Documentação de Projeto

para o sistema

Swagger to SDK

Versão 1.0

Projeto de sistema elaborado pelo aluno Arthur do Nascimento Sita Gomes 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 Cleiton Silva Tavares, orientação acadêmica do professor Leonardo Viela Cardoso e orientação de TCC II do professor (a ser definido no próximo semestre).

24/09/2025

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Arthur do Nascimento Sita Gomes	24/09/2025	• Criação do documento	1.0

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema Swagger to SDK. O Swagger to SDK é uma extensão para o *Visual Studio Code* (VScode), com o intuito de ajudar os desenvolvedores a automatizar a geração de clientes de acesso a serviços baseados no padrão de Transferência de Estado Representacional (*REST*, do inglês Representational State Transfer) para aplicações de interface com o usuário. 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.

2. Modelos de Usuário e Requisitos

Esta seção apresenta os modelos de usuário e os requisitos do sistema. A apresentação é composta pelas seguintes divisões: descrição dos atores (Seção 2.1); descrição dos modelos de usuários (Seção 2.2); descrição dos casos de uso e histórias do usuário (Seção 2.3); e descrição diagrama de sequência do sistema e contrato de operações (Seção 2.4).

2.1 Descrição de Atores

O ator identificado nesta etapa é uma pessoa desenvolvedora que trabalha com o desenvolvimento de aplicações *front-end* ou *full-stack*, que integra aplicações *client* a serviços *REST* via *Swagger/OpenAPI* no *VSCode*. Suas dores concentram-se no excesso de *boilerplates*, inconsistências entre contrato e implementação, erros de tipagem detectados tardiamente, documentação desatualizada e baixa padronização entre projetos.

2.2 Modelos de Usuários

Nesta seção são apresentados os modelos de usuários. O modelo foi construído usando a estratégia de personas, no qual são especificados: a biografia, os objetivos, as tarefas, e as dores. A Figura 1 é apresentada a primeira persona que representa os desenvolvedores *front-end*. A Figura 2 temos a persona que representa os desenvolvedores *full-stack*.

