# Documentação de Projeto

## para o sistema

# StreamSentry

**Versão 1.0**

Projeto de sistema elaborado pelo aluno Rafael Parreira Chequer 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 a ser definido, orientação acadêmica do professor a ser definido e orientação de TCC II do professor (a ser definido no próximo semestre).

**17 de setembro de 2025**

# Tabela de Conteúdo



# Histórico de Revisões

| Nome | Data | Razões para Mudança | Versão |
|---|---|---|---|
| Rafael Parreira Chequer | 17/09/2025 | Criação inicial do documento para entrega da A3 | 1.0 |
| | | | |

# 1. Introdução

Este documento apresenta a análise de usuário, contexto, interface e interação do sistema **StreamSentry**, uma ferramenta web open-source (licença MIT) desenvolvida em React para automação de testes end-to-end em aplicações de videoconferência baseadas em WebRTC ou APIs de videoconferência (ex.: Zoom SDK, Jitsi). A referência principal para a descrição do problema, domínio e requisitos é o **Documento de Visão** (anexo a este documento), que detalha o escopo, as funcionalidades, os usuários-alvo (desenvolvedores React, engenheiros de QA, equipes ágeis e pesquisadores acadêmicos) e as restrições do sistema. Este documento atende às exigências da Resolução de TCC I, incluindo a elaboração de modelos de usuário, diagrama de casos de uso, histórias de usuário e projeto de interfaces, conforme as boas práticas de engenharia de software.

O objetivo desta entrega é prover: (1) o diagrama de casos de uso e histórias de usuário, (2) modelos de usuários na forma de personas e (3) wireframes de baixa fidelidade para as interfaces do sistema, todos devidamente contextualizados. Os artefatos estão armazenados na pasta **Artefatos** do repositório GitHub Classroom, conforme as instruções da atividade.

# 2. Modelos de Usuário e Requisitos

## 2.1 Descrição de Atores

Os atores que interagem com o sistema **StreamSentry** são definidos com base nos usuários-alvo descritos no **Documento de Visão**. Abaixo está a descrição de cada ator:

- **Desenvolvedor React**: Profissional que desenvolve aplicações de videoconferência em React, utilizando WebRTC ou APIs como Zoom SDK ou Jitsi. Este ator utiliza o **Stream Sentry** para automatizar testes end-to-end, reduzindo o tempo e o esforço em validações manuais.
- **Engenheiro de QA**: Responsável por garantir a qualidade de aplicações de videoconferência, utilizando o sistema para executar testes automatizados e analisar relatórios detalhados com métricas como latência, falhas e cobertura de testes.
- **Pesquisador Acadêmico**: Estudante ou professor que investiga automação de testes multimídia, utilizando o **Stream Sentry** como uma ferramenta open-source para experimentação e análise.
- **Equipe Ágil**: Grupo de profissionais (desenvolvedores, testadores e gerentes) que utilizam o sistema em fluxos de trabalho ágeis para integrar testes automatizados e relatórios em seus processos.

## 2.2 Modelos de Usuários

Para representar os usuários do sistema, foram desenvolvidas **personas** que refletem as características, necessidades e objetivos dos atores identificados. As personas são baseadas nas necessidades levantadas no **Documento de Visão** e ajudam a guiar o design do sistema.

**Persona 1: Lucas, o Desenvolvedor React**

- **Idade**: 28 anos
- **Formação**: Bacharel em Ciência da Computação
- **Contexto**: Trabalha em uma startup que desenvolve uma aplicação de videoconferência baseada em WebRTC. Passa grande parte do tempo codificando em React e precisa garantir que novas funcionalidades sejam testadas rapidamente.
- **Objetivos**: Automatizar testes end-to-end para verificar a estabilidade de conexões com múltiplos usuários e reduzir bugs antes da entrega.
- **Necessidades**: Interface simples para configurar testes, relatórios exportáveis em JSON/CSV e suporte a WebRTC.
- **Frustrações**: Testes manuais consomem tempo e são propensos a erros; ferramentas pagas como TestRTC são caras para a startup.
- **Cenário de Uso**: Configura um teste com 5 usuários virtuais, executa a simulação e exporta um relatório JSON para análise.

**Persona 2: Ana, a Engenheira de QA**

- **Idade**: 32 anos
- **Formação**: Engenharia de Software
- **Contexto**: Trabalha em uma empresa de tecnologia que utiliza APIs como Jitsi. É responsável por validar a qualidade de áudio/vídeo e a estabilidade das conexões.
- **Objetivos**: Executar testes automatizados que gerem métricas detalhadas (ex.: latência, falhas) para identificar problemas rapidamente.
- **Necessidades**: Relatórios claros e detalhados, interface intuitiva para configurar cenários de teste e suporte a headless browsers como Puppeteer.
- **Frustrações**: Ferramentas genéricas como Selenium não são otimizadas para WebRTC, dificultando testes multimídia.
- **Cenário de Uso**: Configura um teste de estabilidade para 10 usuários, visualiza métricas de latência e exporta o relatório em CSV para o time.

