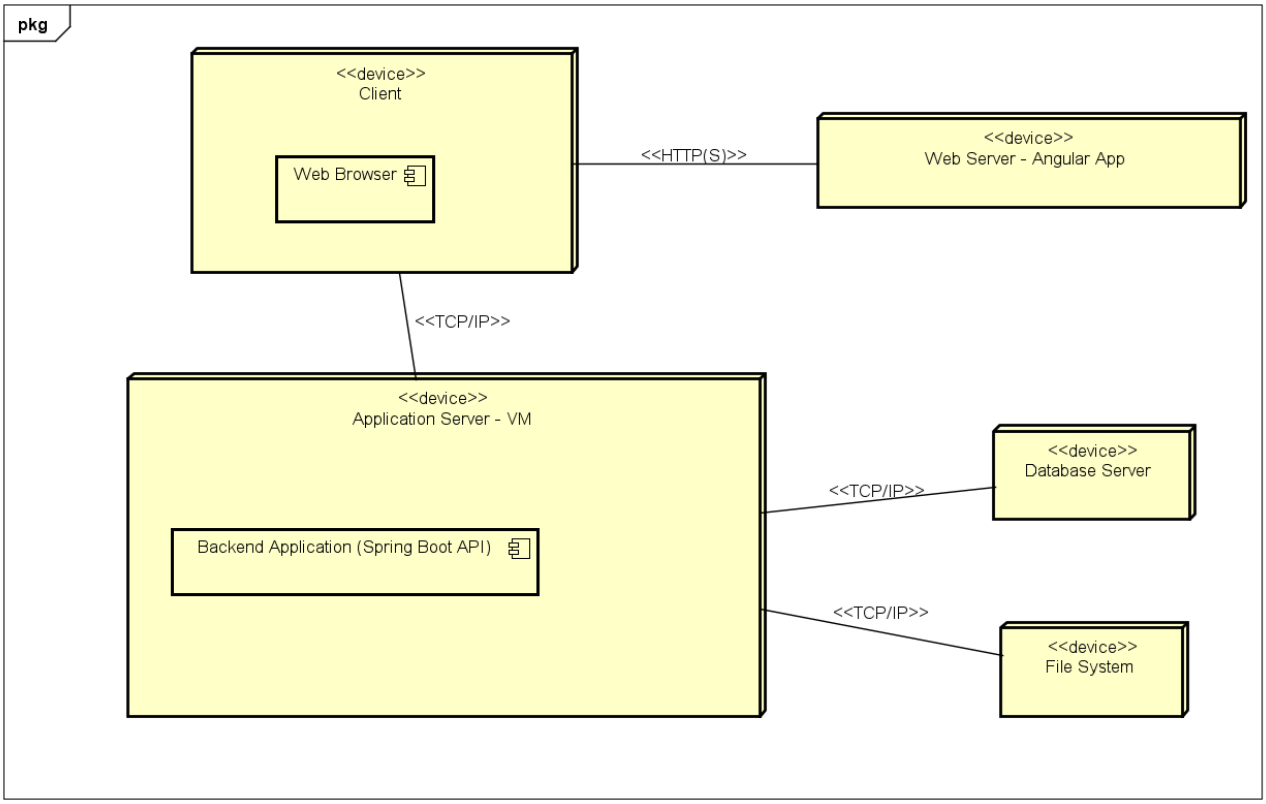

Figura 19. Diagrama de Implantação

4. Projeto de Interface com Usuário

4.1 Esboço das Interfaces Comuns a Todos os Atores

A Figura 20 representa a tela de Dashboard, exibida como ponto de entrada da aplicação. Nela o usuário pode colar trechos de código diretamente na área de entrada para posterior análise. A barra lateral esquerda contém a navegação principal da plataforma, com as opções: Dashboard, GFC, GCE, Relatórios e Tabela de decisão. A presença dessa navegação fixa garante acesso rápido às funcionalidades principais, independentemente do perfil do usuário.

Na parte superior direita da tela encontra-se o campo de pesquisa, permitindo ao usuário localizar rapidamente artefatos ou projetos já analisados. Ao lado do campo de entrada de código, estão presentes os botões “Gerar GFC” e “Novo GCE”, que dão acesso direto às operações mais frequentes: geração de Grafo de Fluxo de Controle e construção de Grafo de Causa-Efeito.

A seção central do Dashboard exibe os resultados gráficos mais recentes. O Grafo de Fluxo de Controle aparece com destaque visual, ao lado do trecho de código correspondente, ressaltando a associação direta entre o artefato de entrada e a representação gráfica. Logo abaixo, é apresentado o

Grafo de Causa-Efeito mais recentemente modelado, permitindo ao usuário visualizar a relação entre causas, operadores lógicos e efeitos derivados.

