INSTITUTO FEDERAL GOIANO
CAMPUS MORRINHOS
CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

VINICIUS VIEIRA DE SOUZA MELO

APLICAÇÃO DE GERENCIAMENTO E DISTRIBUIÇÃO DE ARTEFATOS DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS

MORRINHOS – GO
2024

VINICIUS VIEIRA DE SOUZA MELO

APLICAÇÃO DE GERENCIAMENTO E DISTRIBUIÇÃO DE ARTEFATOS DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS

Monografia apresentada ao Curso Superior de Ciências da Computação do Instituto Federal Goiano – Campus Morrinhos, como requisito parcial para obtenção de título de Bacharel em Ciências da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Alexandre Carvalho Silva.

MORRINHOS - GO
2024

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas – SIBI/IF Goiano Campus Morrinhos

M528a Melo, Vinicius Vieira de Souza.
Aplicação de gerenciamento e distribuição de artefatos de ambientes virtuais tridimencionais. – Morrinhos, GO: IF Goiano, 2024.
62 f. : il. color.

Orientador: Dr. Alexandre Carvalho Silva.

Trabalho de conclusão de curso (graduação) – Instituto Federal Goiano Campus Morrinhos, Tecnologia em Sistemas para Internet, 2024.

1. Integração de softwares. 2. Imagem tridimensional. 3. Modelagem. I. Silva, Alexandre Carvalho. II. Instituto Federal Goiano. III. Titulo.

CDU 004.92

Fonte: Elaborado pela Bibliotecária-documentalista Morgana Guimarães, CRB1/283

INSTITUTO FEDERAL
Goiano

Repositório Institucional do IF Goiano - RIIF Goiano
Sistema Integrado de Bibliotecas

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO

PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610, de 19 de fevereiro de 1998, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano a disponibilizar gratuitamente o documento em formato digital no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

IDENTIFICAÇÃO DA PRODUÇÃO TÉCNICO-CIENTÍFICA

- ☐ Tese (doutorado)
- ☐ Dissertação (mestrado)
- ☐ Monografia (especialização)
- ☑ TCC (graduação)
- ☐ Artigo científico
- ☐ Capítulo de livro
- ☐ Livro
- ☐ Trabalho apresentado em evento
- ☐ Produto técnico e educacional - Tipo:

Nome completo do autor:
Vinicius Vieira de Souza Melo

Matrícula:
2018104201940054

Título do trabalho:
APLICAÇÃO DE GERENCIAMENTO E DISTRIBUIÇÃO DE ARTEFATOS DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS

RESTRIÇÕES DE ACESSO AO DOCUMENTO

Documento confidencial: ☑ Não ☐ Sim, justifique:

Informe a data que poderá ser disponibilizado no RIIF Goiano: 10 /10 /2024

O documento está sujeito a registro de patente? ☐ Sim ☑ Não

O documento pode vir a ser publicado como livro? ☐ Sim ☑ Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara:

• Que o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

• Que obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autoria, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

• Que cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Morrinhos — Local
10 /10 /2024 — Data

Documento assinado digitalmente
gov.br
VINICIUS VIEIRA DE SOUZA MELO
Data: 15/10/2024 14:17:15-0300
Verifique em https://validar.iti.gov.br

Assinatura do autor e/ou detentor dos direitos autorais

Ciente e de acordo:

Documento assinado digitalmente
gov.br
ALEXANDRE CARVALHO SILVA
Data: 14/10/2024 19:13:54-0300
Verifique em https://validar.iti.gov.br

Assinatura do(a) orientador(a)

VINICIUS VIEIRA DE SOUZA MELO

APLICAÇÃO DE GERENCIAMENTO E DISTRIBUIÇÃO DE ARTEFATOS DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS

Data de defesa: 07/10/2024

Resultado: Aprovado

BANCA EXAMINADORA

_____

Prof. Dr. Alexandre Carvalho Silva (Presidente da Banca)
Instituto Federal Goiano Campus Morrinhos

_____

Prof. Esp. José Pereira Alves
Instituto Federal Goiano Campus Morrinhos

_____

Prof. Me. Diogo Aparecido Cavalcante
Faculdade Uniessa

MORRINHOS – GO
2024

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

Ata nº 68/2024 - CCEG-MO/CEG-MO/DE-MO/CMPMHOS/IFGOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

ATA DE DEFESA DA BANCA DE EXAME DE TRABALHO DE CURSO POR VIDEOCONFERÊNCIA

Aos **07** dias do mês de outubro de **2024**, às **19** horas foi realizada a Banca de Exame, em formato remoto para a apresentação pública e defesa do trabalho de curso do discente **Vinicius Vieira de Souza Melo** intitulado **APLICAÇÃO DE GERENCIAMENTO E DISTRIBUIÇÃO DE ARTEFATOS DE AMBIENTES VIRTUAIS TRIDIMENSIONAIS,** como requisito necessário para a conclusão do curso.

A Banca de Exame foi constituída pelos membros: **Prof. Dr. Alexandre Carvalho Silva (orientador)**, **Prof. Esp. José Pereira Alves**, **Prof. Me. Diogo Aparecido Cavalcante de Lima.** Após a análise, emitiram o seguinte resultado:

1 - (X) Aprovado

2 - ( ) Aprovado com ressalva

(A Banca Examinadora deve definir as exigências a serem cumpridas pelo aluno na revisão, ficando o orientador responsável pela verificação do cumprimento das mesmas.)

Observações:_____

3 - ( ) Reprovado com o seguinte parecer: _____
_____

Morrinhos -GO, 07 de outubro de 2024.

Por ser verdade firmamos a presente:

(Assinado Eletronicamente)

Prof. Dr. Alexandre Carvalho Silva

Orientador(a)

(Assinado Eletronicamente)

Prof. Esp. José Pereira Alves

Membro

Documento assinado digitalmente
gov.br
DIOGO APARECIDO CAVALCANTE DE LIMA
Data: 09/10/2024 21:38:28-0300
Verifique em https://validar.iti.gov.br

(Assinado Eletronicamente)

Prof. Me. Diogo Aparecido Cavalcante de Lima

Membro

Documento assinado eletronicamente por:

- Jose Pereira Alves, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 07/10/2024 21:04:27.
- Alexandre Carvalho Silva, CHEFE - FG2 - UPIC-MO, em 07/10/2024 20:31:19.

Este documento foi emitido pelo SUAP em 07/10/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 641229
Código de Autenticação: d0dbac24e0

RESUMO

Com a alta demanda da modelagem tridimensional na contemporaneidade, observa-se um notável incremento no interesse por esse campo, suscitando a reflexão acerca do gerenciamento dos artefatos tridimensionais empregados em distintos softwares de modelagem. Estes artefatos englobam modelos 3D, texturas, materiais, posicionamento no espaço tridimensional, escala, orientação, entre outros elementos relevantes. Diante da imperatividade de uma eficaz gestão desses artefatos e da necessidade de integração entre os diversos softwares utilizados para compor ambientes tridimensionais interativos, foi concebida uma solução computacional que visa ordenar, armazenar e integrar tais elementos. Adicionalmente, foram conduzidos testes funcionais e elaborada a documentação referente à tecnologia desenvolvida, a fim de assegurar sua eficácia e utilidade no contexto da modelagem tridimensional.

Palavras-chave: Modelagem tridimensional, gerenciamento de artefatos 3D, integração de softwares, ambientes tridimensionais, artefatos tridimensionais.

ABSTRACT

With the proliferation of three-dimensional modeling in contemporary times, there has been a notable increase in interest in this field, prompting reflection on the management of three-dimensional artifacts used in different modeling software. These artifacts include 3D models, textures, materials, positioning in three-dimensional space, scale, orientation, among other relevant elements. Faced with the imperative of effective management of these artifacts and the need for integration between the various software used, a computational solution has been conceived to organize, store, and integrate such elements. Additionally, functional tests were conducted and documentation related to the developed technology was prepared in order to ensure its effectiveness and usefulness in the context of three-dimensional modeling.

Keywords: Three-dimensional modeling, 3D artifact management, software integration, three-dimensional environments, three-dimensional artifacts.