**Persona 3: Dr. Carla, a Pesquisadora Acadêmica**

- **Idade**: 40 anos
- **Formação**: Doutora em Ciência da Computação
- **Contexto**: Professora universitária que estuda automação de testes multimídia. Usa ferramentas open-source para experimentação em projetos acadêmicos.
- **Objetivos**: Utilizar uma ferramenta flexível e modificável para testar hipóteses sobre automação de videoconferências.

- **Necessidades**: Código open-source (MIT), documentação clara e suporte a APIs de videoconferência como Zoom SDK.
- **Frustrações**: Ferramentas proprietárias limitam experimentação; falta de documentação em ferramentas open-source.
- **Cenário de Uso**: Configura um teste com Jitsi, modifica o código-fonte para adicionar uma métrica personalizada e documenta os resultados.

## 2.3 Modelo de Casos de Uso e Histórias de Usuários

O diagrama de casos de uso do sistema **StreamSentry** foi desenvolvido para representar as interações principais dos atores com o sistema, conforme as funcionalidades descritas no **Documento de Visão**. O diagrama está disponível na pasta **Artefatos** do repositório com o nome UseCaseDiagram.puml e é apresentado na Figura 1.

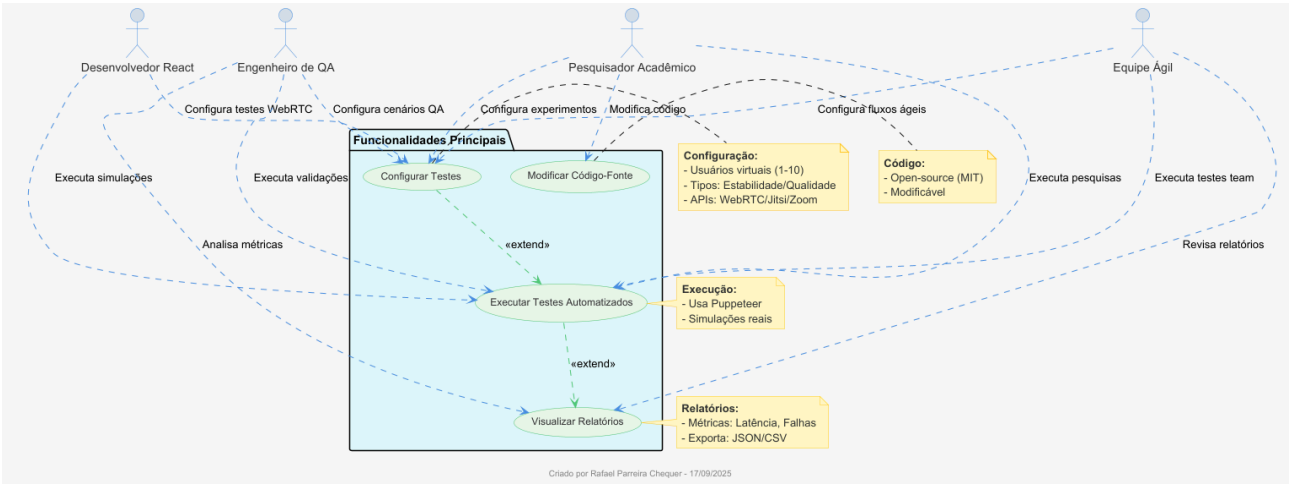


**Figura 1: Diagrama de Casos de Uso do StreamSentry**

O diagrama ilustra os atores (Desenvolvedor React, Engenheiro de QA, Pesquisador Acadêmico, Equipe Ágil) e suas interações com os casos de uso principais: configurar testes, executar testes automatizados, visualizar relatórios e modificar código-fonte. Cada caso de uso reflete uma funcionalidade crítica ou importante identificada no Documento de Visão.

Abaixo está a lista de histórias de usuário, mapeadas com base nas necessidades e funcionalidades do sistema, utilizando identificadores únicos para referência:

- **US01**: Como Desenvolvedor React, eu quero configurar um teste com múltiplos usuários virtuais, para verificar a estabilidade da conexão em uma aplicação WebRTC, garantindo que a funcionalidade seja testada rapidamente.