Por fim, a parte inferior da tela mostra cartões de artefatos já processados. Cada cartão contém informações como nome do artefato, data de geração e métricas de cobertura (por exemplo, comandos e decisões). Esse componente atende tanto alunos que precisam visualizar rapidamente seus exercícios quanto engenheiros que desejam acompanhar a evolução da cobertura em diferentes versões do código.

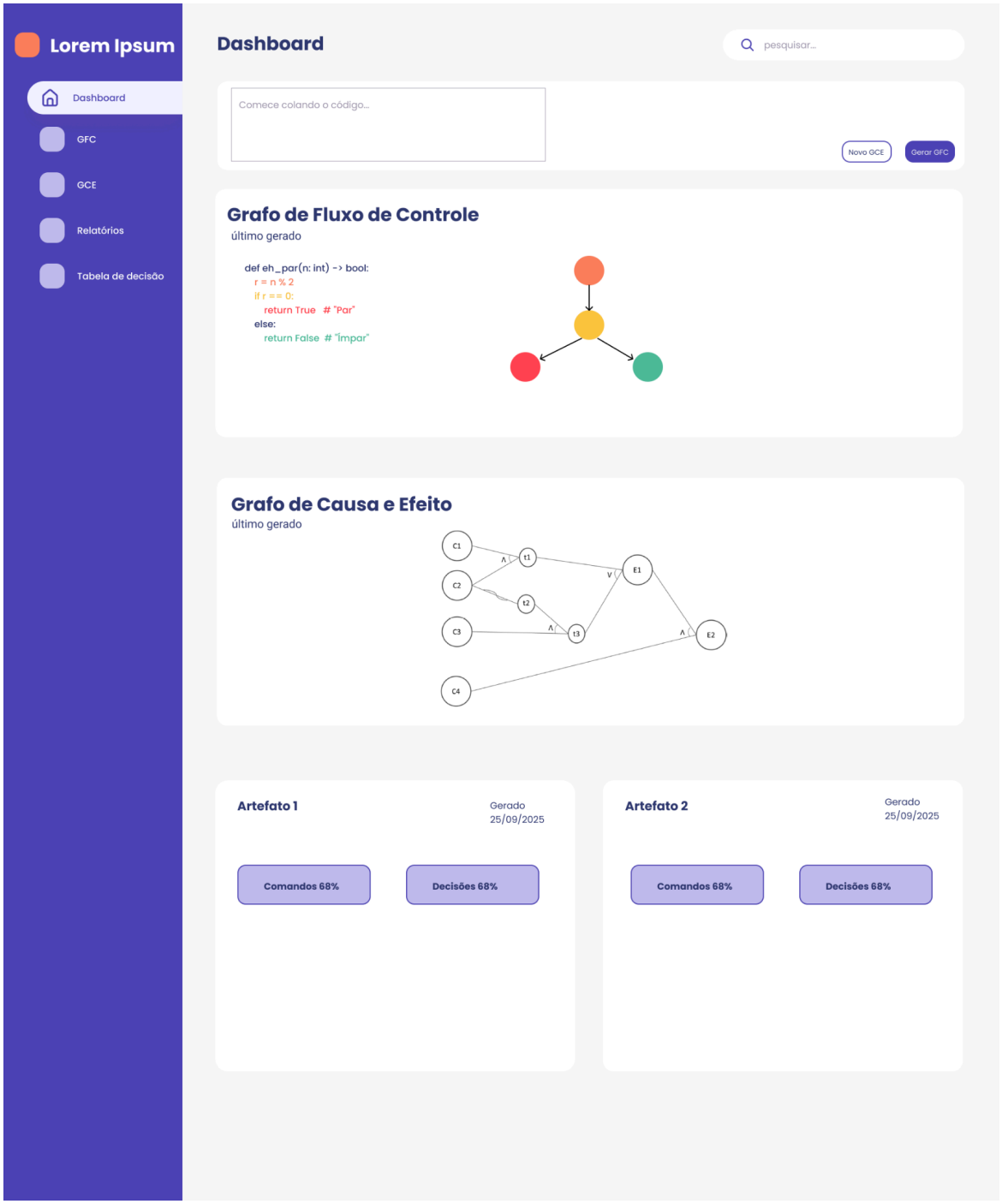

Figura 20. Protótipo da Tela de Dashboard da Plataforma

5. Glossário e Modelos de Dados

O glossário apresentado nesta seção descreve os principais atributos reconhecidos pelos usuários do sistema GraphTest, tanto nas entradas de dados (inserção e configuração de grafos, artefatos e testes) quanto nas saídas de dados (relatórios, métricas e visualizações de cobertura). O objetivo é esclarecer o significado de cada termo presente nas interfaces de interação com o usuário.