LISTA DE FIGURAS

Figura 1 – Diagrama de arquitetura 17
Figura 2 – Casos de uso 18
Figura 3 – Diagrama de classe 21
Figura 4 – Fluxograma de criação de usuário 27
Figura 5 – Fluxograma de autenticação 29
Figura 6 – Fluxograma de consulta de modelo por ID 31
Figura 7 – Fluxograma de exclusão de texturas 33
Figura 8 – Captura de Tela Postman Create User 37
Figura 9 – Captura de Tela Azure Data Studio 37
Figura 10 – Captura de Tela Postman User Login 38
Figura 11 – Captura de Tela Postman Material BuscaPorId 39
Figura 12 – Captura de Tela Postman Listagem Texturas 40
Figura 13 – Captura de Tela Postman Deleta Textura 41
Figura 14 – Captura de Tela Swagger 42
Figura 15 – Captura de Tela Swagger Métodos 43
Figura 16 – Captura de Tela Swagger Método Expandido 44

LISTA DE TABELAS

Tabela 1 – Descrição UC01 19
Tabela 2 – Descrição UC02 19
Tabela 3 – Descrição UC03 20
Tabela 4 – Descrição UC04 20
Tabela 5 – Teste UC01 47
Tabela 6 – Resultado do teste UC01 47
Tabela 7 – Teste UC02 49
Tabela 8 – Resultado do teste UC02 50
Tabela 9 – Teste UC03 51
Tabela 10 – Resultado do teste UC03 52
Tabela 11 – Teste UC04 53
Tabela 12 – Resultado do teste UC04 54

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
C#	C Sharp
DDD	Domain-Driven Design
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
REST	Representational State Transfer
SQL	Structured Query Language
UC	Caso de uso

SUMÁRIO

1	**INTRODUÇÃO**	12
1.1	**Problemática**	12
1.2	**Área Com Potencial de Uso na Problemática**	13
1.3	**Objetivo Geral**	13
1.4	**Objetivos Específicos**	14
1.5	**Justificativa**	14
1.5.1	Desafios da Gestão Manual de Artefatos 3D	14
1.5.2	Contribuição do Software	15
1.6	**Organização do Trabalho**	15
2	**ARQUITETURA DO SISTEMA**	16
2.1	**Casos de Uso**	18
2.1.1	Descrição dos atores	18
2.1.2	Descrição dos casos de uso	19
2.2	**Diagramas De Classe**	21
2.3	**Tecnologias Utilizadas**	23
2.3.1	C# (C Sharp)	23
2.3.2	NET	23
2.3.3	SQL Server	24
2.3.4	Docker	24
2.3.5	Fluent Validation	25
2.3.6	Swagger	25
2.4	**Fluxogramas**	25
2.4.1	Fluxograma de criação de usuário	26
2.4.2	Fluxograma de autenticação	28
2.4.3	Fluxograma de consulta de modelo por ID	30
2.4.4	Fluxograma de exclusão de texturas	32
2.5	**Observações Finais**	34
3	**FUNCIONAMENTO DO SISTEMA**	36
3.1	**Cadastro de Usuário**	36
3.2	**Login do Usuário**	38
3.3	**Consulta de Materiais Por Id**	39

3.4 **Listagem de Texturas** 40
3.5 **Deleção de Textura** 41
3.6 **Documentação** 42
3.7 **Considerações Finais** 45
4 **TESTES DE FUNCIONAMENTO** 46
4.1 **Considerações Iniciais** 46
4.2 **Teste De Caixa Preta** 46
4.3 **Validação Dos Casos De Teste** 46
4.3.1 Autenticação 47
4.3.2 Upload de artefatos 49
4.3.3 Gerenciamento de artefatos 51
4.3.4 Integração de software 53
4.4 **Resultados** 55
4.4.1 Resultado do Teste UC01: Autenticação 56
4.4.2 Resultado do Teste UC02: Upload de Artefatos 56
4.4.3 Resultado do Teste UC03: Gerenciamento de Artefatos 56
4.4.4 Resultado do Teste UC04: Integração de Software 57
5 **CONCLUSÃO** 58
5.1 **Trabalhos Futuros** 58
REFERÊNCIAS 60

1 INTRODUÇÃO

Com a disseminação da modelagem 3D nos últimos anos, esta técnica transcendeu seu domínio exclusivo em grandes corporações e projetos de elevado aporte financeiro, desvelando novas fronteiras até então inexploradas. No âmbito da modelagem tridimensional, diversas ferramentas colaboram na concepção dos artefatos imprescindíveis para a composição de um ambiente 3D interativo.

Segundo a pesquisa de Gonçalves e Silva (2020), a gestão eficiente dos artefatos tridimensionais é crucial para a integridade e eficácia dos projetos de modelagem. Além disso, conforme apontado por Santos (2018), a utilização de ferramentas adequadas para o gerenciamento desses artefatos é essencial para evitar retrabalhos e perdas de dados.

Entretanto, a administração desses artefatos frequentemente ocorre de maneira manual, em ambientes inadequados, o que acarreta possíveis retrabalhos, extravios de arquivos, demora na localização dos mesmos e outros transtornos. Tais falhas podem impactar diretamente a eficiência, a qualidade do gerenciamento e a salvaguarda dos artefatos empregados em projetos de grande envergadura ou por usuários que exploram essa tecnologia em suas atividades diárias.

Diante desse panorama, torna-se cada vez mais premente a utilização de ferramentas que simplifiquem e garantam uma integração e organização eficazes, visando assegurar uma gestão otimizada e segura dos artefatos tridimensionais em diversos contextos de aplicação.

1.1 Problemática

A gestão eficiente de conteúdos para a montagem de cenários 3D de forma integrada e dinâmica é um desafio significativo na área de modelagem tridimensional. A criação de cenários 3D envolve uma ampla gama de componentes, incluindo modelos 3D, texturas, materiais, posicionamento no espaço tridimensional, dimensionamento e orientação. A complexidade desses elementos torna a gestão e manutenção desses cenários uma tarefa manual e trabalhosa.

Nesse contexto, torna-se imperativo desenvolver estratégias computacionais que otimizem o processo de gerenciamento desses conteúdos. A implementação de soluções tecnológicas pode facilitar a organização, atualização e manutenção

eficiente dos elementos que compõem os cenários 3D, contribuindo para uma abordagem mais ágil e eficaz na criação e gestão desses ambientes virtuais.

Assim, diante desse cenário, surge a seguinte problemática: Como gerir conteúdos para a montagem de cenários 3D de forma integrada e dinâmica?

1.2 Área com Potencial de uso na Problemática

O domínio dos sistemas computacionais distribuídos baseados na internet é uma realidade incontestável e uma necessidade premente em todos os setores globais, com seu crescimento em franca ascensão. Nesse cenário, ferramentas integradas que aprimoram a experiência do usuário, garantindo a segurança de seus dados, tornam-se absolutamente imprescindíveis.

A integração permite que diferentes sistemas compartilhem informações de forma fluida, eliminando silos de dados e melhorando a eficiência operacional. Isso resulta em maior visibilidade e em uma tomada de decisões mais informada, já que os dados estão disponíveis em tempo real. O acoplamento promove a interoperabilidade necessária em um ambiente digital dinâmico, onde as necessidades dos usuários estão em constante mudança. Ferramentas de integração facilitam a migração contínua de dados entre sistemas legados e modernos, mantendo a integridade e a consistência das informações. Além disso, técnicas de sincronização em tempo real garantem que todas as partes interessadas tenham acesso aos dados mais atualizados, algo crucial para a maioria dos setores.

Por fim, a governança de dados se beneficia da integração, permitindo que as organizações gerenciem informações conforme as regulamentações vigentes, promovendo segurança e conformidade legal. Assim, fica claro que a integração e o acoplamento de aplicações são essenciais para organizações que desejam se destacar em um mercado competitivo e orientado por dados.

1.3 Objetivo Geral

Com base na problemática exposta, o objetivo geral deste estudo consiste em desenvolver um sistema computacional apto a gerenciar e disseminar informações contextualizadas para a montagem de cenários tridimensionais interativos.

1.4 Objetivos Específicos

- ✓ Obter requisitos validáveis para a concepção do sistema.
- ✓ Produzir uma estratégia para a integração do sistema proposto com outras aplicações por meio de APIs.
- ✓ Garantir resultados eficientes em relação ao funcionamento do sistema computacional proposto.

1.5 Justificativa do Trabalho

Com o intuito de aprimorar a eficiência na construção de cenários 3D e promover a rápida adoção dessa tecnologia, torna-se justificada a elaboração de métodos construtivos para automatizar tarefas dispendiosas, que demandam tempo e são propensas a falhas, podemos listar alguns desafios da gestão manual de artefatos abaixo.

1.5.1 Desafios da Gestão Manual de Artefatos 3D

A gestão manual de artefatos 3D apresenta uma série de dificuldades que podem comprometer a qualidade e a eficiência dos projetos. Entre os principais desafios, destacam-se:

1. **Posicionamento Incorreto**: A manipulação manual de artefatos pode resultar em posicionamentos imprecisos, afetando a integridade do cenário e exigindo retrabalho significativo.
2. **Trocas Equivocadas**: A falta de um sistema automatizado pode levar a trocas equivocadas de modelos ou texturas, resultando em inconsistências visuais e atrasos na entrega dos projetos.
3. **Atrasos no Processo**: O gerenciamento manual é frequentemente lento e propenso a erros, o que pode causar atrasos consideráveis na conclusão dos projetos. Isso é especialmente crítico em ambientes competitivos onde prazos são essenciais.
4. **Ajustes e Atualizações**: A atualização manual de artefatos e suas propriedades é uma tarefa árdua que consome tempo e aumenta o risco de

falhas. Um software eficiente pode facilitar esse processo, garantindo que todas as alterações sejam aplicadas uniformemente e em tempo real.

5. **Risco de Perda de Dados**: A gestão manual também está sujeita a riscos elevados de perda de dados devido à falta de backups adequados e à dificuldade em rastrear alterações feitas nos artefatos.

1.5.2 Contribuição do Software

Diante desses desafios, a implementação de um software de gerenciamento de artefatos 3D se torna essencial. Ele não apenas automatiza tarefas dispendiosas, mas também oferece uma interface intuitiva para o gerenciamento eficiente dos artefatos. Com funcionalidades como controle de versão, posicionamento automático, integração com outras ferramentas e segurança dos dados, o software poderá minimizar erros humanos e otimizar o fluxo de trabalho. Assim, este trabalho visa desenvolver um sistema que não apenas atenda às necessidades atuais do mercado, mas também promova a rápida adoção da tecnologia 3D, contribuindo para a inovação e competitividade nas diversas áreas que utilizam cenários tridimensionais.

1.6 Organização do Trabalho

Este trabalho monográfico está organizado em cinco seções, delineadas brevemente a seguir:

A primeira seção introduz a questão em pauta e sugere uma resolução com base nos objetivos delineados.