- **US02**: Como Engenheiro de QA, eu quero executar testes automatizados de qualidade de áudio e vídeo, para identificar falhas de latência ou interrupções antes da entrega do software.
- **US03**: Como Engenheiro de QA, eu quero visualizar relatórios detalhados em formato JSON ou CSV, para compartilhar métricas de teste com a equipe e tomar decisões baseadas em dados.
- **US04**: Como Pesquisador Acadêmico, eu quero acessar o código-fonte do StreamSentry, para modificá-lo e experimentar novos cenários de teste multimídia, atendendo aos meus objetivos de pesquisa.
- **US05**: Como membro de uma Equipe Ágil, eu quero configurar testes via uma interface web intuitiva, para integrar a automação ao fluxo de trabalho do time sem complicações.
- **US06**: Como Desenvolvedor React, eu quero que o sistema suporte APIs de videoconferência como Jitsi, para testar aplicações específicas sem necessidade de ferramentas adicionais.
- **US07**: Como Pesquisador Acadêmico, eu quero acessar documentação técnica completa, para entender a arquitetura do sistema e facilitar sua modificação para experimentos acadêmicos.

## 2.4 Diagrama de Sequência do Sistema e Contrato de Operações

Nesta subseção é apresentado o diagrama de sequência do sistema e os Contratos de Operações.

Formato para cada contrato de operação

| | |
|---|---|
| **Contrato** | |
| **Operação** | |
| **Referências cruzadas** | |
| **Pré-condições** | |
| **Pós-condições** | |

# 3. Projeto de Interface com Usuário

## 3.1 Interfaces Comuns a Todos os Atores

A tela principal do sistema, o Dashboard, é uma interface comum a todos os atores (Desenvolvedor React, Engenheiro de QA, Pesquisador Acadêmico, Equipe Ágil). Ela serve como ponto de entrada para configurar testes, executar simulações, visualizar relatórios e acessar o código-fonte (para pesquisadores). O wireframe do Dashboard foi desenvolvido com base nas necessidades das personas descritas na seção 2.2 e está disponível na pasta Artefatos com o nome DashboardWireframe.png.

Descrição do Dashboard:

Objetivo: Fornecer uma interface intuitiva para gerenciar testes automatizados de videoconferência, com acesso rápido às funcionalidades principais.

Componentes:

Menu Superior: Contém opções para "Configurar Teste", "Executar Teste", "Relatórios" e "Documentação" (para pesquisadores).

Painel de Configuração Rápida: Permite definir parâmetros de teste, como número de usuários virtuais, tipo de API (WebRTC, Jitsi, Zoom SDK) e duração do teste.

Área de Resultados: Exibe um resumo dos testes recentes, com métricas como latência média, taxa de falhas e cobertura de teste.

Botão de Exportação: Permite exportar relatórios em JSON ou CSV.

Barra Lateral: Inclui links para configurações avançadas e acesso ao código-fonte (para pesquisadores).

Contexto de Uso:

Lucas (Desenvolvedor React): Acessa o Dashboard para configurar rapidamente um teste com 5 usuários virtuais e visualizar o resumo de estabilidade.

Ana (Engenheira de QA): Usa o Dashboard para executar testes de áudio/vídeo e exportar relatórios detalhados em CSV.

Dr. Carla (Pesquisadora Acadêmica): Navega até a seção de documentação e código-fonte para modificações experimentais.

Equipe Ágil: Utiliza o Dashboard para integrar testes ao fluxo de trabalho, com configuração simplificada e relatórios acessíveis.