Os atributos estão organizados de acordo com as principais funcionalidades do sistema, e estão representados nas Tabelas 4 a 9.
A Figura 20 apresenta o modelo de dados (DER) do sistema, evidenciando as entidades, relacionamentos e atributos que dão suporte às informações descritas neste glossário.

Autenticação na Plataforma		
Atributo	**Formato**	**Descrição**
Nome de Usuário	Texto (até 120 caracteres)	Nome cadastrado do usuário na plataforma.
E-mail	Texto (até 160 caracteres)	Endereço de e-mail utilizado para autenticação no sistema.
Senha	Texto (até 50 caracteres)	Senha inserida pelo usuário para acesso à conta (armazenada de forma criptografada).

Tabela 4. Glossário de Autenticação na Plataforma

Gerenciamento de Projetos		
Atributo	**Formato**	**Descrição**
Nome do Projeto	Texto (até 140 caracteres)	Identificação atribuída ao projeto criado pelo usuário.
Descrição do Projeto	Texto longo	Texto explicativo sobre o propósito e o conteúdo do projeto.
Data de Criação	Data e hora	Indica quando o projeto foi criado no sistema.
Última Atualização	Data e hora	Data da última modificação ou sincronização do projeto.

Tabela 5. Glossário de Gerenciamento de Projetos

Análise de Código e Geração de Grafo de Fluxo de Controle (GFC)		
Atributo	**Formato**	**Descrição**
Código-Fonte	Texto longo	Código Java inserido ou colado pelo usuário para análise.
Grafo de Fluxo de Controle (GFC)	Objeto visual	Representação gráfica gerada automaticamente a partir do código-fonte.
Rótulo do Nó (Label)	Texto (até 255 caracteres)	Identificação textual de um nó no grafo (ex.: "If", "While", "Return").
Tipo de Nó	Lista pré-definida	Classificação do nó (ENTRY, EXIT, ACTION, DECISION).
Condição da Aresta (Edge Condition)	Texto curto	Expressão condicional associada à transição entre nós (ex.: "x > 0").
Data de Geração	Data e hora	Momento em que o grafo foi criado e renderizado pelo sistema.

Tabela 6. Glossário de Geração do GFC

Cálculo e Visualização de Cobertura Estrutural		
Atributo	**Formato**	**Descrição**
Critério de Cobertura	Lista pré-definida	Tipo de métrica analisada (COMANDO, DESVIO, CONDIÇÃO, CAMINHO).
Elementos Cobertos	Número inteiro	Quantidade de elementos (nós, arestas) cobertos pelos testes.
Elementos Totais	Número inteiro	Quantidade total de elementos analisados para o critério selecionado.
Percentual de Cobertura	Decimal (xx.xx)	Porcentagem calculada de cobertura do código.
Relatório de Cobertura	Painel informativo	Visualização interativa que destaca o grau de cobertura no grafo (heatmap).

Tabela 7. Glossário de Cálculo e Visualização de Cobertura

Modelagem de Grafo de Causa e Efeito (GCE)		
Atributo	**Formato**	**Descrição**
Nome do Grafo GCE	Texto (até 160 caracteres)	Nome atribuído ao grafo de causa e efeito modelado pelo usuário.
Nome da Causa	Texto curto	Nome de uma condição lógica (causa) definida no grafo.
Nome do Efeito	Texto curto	Nome de um resultado esperado (efeito) associado às causas.
Tipo de Operador	Lista pré-definida	Operador lógico utilizado na modelagem (AND, OR, NOT).
Tipo de Restrição	Lista pré-definida	Tipo de restrição aplicada às relações (I, E, R, D, L, M).
Status do Grafo	Lista pré-definida	Situação atual do grafo (EM EDIÇÃO, VALIDADO, INCONSISTENTE).

Tabela 8. Glossário do Grafo de Causa e Efeito

Geração da Tabela de Decisão		
Atributo	**Formato**	**Descrição**
Tabela de Decisão	Objeto visual	Tabela derivada automaticamente do GCE, contendo condições e ações.
Índice da Regra (Rule Index)	Número inteiro	Posição ordinal de uma regra na tabela.
Valor da Condição	Lista pré-definida	Valor lógico associado à condição (TRUE, FALSE, -).
Valor da Ação	Lista pré-definida	Valor lógico associado à ação (TRUE, FALSE, -).
Número de Regras	Número inteiro	Quantidade total de regras presentes na tabela de decisão.