A segunda seção apresenta a estrutura da solução, abrangendo diagramas e casos de uso, permitindo uma compreensão das demandas do projeto. Além disso, detalha a execução do sistema, as tecnologias empregadas e os fluxogramas.

A terceira seção expõe e caracteriza a interface do sistema e suas funcionalidades.

A quarta seção aborda os testes de funcionamento, conduzidos por meio de testes de caixa preta.

Por fim, são apresentadas as conclusões e as perspectivas para futuras investigações desta pesquisa.

2 ARQUITETURA DO SISTEMA

A arquitetura adotada para o desenvolvimento do sistema é baseada na arquitetura REST, que significa "Transferência de Estado Representacional" (*Representational State Transfer).* Essa arquitetura, proposta por Roy Fielding em sua tese de doutorado, estabelece um conjunto de restrições para a criação de serviços web, permitindo uma comunicação eficiente entre sistemas distribuídos (Fielding, 2000).

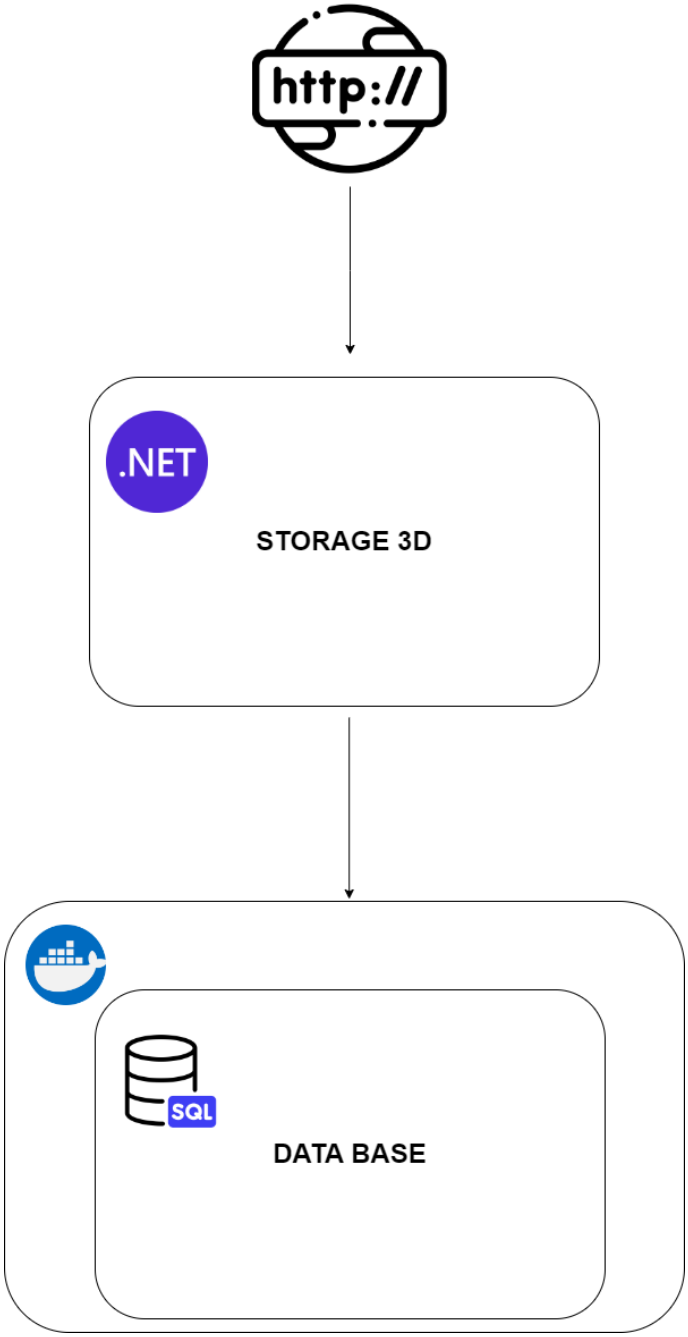
Além disso, a utilização de APIs (Application Programming Interfaces) possibilita a interação entre diferentes serviços e aplicações, facilitando a integração e a reutilização de funcionalidades. O sistema também utiliza o formato JSON (JavaScript Object Notation), amplamente empregado para entrada e exposição de dados, caracterizado por sua leveza e facilidade de leitura e escrita, tornando-o ideal para a troca de informações entre cliente e servidor (Crockford, 2006). Essa combinação favorece a ampla disseminação do sistema.

A API permite que os usuários acessem o sistema por meio de um sistema de autenticação baseado em *Bearer Authentication*. Essa abordagem utiliza tokens de segurança, conhecidos como *bearer tokens*, que devem ser incluídos no cabeçalho *Authorization* das requisições HTTP, garantindo que apenas usuários autenticados possam interagir com a API (Marcus, 2013). A partir disso, é possível realizar chamadas para edição, criação, exclusão e busca de dados. Ao empregar a arquitetura REST, facilita-se a integração e interpretação dos resultados obtidos nas interações, além de se disponibilizar documentação para simplificar a navegação do usuário.

É importante destacar a relevância de atender às principais diretrizes de segurança, utilizando autenticação, conforme mencionado anteriormente, criptografia de dados sensíveis e seguindo os principais padrões e boas práticas no desenvolvimento de software, como clean code e DDD (*Domain-Driven Design*).

A arquitetura do sistema pode ser analisada resumidamente na figura abaixo.

Figura 1 – Diagrama de arquitetura.



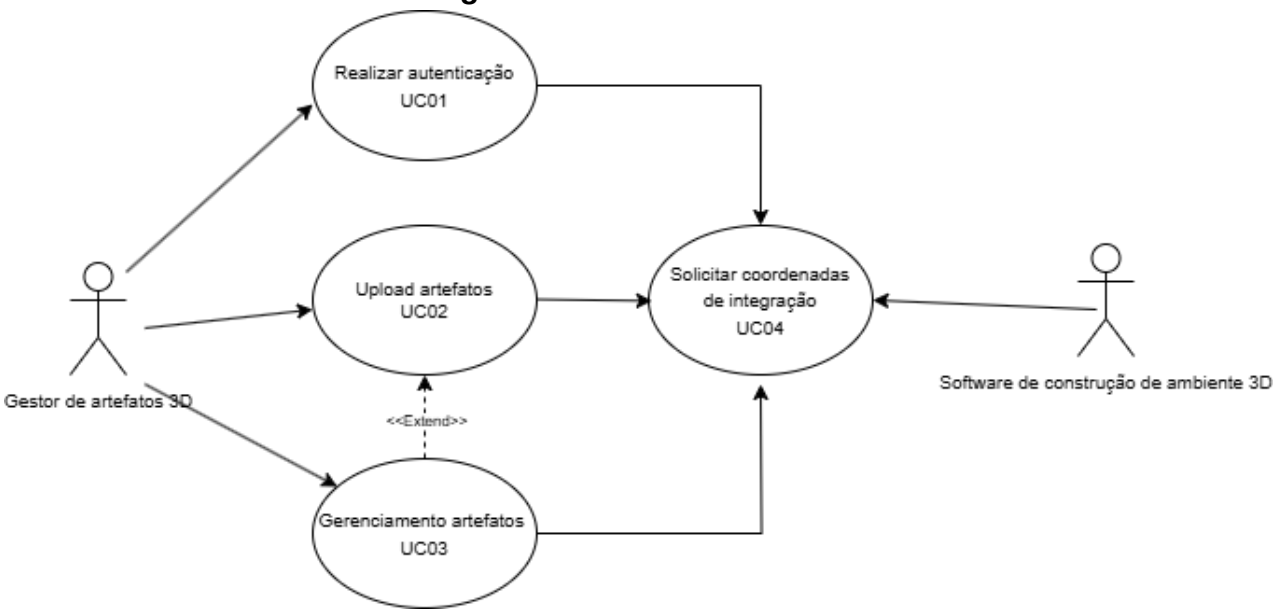
Fonte: Elaborada pelo autor (2024).

A Figura 1 ilustra a arquitetura da aplicação, juntamente com as tecnologias utilizadas em seu funcionamento. A análise da imagem revela três áreas principais, começando pelo tratamento e armazenamento dos dados. Para o desenvolvimento, foi adotada uma estrutura de banco de dados relacional, que organiza e armazena informações em tabelas compostas por linhas e colunas. Cada tabela representa uma entidade, e os dados são acessados por meio da *Structured Query Language* (SQL). O banco de dados selecionado para esta aplicação foi o SQL Server, disponibilizado

pela Microsoft. Para viabilizar o uso do banco de dados, este foi inserido em um ambiente Docker, utilizando a imagem padrão do SQL Server disponível no Docker Hub. Para a construção da API, foi utilizado o .NET 6.0, um *framework* fornecido pela *Microsoft*, amplamente reconhecido por sua eficácia na criação de grandes aplicações. A arquitetura da API segue as diretrizes REST, permitindo que a API seja acessada por meio de chamadas HTTP, transacionando os dados.

2.1 Casos de Uso

Figura 2 - Casos de uso.



Fonte: Elaborada pelo autor (2024).

O diagrama de casos de uso representa a interação do gestor de artefatos 3D com a aplicação, destacando suas atividades principais. O ator tem a capacidade de autenticar-se, efetuar o upload de artefatos, gerenciar esses artefatos e disponibilizá-los por meio de chamadas de retorno, permitindo a integração com outros softwares.