Figura 1. Persona que representa um desenvolvedor *front-end.*

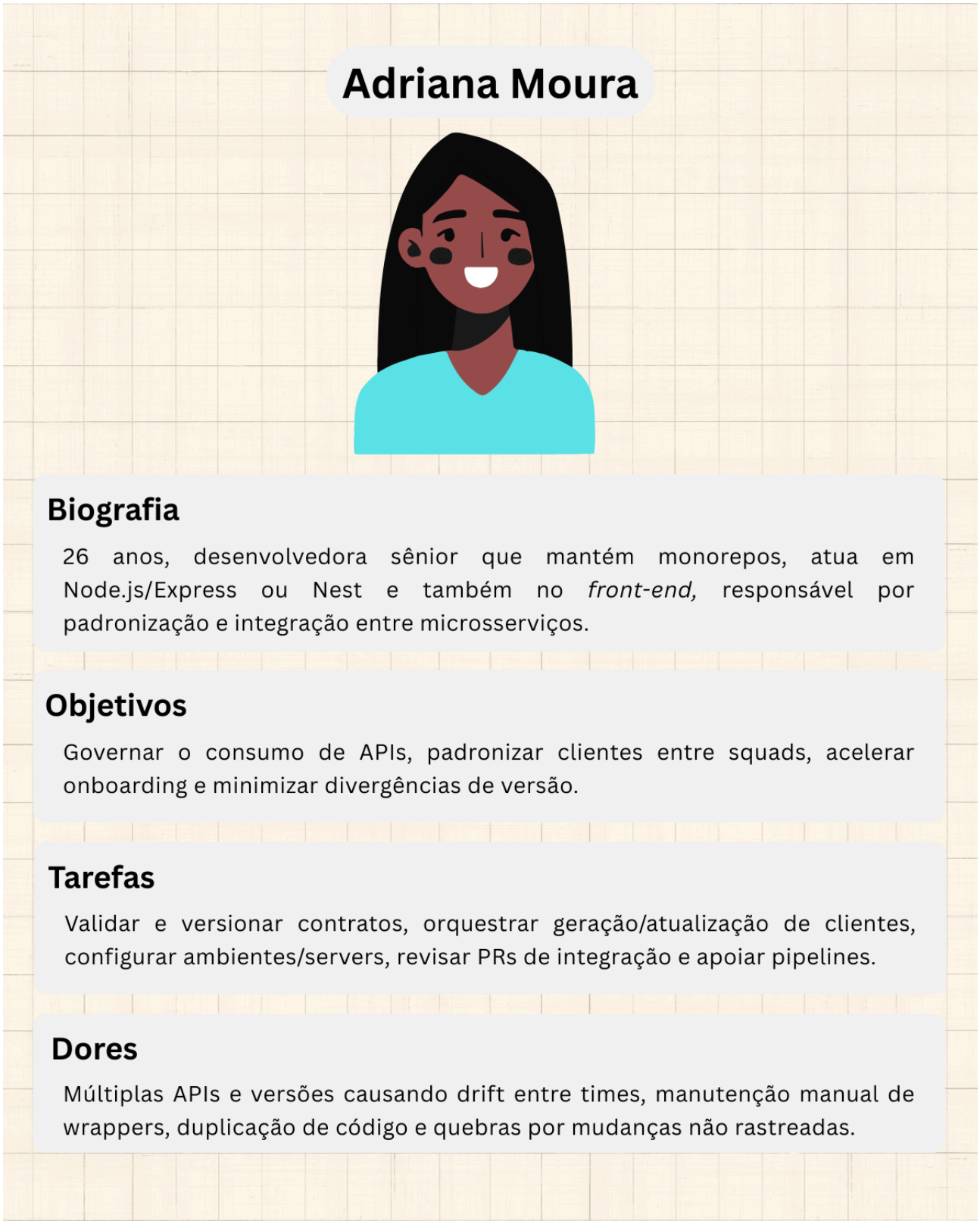

Figura 2. Persona que representa um desenvolvedor *full-stack*.

2.3 Modelo de Casos de Uso e Histórias de Usuários

Esta seção descreve os casos de uso e as histórias de usuário que contemplam as possíveis ações dos usuários da extensão, apresentados respectivamente nas seções 2.3.1 e 2.3.2. Para descrever os casos de uso, foi criado o Diagrama de Caso de Uso, o qual é responsável por descrever as ações que um usuário pode realizar na extensão. As histórias de usuário têm o objetivo de apresentar as necessidades do usuário em relação à extensão de uma maneira clara e concisa para facilitar o entendimento das funcionalidades.