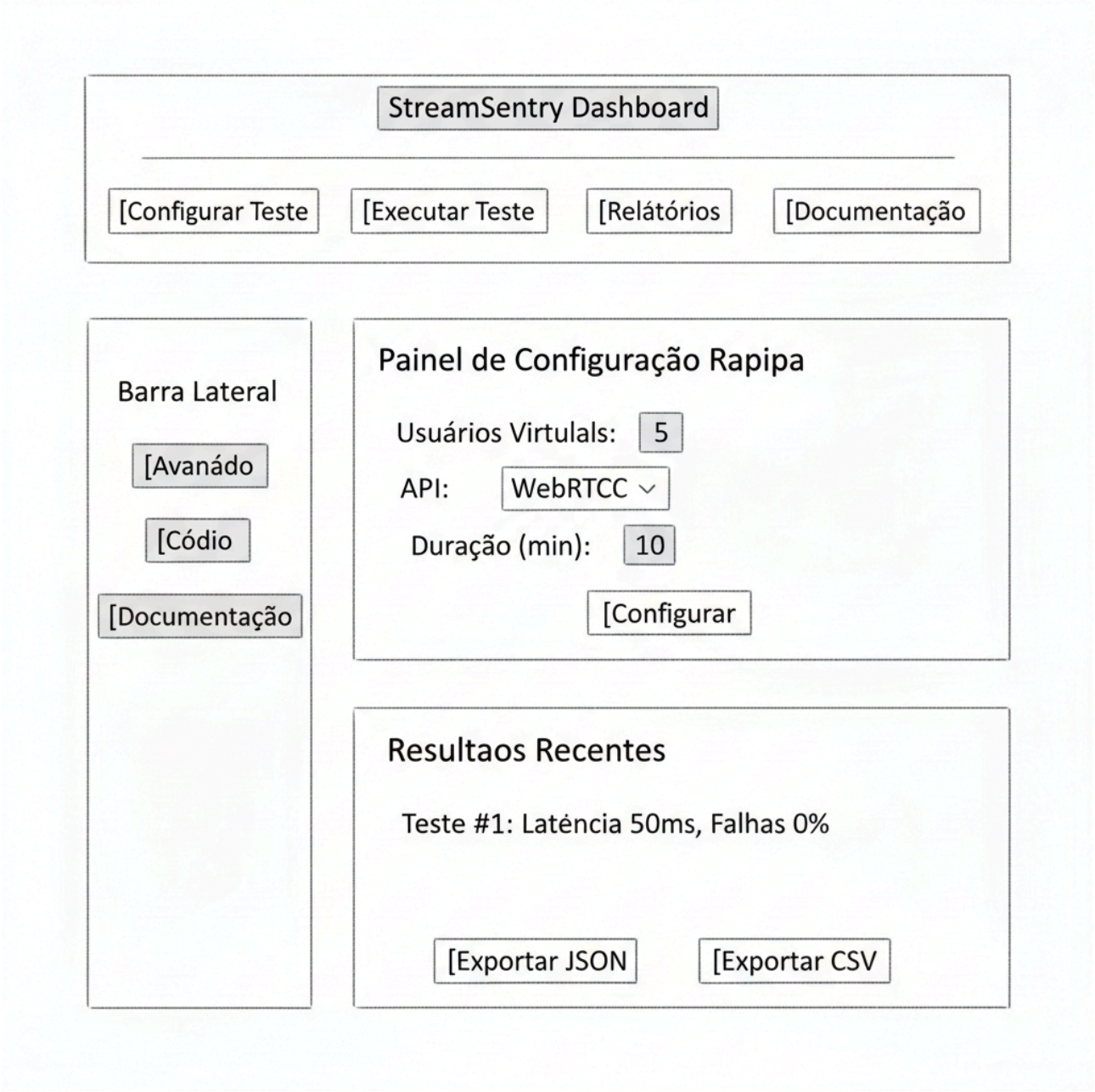

Figura 2: Wireframe do Dashboard do StreamSentry

(O wireframe está disponível na pasta Artefatos como DashboardWireframe.png.)

Justificativa:

O Dashboard foi projetado para ser simples e funcional, atendendo às necessidades de todos os atores. A interface é minimalista, com botões claros e áreas bem definidas, garantindo usabilidade para usuários técnicos (como desenvolvedores e pesquisadores) e equipes ágeis que precisam de agilidade. O suporte a exportação de relatórios atende às demandas de Ana (Engenheira de QA), enquanto o acesso ao código-fonte e à documentação supre as necessidades de Dr. Carla (Pesquisadora Acadêmica).

# 4. Modelos de Projeto

## 4.1 Diagrama de Classes

Diagrama de classes do sistema

## 4.2 Diagramas de Sequência

Diagramas de sequência para realização de casos de uso.

## 4.3 Diagramas de Comunicação

Diagramas de comunicação para realização de casos de uso.

## 4.4 Arquitetura

Pode ser descrita com um diagrama apropriado da UML ou C4 Model

## 4.5 Diagramas de Estados

Diagramas de estados do sistema.

### 4.6 Diagrama de Componentes e Implantação.

Diagramas de componentes do sistema. Diagrama de implantação mostrando onde os componentes estarão alocados para a execução.

# 5. Projeto de Interface com Usuário

### 5.1 Esboço das Interfaces Comuns a Todos os Atores

Wireframe/mockup/storyboard das interfaces que são comuns a todos os atores do sistema.

### 5.2 Esboço das Interfaces Usadas pelo Ator <*A*>

Wireframe/mockup/storyboard das interfaces exclusivas do ator <*A*>

### 5.3 Esboço das Interfaces Usadas pelo Ator <*B*>

Wireframe/mockup/storyboard das interfaces exclusivas do ator <B>

# 6. Glossário e Modelos de Dados

Deve-se apresentar o glossário para o sistema. Também apresente esquemas de banco de dados e as estratégias de mapeamento entre as representações de objetos e não-objetos.

# 7. Casos de Teste

Uma descrição de casos de teste para validação do sistema.

# 8. Cronograma e Processo de Implementação

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