2.1.1 Descrição dos Atores

O caso de uso envolve dois atores, o gestor de artefatos 3D, que obtém acesso completo à aplicação após autenticação. Nesse contexto, o usuário pode acessar a aplicação por meio da API para executar os fluxos disponibilizados, e também, o software de construção de ambientes 3D, representando softwares externos que podem se integrar com o sistema desenvolvido neste trabalho.

2.1.2 Descrição dos casos de uso

Tabela 1 - Descrição UC01.

Caso de uso	UC01 – Realizar autenticação
Ator Principal	Gestor de artefatos 3D
Descrição	Possibilitar o usuário a se cadastrar para realizar a autenticação junto a API para acessar os demais recursos da mesma.
Pré-Condição	Inexistente
Fluxo Normal	1. Informar as credenciais
Fluxos Excepcionais	Inexistente
Pós-Condição	UC02 - Upload de artefatos, UC03 - Gerenciamento de artefatos, UC04 – Solicitar coordenadas de integração

Fonte: Elaborada pelo autor (2024).

Tabela 2 - Descrição UC02.

Caso De Uso	UC02 - Upload de artefatos
Ator Principal	Gestor de artefatos 3D
Descrição	Possibilita o usuário a fazer upload dos artefatos e seus respectivos dados, para serem armazenados e controlados pela aplicação.
Pré-Condição	UC01 - Realizar autenticação
Fluxo Normal	1. Criação Objeto 2. Upload Informações Objeto 3. Upload Imagem Objeto
Fluxos Excepcionais	1. Retorno da Inconsistência tratada ao Usuário
Pós-Condição	UC03 - Gerenciamento de artefatos, UC04 - Solicitar coordenadas de integração

Fonte: Elaborada pelo autor (2024).

Tabela 3 - Descrição UC03.

Caso de uso	UC03 - Gerenciamentos de artefatos
Ator Principal	Gestor de artefatos 3D
Descrição	Possibilita o usuário a gerenciar os artefatos existentes em sua aplicação.
Pré-Condição	UC01 - Realizar autenticação
Fluxo Normal	1. Delete Artefatos 2. Listagem Artefatos 3. Busca Individuais por Artefatos
Fluxos Excepcionais	1. Retorno da inconsistência tratada ao Usuário
Pós-Condição	UC04 - Solicitar coordenadas de integração

Fonte: Elaborada pelo autor (2024).

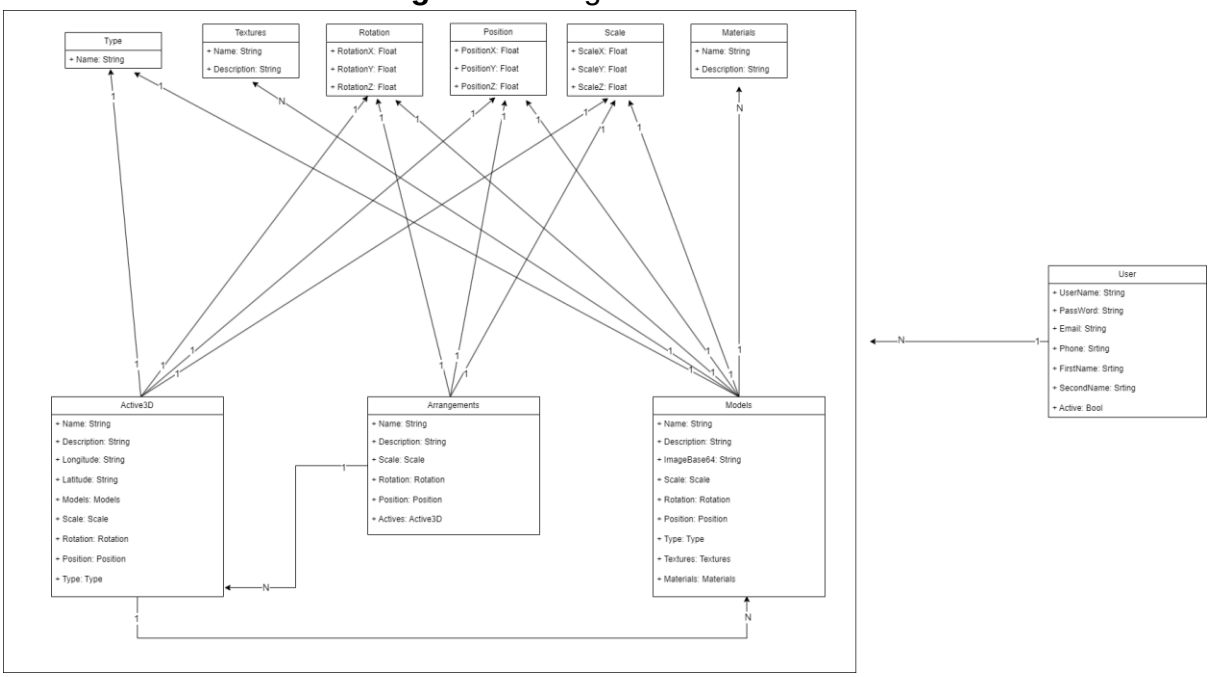
Tabela 4 - Descrição UC04.

Caso de uso	UC04 – Solicitar coordenadas de integração
Ator Principal	Gestor de artefatos 3D
Descrição	Possibilita ao usuário através das buscas e retornos em aplicação em formato JSON, se integrar com outras aplicações.
Pré-Condição	UC01 - Realizar autenticação
Fluxo Normal	1. Disponibilizar resultados no formato JSON
Fluxo Excepcionais	1. Retorno da inconsistência tratada ao Usuário
Pós-Condição	Inexistente

Fonte: Elaborada pelo autor (2024).

2.2 Diagrama de Classe

Figura 3 - Diagrama de classe.



Fonte: Elaborada pelo autor (2024).

O diagrama de classes é uma representação visual da estrutura de classes, atributos e relacionamentos de um sistema. Ao analisar o diagrama apresentado, é possível identificar as seguintes classes:

Type: Possui um atributo "Name" do tipo texto. A classe "Type" se refere ao tipo do artefato existente em questão.

Textures: Possui dois atributos: "Name", do tipo texto, e "Description", também do tipo texto. A classe "Textures" tem a função de relacionar os artefatos às texturas utilizadas no artefato em questão.

Rotation: Possui três atributos: "Rotation X", "Rotation Y" e "Rotation Z", todos do tipo float. A classe "Rotation" tem o objetivo de definir a rotação do artefato em relação ao plano cartesiano.

Position: Possui três atributos: "Position X", "Position Y" e "Position Z", todos do tipo float. A classe "Position" tem o objetivo de demarcar a posição do artefato 3D em relação ao plano cartesiano.

Scale: Possui três atributos: "Scale X", "Scale Y" e "Scale Z", todos do tipo float. A classe "Scale" tem o objetivo de demarcar a escala do artefato 3D em relação ao plano cartesiano.

Materials: Possui dois atributos: "Name" e "Description", ambos do tipo texto. A classe "Materials" tem o objetivo de especificar os materiais utilizados para compor o artefato 3D.

Essas classes são elementos de composição das classes principais de objetos:

Active3D: Possui os seguintes atributos:

"Name": do tipo texto, campo que conterá o nome do ativo 3D.

"Description": do tipo texto, campo responsável por uma breve descrição relacionada ao ativo 3D.

"Longitude": do tipo texto, atributo que contém a longitude do ativo 3D em relação ao plano esférico.

"Latitude": do tipo texto, atributo que contém a latitude do ativo 3D em relação ao plano esférico.

"Models": do tipo "Models", atributo que contém os modelos utilizados na constituição desse ativo 3D.

"Scale": do tipo "Scale", atributo que contém as escalas utilizadas na constituição desse ativo 3D.

"Rotation": do tipo "Rotation", atributo que contém as rotações utilizadas na constituição desse ativo 3D.

"Position": do tipo "Position", atributo que contém as posições utilizadas na constituição desse ativo 3D.

"Type": do tipo "Type", atributo que contém o tipo desse ativo 3D.

Os ativos 3D são elementos digitais que representam objetos, cenários ou personagens em um espaço tridimensional.

Arrangements: Possui os seguintes atributos:

"Name": do tipo texto, atributo que contém o nome do arranjo.

"Description": do tipo texto, campo responsável por uma breve descrição relacionada ao arranjo.

"Scale": do tipo "Scale", campo responsável pelas escalas do arranjo.

"Rotation": do tipo "Rotation", campo responsável pelas rotações do arranjo.

"Position": do tipo "Position", campo responsável pelas posições do arranjo.

"Actives": do tipo "Active3D", campo responsável pelos ativos 3D relacionados ao arranjo.

Arranjos em ambientes 3D referem-se à disposição e organização de elementos tridimensionais dentro de um espaço virtual.

Models: Possui os seguintes atributos:

"Name": do tipo texto, atributo que contém o nome do modelo.

"Description": do tipo texto, campo responsável por uma breve descrição relacionada ao modelo.

"Scale": do tipo "Scale", campo responsável pelas escalas do modelo.

"Rotation": do tipo "Rotation", campo responsável pelas rotações do modelo.

"Position": do tipo "Position", campo responsável pelas posições do modelo.

"Type": do tipo "Type", campo responsável pelo tipo do modelo.

"Textures": do tipo "Textures", campo responsável pelas texturas do modelo.

"Materials": do tipo "Materials", campo responsável pelos materiais utilizados no modelo.

Os modelos em ambientes 3D são representações digitais de objetos ou cenários criados em um espaço tridimensional.

Essas classes interagem entre si para fornecer as informações essenciais na criação de objetos virtuais em 3D, estando todas elas associadas à classe User.