2.3.1 Diagrama de Casos de Uso

Esta seção descreve os casos de uso e as histórias de usuário que contemplam

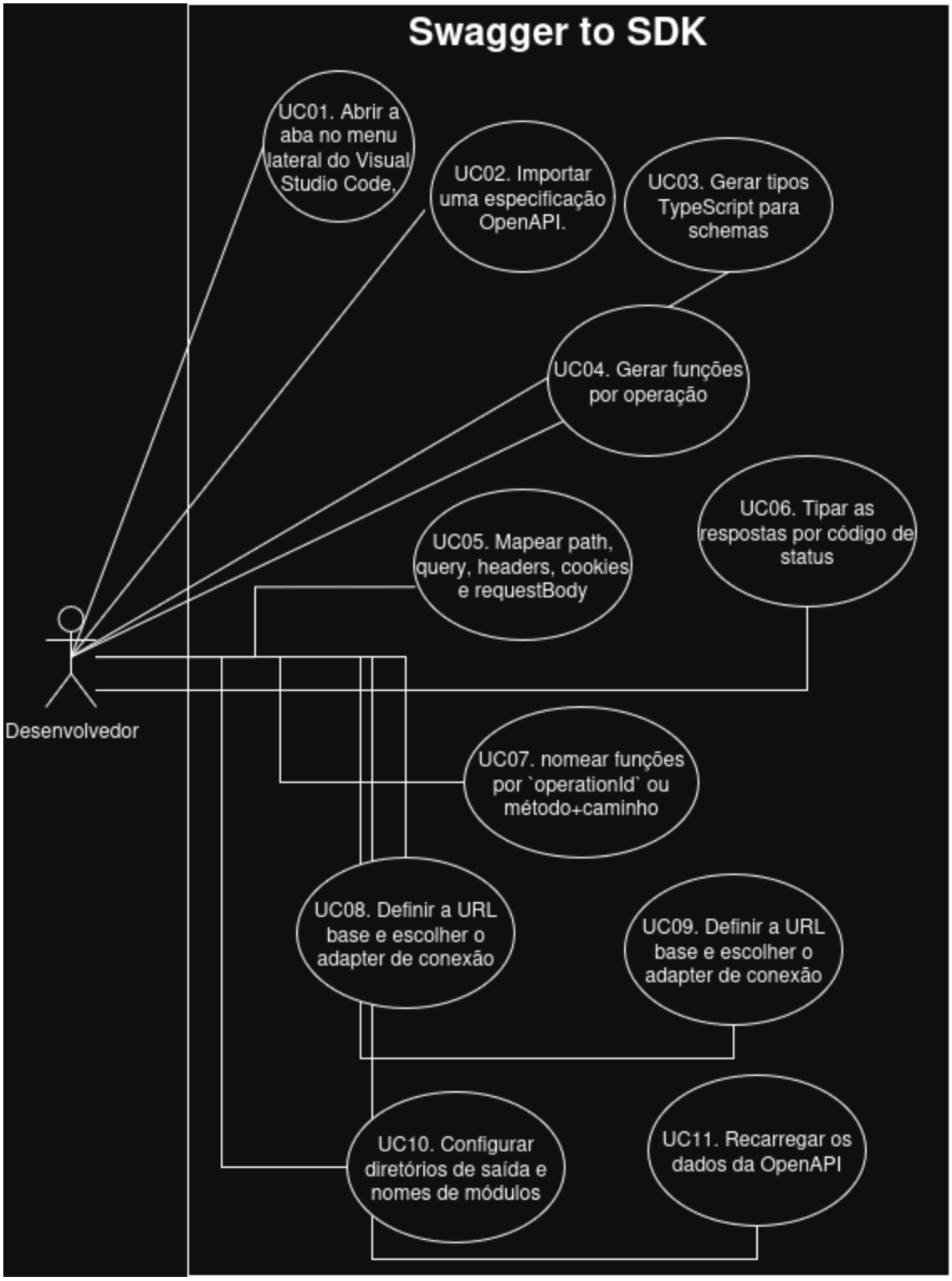

Figura 3. Diagrama de Caso de Uso

2.3.2 Histórias de Usuários

As histórias de usuário (US, do inglês User Stories) são uma forma de representar os requisitos do usuário. Trata-se de uma descrição simples, que informa quem realizará uma determinada ação, qual será essa ação, e a razão que ela é realizada. A seguir, são descritas as histórias de usuários levantadas para a plataforma, identificadas pela sigla US e uma numeração.

US01. Como pessoa desenvolvedora, desejo abrir a aba no menu lateral do *VScode*, para trazer informações pertinentes à extensão;

US02. Como pessoa desenvolvedora, desejo importar uma especificação *OpenAPI* através de um arquivo local ou *URL*, para iniciar a geração do cliente sem escrever integração manual;

US03. Como pessoa desenvolvedora, desejo gerar tipos *TypeScript* para *schemas*, parâmetros e respostas, para ter uma tipagem estática e reduzir erros.

US04. Como pessoa desenvolvedora, desejo gerar funções por operação com assinatura tipada completa, para consumir a *API* com segurança e autocomplete.

US05. Como pessoa desenvolvedora, desejo mapear *path*, *query*, *headers*, *cookies* e *requestBody*, para ter previsibilidade e validação das chamadas.

US06. Como pessoa desenvolvedora, desejo tipar as respostas por código de *status*, para tratar fluxos felizes e de erro corretamente.

US07. Como pessoa desenvolvedora, desejo nomear funções por `*operationId*` ou método+caminho quando ausente, para manter rastreabilidade entre contrato e código.

US08. Como pessoa desenvolvedora, desejo definir a *URL* base e escolher o adaptador de conexão, para configurar ambientes de forma consistente.

US09. Como pessoa desenvolvedora, desejo injetar credenciais *Bearer* ou *API Key* e *interceptors*, para atender requisitos de segurança sem acoplar a autenticação.

US10. Como pessoa desenvolvedora, desejo configurar diretórios de saída e nomes de módulos, para organizar o código gerado conforme o projeto.

US11. Como pessoa desenvolvedora, desejo recarregar os dados da *OpenAPI* por meio de um botão na interface, para poder reiniciar a extensão, caso exista algum dado inconsistente com o estado atual da aplicação.

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. Modelos de Projeto

3.1 Diagrama de Classes

Diagrama de classes do sistema

3.2 Diagramas de Sequência

Diagramas de sequência para realização de casos de uso.

3.3 Diagramas de Comunicação

Diagramas de comunicação para realização de casos de uso.

3.4 Arquitetura

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

3.5 Diagramas de Estados

Diagramas de estados do sistema.

3.6 Diagrama de Componentes e Implantação.

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

4. Projeto de Interface com Usuário

4.1 Esboço das Interfaces Comuns a Todos os Atores

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

4.2 Esboço das Interfaces Usadas pelo Ator <*A*>

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

4.3 Esboço das Interfaces Usadas pelo Ator <*B*>

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

5. 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.

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.