Status da Tabela	Lista pré-definida	Estado atual da tabela (GERADA, EXPORTADA, VÁLIDA).

Tabela 9. Glossário da Geração da Tabela de Decisão

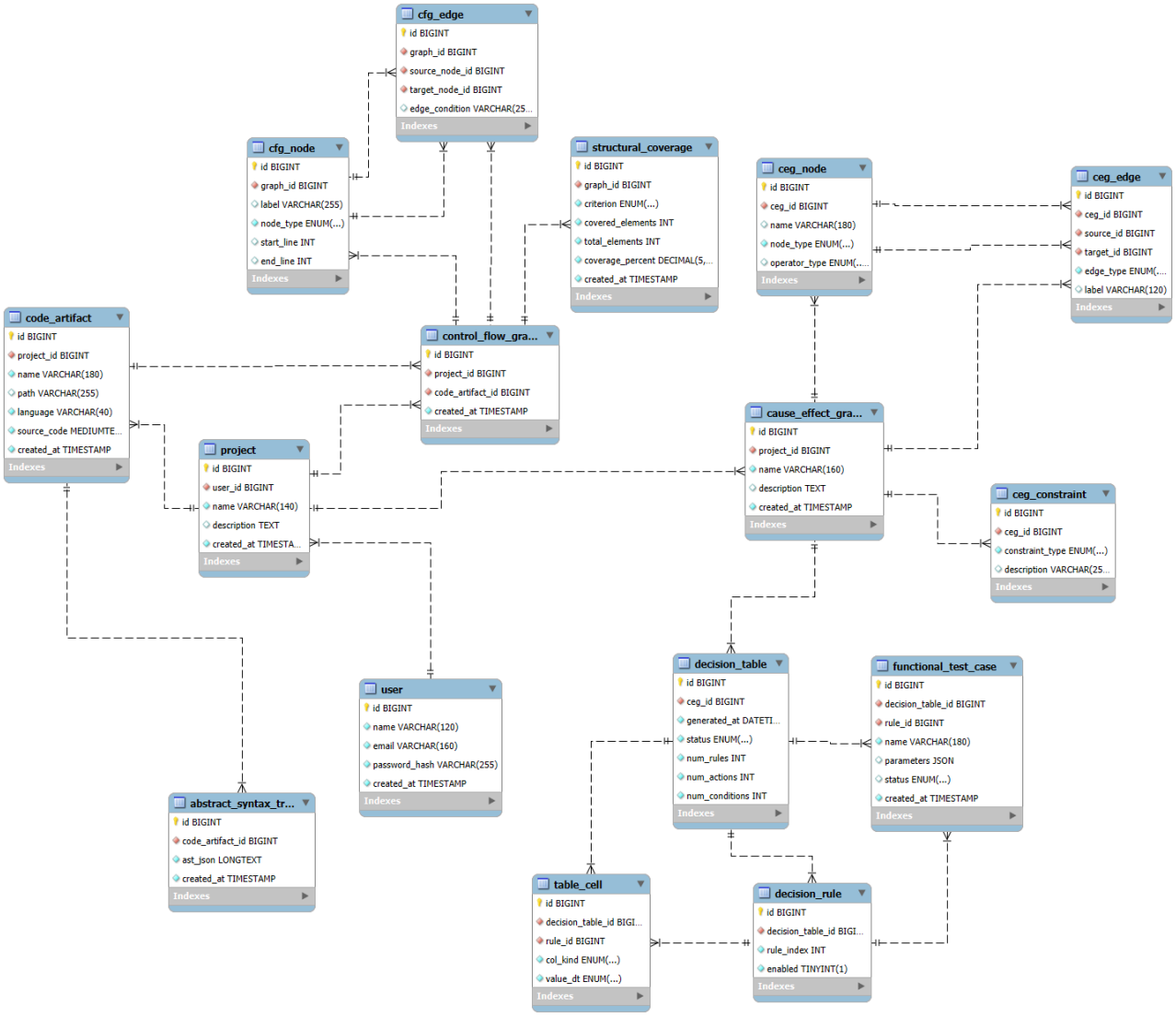

Figura 20. Diagrama de Entidade e Relacionamento

6. Casos de Teste

Uma descrição de casos de teste para validação do sistema.

7. Cronograma e Processo de Implementação

Uma descrição do cronograma para implementação do sistema e do processo que será seguido durante a implementação.

Referências

LARMAN, C. Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development. Upper Saddle River, N.J.: Prentice Hall Ptr, 2004.