2.3 Tecnologias Utilizadas

2.3.1 C#, (C Sharp)

Com o objetivo de priorizar a performance, robustez e integração com outras plataformas por meio de APIs, optou-se pela linguagem C# para o desenvolvimento do software. O C# é uma linguagem amplamente adotada no mercado, conhecida por suas atualizações frequentes, documentação abrangente e uma comunidade ativa.

O C# (pronuncia-se "C Sharp") é uma linguagem de programação moderna, orientada a objeto e fortemente tipada. O C# permite que os desenvolvedores criem muitos tipos de aplicativos seguros e robustos que são executados no .NET (Microsoft, 2024a).

2.3.2 .NET

Usar o .NET em um projeto C# oferece benefícios significativos. O framework fornece estabilidade, desempenho e uma vasta biblioteca de classes que aceleram o desenvolvimento e facilitam a manutenção do código. Com interoperabilidade entre várias linguagens e suporte multiplataforma, os desenvolvedores têm flexibilidade para criar aplicativos para diferentes dispositivos e sistemas operacionais. Com

suporte contínuo da Microsoft e uma comunidade ativa, o .NET é uma escolha sólida para projetos que exigem escalabilidade, confiabilidade e produtividade.

O .NET é uma plataforma de desenvolvedor gratuita, multiplataforma e de código aberto para a criação de muitos tipos de aplicativos. Ele pode executar programas escritos em vários idiomas, com C# sendo o mais popular. Ele depende de um tempo de execução de alto desempenho que é usado na produção por muitos aplicativos de alta escala (Microsoft, 2024a).

2.3.3 SQL SERVER

Usar o SQL Server no projeto oferece várias vantagens. Além da confiabilidade e desempenho reconhecidos, o SQL Server possui recursos avançados como suporte a transações ACID, procedimentos armazenados e integração com tecnologias Microsoft. Com ferramentas de administração poderosas e segurança avançada, o SQL Server é uma escolha sólida para projetos que exigem um banco de dados confiável e escalável.

O Microsoft SQL Server é um sistema de gerenciamento de banco de dados relacional (RDBMS). Aplicativos e ferramentas se conectam a uma instância ou banco de dados do SQL Server e se comunicam usando Transact-SQL (T-SQL) (Microsoft, 2024b).

2.3.4 DOCKER

Usar Docker no projeto oferece benefícios significativos, simplificando o desenvolvimento, implantação e execução de aplicativos. Os contêineres garantem consistência entre ambientes, reduzindo o tempo de configuração e evitando problemas de compatibilidade. Além disso, são altamente escaláveis e facilitam a implementação de microserviços. Com suporte a orquestradores como *Kubernetes, Docker Swarm*, gerenciar um grande número de contêineres em produção é fácil. Em suma, o Docker é essencial para construir e implantar aplicativos de forma eficiente, confiável e escalável.

Docker é uma plataforma aberta para desenvolvimento, envio e execução de aplicativos. O *Docker* permite separar seus aplicativos de sua infraestrutura para que você possa entregar software rapidamente. Com o *Docker*, você pode gerenciar sua infraestrutura da mesma forma que gerencia seus aplicativos. Aproveitando as

vantagens das metodologias do *Docker* para envio, teste e implantação de código, você pode reduzir significativamente o atraso entre escrever o código e executá-lo na produção (Docker, 2024).

2.3.5 Fluent Validation

Foi utilizado *FluentValidation* no projeto devido à sua flexibilidade e poder na validação de entrada de dados. Ele oferece uma maneira intuitiva de definir regras de validação, suportando uma variedade de cenários, incluindo validações complexas e condicionais. Com o *FluentValidation*, você garante a conformidade dos dados com as regras de negócios, melhorando a integridade e qualidade dos dados. Sua integração fácil com estruturas populares, como ASP.NET Core, o torna uma escolha poderosa para projetos web e aplicativos .NET em geral.

FluentValidation é uma biblioteca .NET para construir regras de validação fortemente tipadas (FluentValidation, 2024).

2.3.6 Swagger

Foi utilizado o Swagger no projeto por sua capacidade de simplificar a documentação, teste e interação com a API. Com o Swagger, gerar uma documentação interativa e testar endpoints da API diretamente no navegador se tornou fácil. Sua integração simples com várias estruturas e linguagens de programação, incluindo ASP.NET Core, o torna uma escolha conveniente para projetos de qualquer escala. Em resumo, o Swagger é uma ferramenta valiosa para melhorar a usabilidade e a acessibilidade da API no meu projeto.

Swagger é um conjunto de ferramentas de código aberto desenvolvidas em torno da especificação OpenAPI que pode ajudá-lo a projetar, construir, documentar e consumir APIs REST (Swagger, 2024).

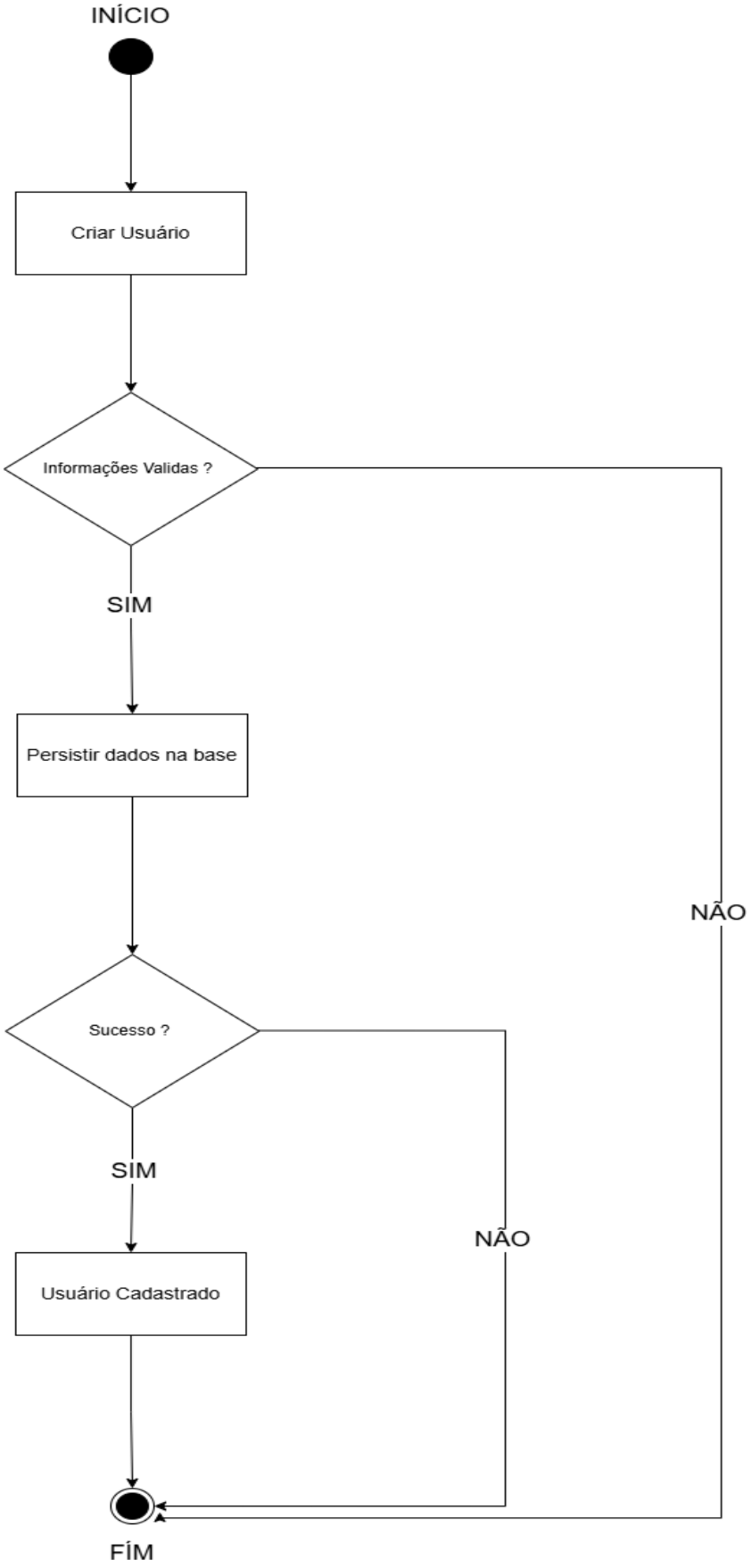
2.4 Fluxogramas

Neste segmento, serão exibidos os diagramas de fluxo de algoritmos que ilustram as funcionalidades essenciais da plataforma. Esses diagramas são ferramentas visuais de modelagem que possibilitam a representação da sequência de processos ou ações contidas em um algoritmo, facilitando a comunicação visual sobre

sua execução e identificando os pontos de decisão, juntamente com suas opções de execução.

2.4.1 Fluxograma de criação de usuário

Figura 4 - Fluxograma de criação de usuário.



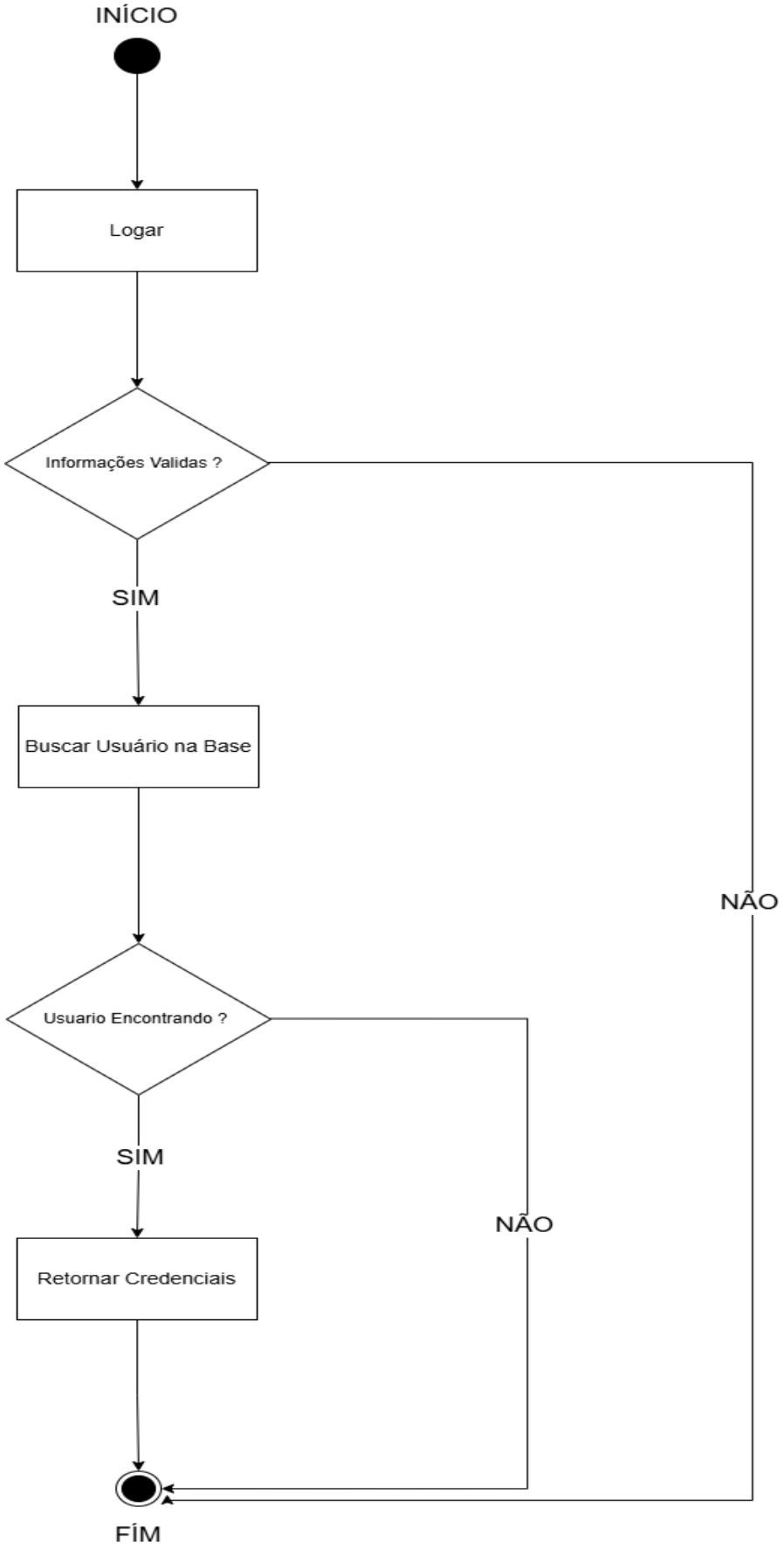
Fonte: Elaborada pelo autor (2024).

A criação do usuário é obrigatória para habilitar todas as funcionalidades da aplicação e para associar corretamente os artefatos gerenciáveis ao usuário. Para compreender as sequências de ações desta operação, a Figura 5 ilustra o processo.

1. O fluxo tem início com uma requisição do tipo POST para a criação do usuário (CreateUser).
2. Após o envio das informações, é realizada uma verificação das mesmas, incluindo a verificação da existência dos dados de registro único (E-mail) na base de dados.
    a. Caso as Informações sejam válidas o software tentará persistir as informações na base de dados.
    b. Caso as informações sejam inválidas o software retornará um erro com o status 400 (Bad Request), e com a mensagem tratada mencionando os pontos de falha, e o processo chegará ao fim.
3. Após as informações enviadas e conferidas é feita uma tentativa de persistir os dados na base de dados.
    a. Caso as informações sejam persistidas com sucesso, o usuário será criado com sucesso.
    b. Caso ocorra um erro ao persistir as informações é retornado um erro com o status 500 e com uma mensagem de erro padrão, e o processo chegará ao fim.
4. Caso tudo ocorra como esperado, o usuário receberá um retorno com o status 201 e uma mensagem de usuário cadastrada com sucesso, e o processo chegará ao fim.

2.4.2 Fluxograma de autenticação

Figura 5 - Fluxograma de autenticação.



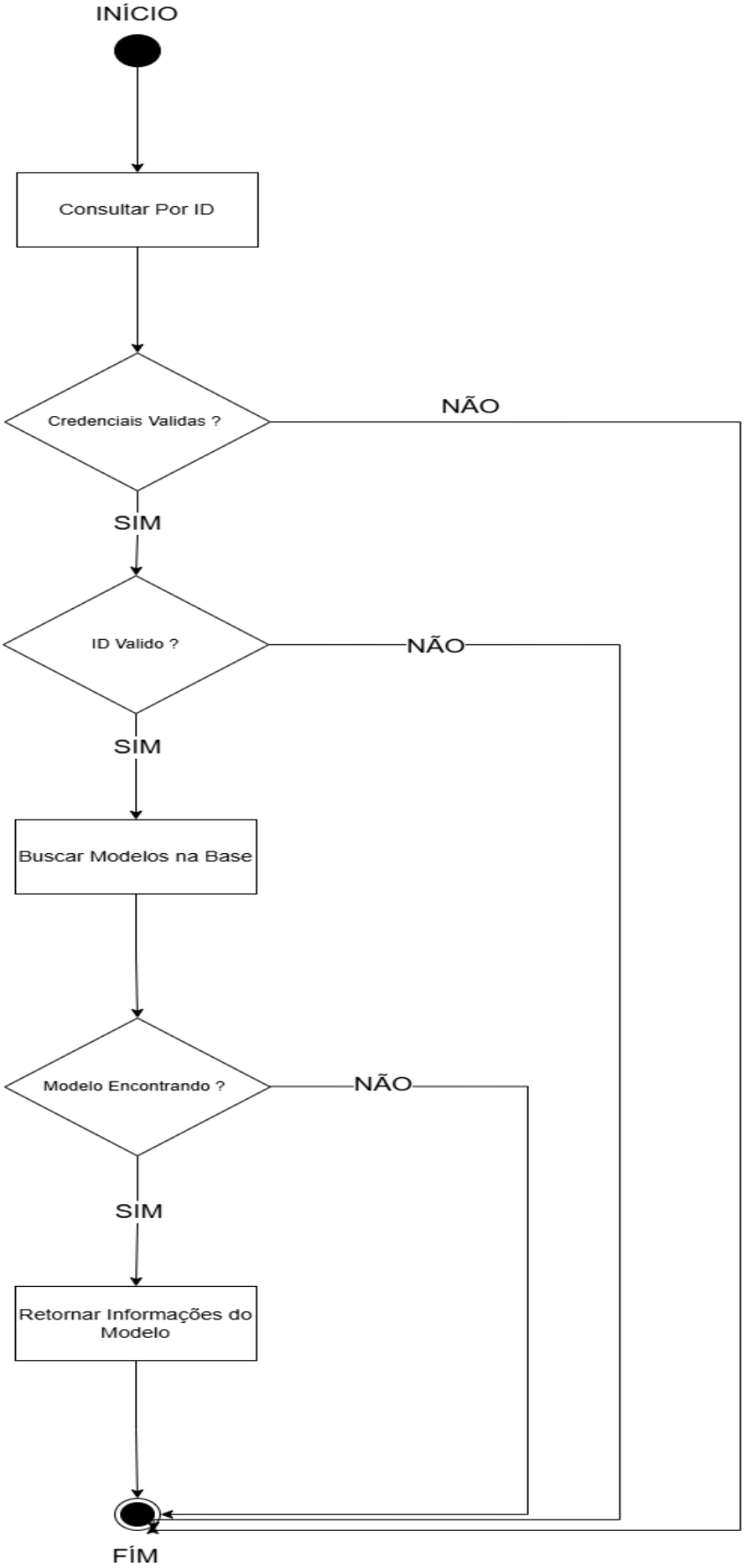
Fonte: Elaborada pelo autor (2024).

A autenticação do usuário é essencial para a criação de suas credenciais, permitindo o acesso às demais funcionalidades da aplicação, as quais só podem ser alcançadas por meio da validação de um token válido, a Figura 6 ilustra o processo.

1. O fluxo tem início com uma requisição do tipo POST com as informações do usuário.
2. Após o envio da requisição é verificado se o token do usuário possui permissão e validade para acessar o recurso solicitado.
    a. Caso o token seja válido a requisição continuará o seu fluxo.
    b. Caso o token seja invalido será retornado o erro 401 e o processo se encerra.
3. Após a verificação do token é realizado uma verificação dos parâmetros fornecidos pelo usuário.
    a. Caso as Informações sejam válidas o software buscará o usuário na base de dados.
    b. Caso as informações sejam inválidas o software retornará um erro com o status 400,(Bad Request), e com a mensagem tratada mencionando os pontos de falha, e o processo chegará ao fim.
5. Com as informações corretas é realizada uma busca do usuário na base.
    a. Caso o usuário for encontrado, as informações de credenciais do mesmo serão retornadas juntamente com o status 200.
    b. Caso o usuário não for encontrado, será retornado o status 200 com uma mensagem de usuário não encontrado.

2.4.3 Fluxograma de consulta de modelo por ID

Figura 6 - Fluxograma de consulta de modelo por ID.



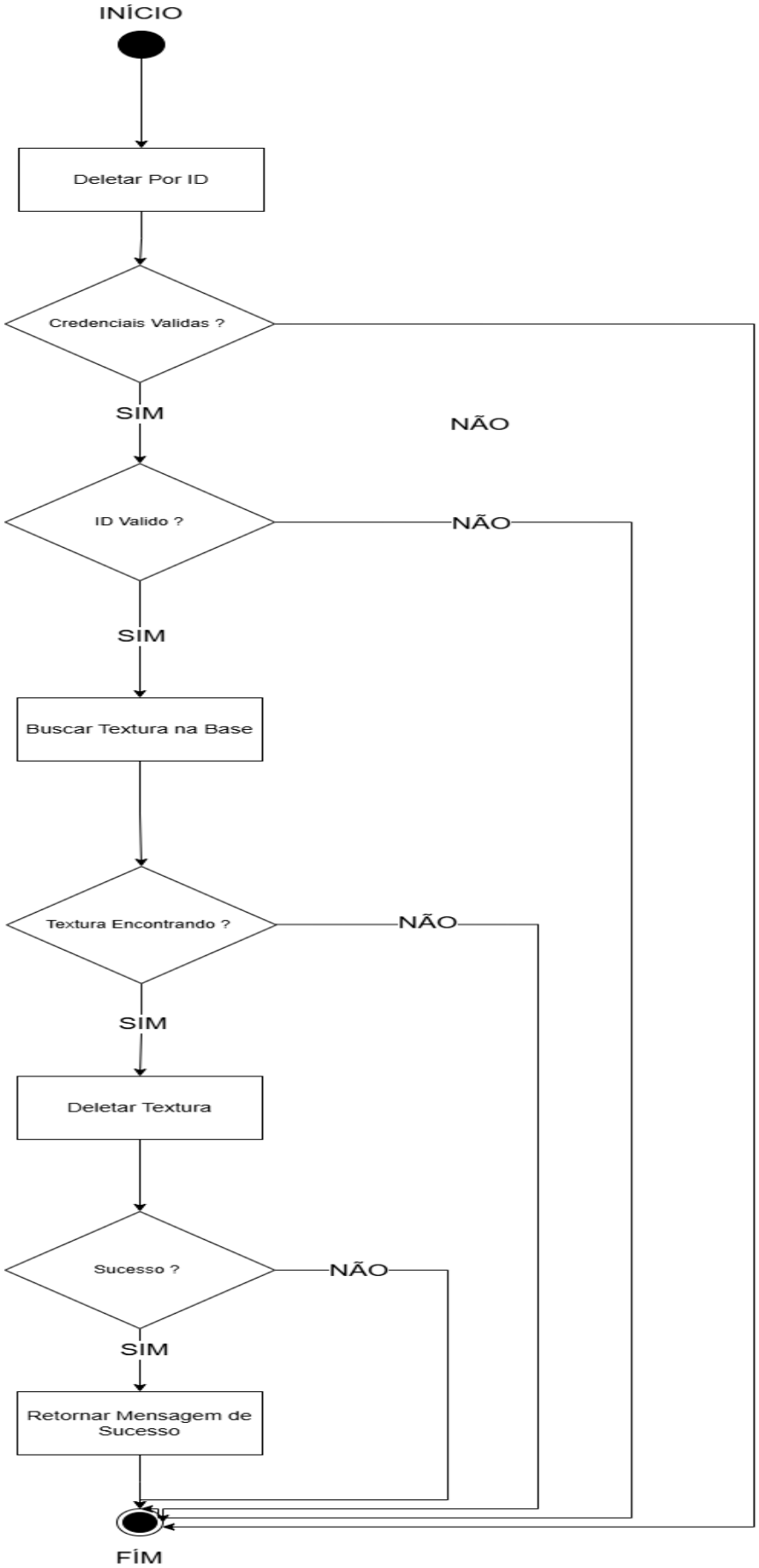
Fonte: Elaborada pelo autor (2024).

O diagrama de fluxo acima ilustra o procedimento de consulta por ID da classe "Modelo", sendo este padrão aplicável a todas as demais classes presentes no projeto.

1. O fluxo se inicia com uma requisição do tipo GET que tem como parâmetro o id do modelo buscado.
2. Após o envio da requisição é verificado se o token do usuário possui permissão e validade para acessar o recurso solicitado.
    a. Caso o token seja válido a requisição continuará o seu fluxo.
    b. Caso o token seja invalido será retornado o erro 401 e o processo se encerra.
3. Após a verificação do token é realizado uma verificação no parâmetro fornecido pelo usuário.
    a. Caso o parâmetro seja válido, a requisição continuará o seu fluxo.
    b. Caso o parâmetro seja invalido será retornado o erro 400 com uma mensagem de erro tratada informando a invalidação do parâmetro.
4. Após a verificação do parâmetro é realizada uma busca na base de dados para procurar um registro correspondente ao parâmetro fornecido.
    a. Caso seja encontrado um registro equivalente as informações do mesmo serão retornadas para o usuário, acompanhadas do status 200.
    b. Caso não seja encontrado um registro equivalente, será retornado o status 200 acompanhado de uma mensagem “Modelo não encontrado”.

2.4.4 Fluxograma de exclusão de texturas

Figura 7 - Fluxograma de exclusão de texturas



Fonte: Elaborada pelo autor (2024).

O diagrama de fluxo acima ilustra o procedimento de exclusão da classe "Textura", sendo este padrão aplicável a todas as demais classes presentes no projeto.

1. O fluxo se inicia com uma requisição do tipo DELETE que tem como parâmetro o id do modelo a ser excluído.
2. Após o envio da requisição é verificado se o token do usuário possui permissão e validade para acessar o recurso solicitado.
    a. Caso o token seja válido a requisição continuará o seu fluxo.
    b. Caso o token seja invalido será retornado o erro 401 e o processo se encerra.
3. Após a verificação do token é realizado uma verificação no parâmetro fornecido pelo usuário.
    a. Caso o parâmetro seja válido, a requisição continuará o seu fluxo.
    b. Caso o parâmetro seja invalido será retornado o erro 400 com uma mensagem de erro tratada informando a invalidação do parâmetro.
4. Após a verificação do parâmetro é realizada uma busca na base de dados para procurar um registro correspondente ao parâmetro fornecido.
    a. Caso seja encontrado um registro equivalente, o sistema iniciará uma operação de exclusão no mesmo.
    b. Caso não seja encontrado um registro equivalente, será retornado o status 200 acompanhado de uma mensagem “Textura não encontrado”.
5. Após a busca do registro é feita a exclusão.
    a. Caso a exclusão ocorra com sucesso é retornada uma mensagem de sucesso acompanhada do status 200.
    b. Caso ocorra falha na exclusão será retornada uma mensagem de erro padrão acompanhada do status 500.

2.5 Considerações Finais

O objetivo desta seção foi apresentar a organização da plataforma por meio da demonstração de um esquema arquitetônico, cenários de uso e esquema de entidades, além de introduzir as tecnologias utilizadas e os diagramas para processos específicos. É relevante destacar que quaisquer outros fluxos não abordados nesta

seção foram deliberadamente omitidos, seja por se tratarem de algoritmos elementares ou por serem essencialmente similares aos já descritos.

3 FUNCIONAMENTO DO SISTEMA

Este capítulo proporciona uma análise da operacionalidade prática do sistema em discussão, por meio da simulação de um usuário que interage com o sistema via API. O conteúdo inclui a apresentação de imagens e uma descrição minuciosa das funcionalidades selecionadas para demonstração.

3.1 Cadastro de Usuário

Para cadastrar um novo usuário na aplicação por meio da API, é necessário fornecer os seguintes dados:

UserName: String, campo onde o nome do novo usuário é imputado.

PassWord: String, campo onde a senha do novo usuário é imputada.

Email: String, campo onde o email do novo usuário é imputado.

Phone: String, campo onde o número de contato do novo usuário é imputado.

FirstName: String, campo onde o primeiro nome do novo usuário é imputado.

SecondName: String, campo onde o restante do nome do novo usuário é imputado.

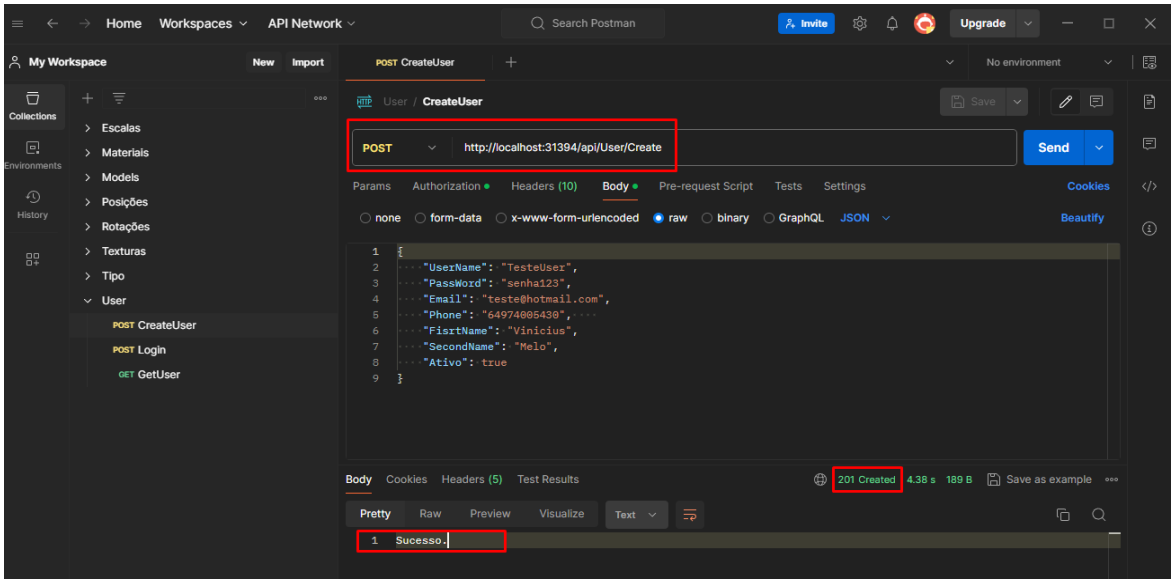
Active: Bool, campo onde é possível verificar se o novo usuário está ativo ou não, incialmente sempre recebera o valor true.

O terminal de conexão que possibilita o acesso ao serviço para a realização dessa operação é designado como: **api/User/Create**, podendo também ser referido de maneira simplificada como caminho de acesso.

A figura 8 ilustra o processo bem-sucedido de adição de um novo usuário, utilizando a ferramenta Postman. O Postman é uma aplicação amplamente utilizada por desenvolvedores para simular requisições HTTP, permitindo uma interação prática com APIs. Neste caso específico, a API está sendo executada localmente no endereço http://localhost:31394, onde "localhost" se refere ao próprio computador em que a aplicação está rodando e "31394" é a porta designada para essa comunicação.O processo de adição de um novo usuário ocorre através de uma requisição do tipo POST, que é utilizada para enviar dados ao servidor. Nesse contexto, os dados incluem informações relevantes sobre o novo usuário, como nome, e-mail e senha. Ao utilizar o Postman, o usuário insere a URL da API e seleciona o método apropriado (neste caso, POST), além de fornecer os dados necessários para

o cadastro do novo usuário. Após o envio da requisição, o Postman apresenta a resposta do servidor, que pode confirmar se o novo usuário foi adicionado com sucesso ou indicar se houve algum erro durante o processo. Essa dinâmica demonstra como a ferramenta Postman facilita a interação com APIs e torna o procedimento de adição de usuários mais acessível, mesmo para aqueles que não possuem um conhecimento técnico aprofundado.

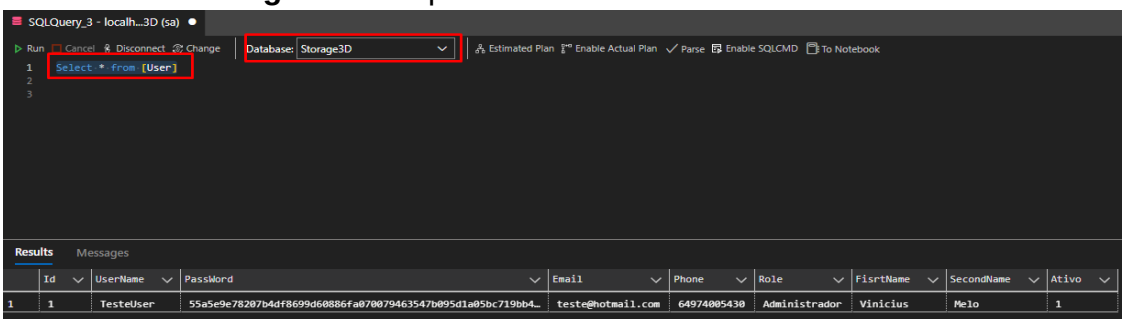
Figura 8 - Captura de Tela Postman Create User.



Fonte: Elaborada pelo autor (2024).

A Figura 9 apresenta os dados consolidados no banco de dados, utilizando o **Azure Data Studio** para a consulta das informações. É possível observar, em destaque na imagem, que o banco de dados utilizado para armazenar as informações é denominado **Storage3D**. Adicionalmente, a tabela **User**, na qual a consulta foi realizada, também está evidenciada. Por fim, na parte inferior da figura, são exibidos os dados resultantes da operação de cadastro que foram salvos no banco de dados.

Figura 9 - Captura de Tela Azure Data Studio.



Fonte: Elaborada pelo autor (2024).

3.2 Login do Usuário

Conforme já citado, para acessar as demais funcionalidades da API é necessário da autenticação do usuário, esta é feita através do Login do mesmo. Para autenticar o usuário na aplicação é necessário fornecer os seguintes dados:

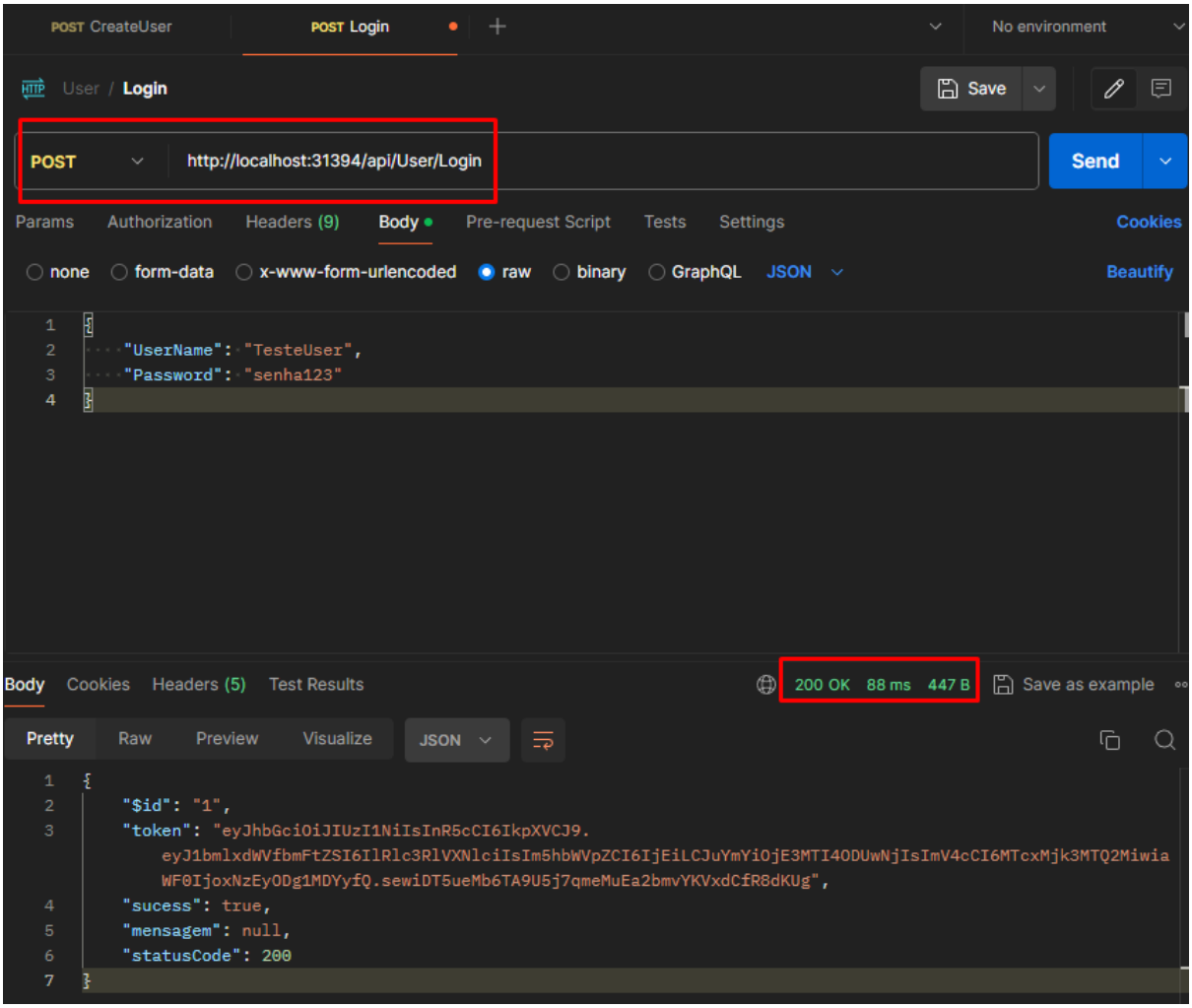
UserName: String, campo onde o nome do usuário previamente cadastro é imputado.

PassWord: String, campo onde a senha do usuário previamente cadastro é imputado.

O caminho de acesso para realizar esta operação é : **api/User/Login.**

A figura 10 a seguir representa a bem-sucedida autenticação do usuário, fornecendo os dados citados acima.

Figura 10 - Captura de Tela Postman User Login.



Fonte: Elaborada pelo autor (2024).

Na figura 10, observa-se que é retornado à variável **“token”** um token JWT (JSON Web Token), o qual será utilizado para acessar as demais funcionalidades da API. Este token é fundamental para a autenticação e autorização do usuário em futuras requisições. Além disso, é possível notar o retorno do campo **id**, que representa a identificação única do usuário no banco de dados. Outro aspecto relevante é a variável **success**, que apresenta o valor **true**, confirmando assim o êxito da chamada realizada. Esses elementos são essenciais para garantir a integridade e a segurança das operações subsequentes na API.

3.3 Consulta de Materiais por ID

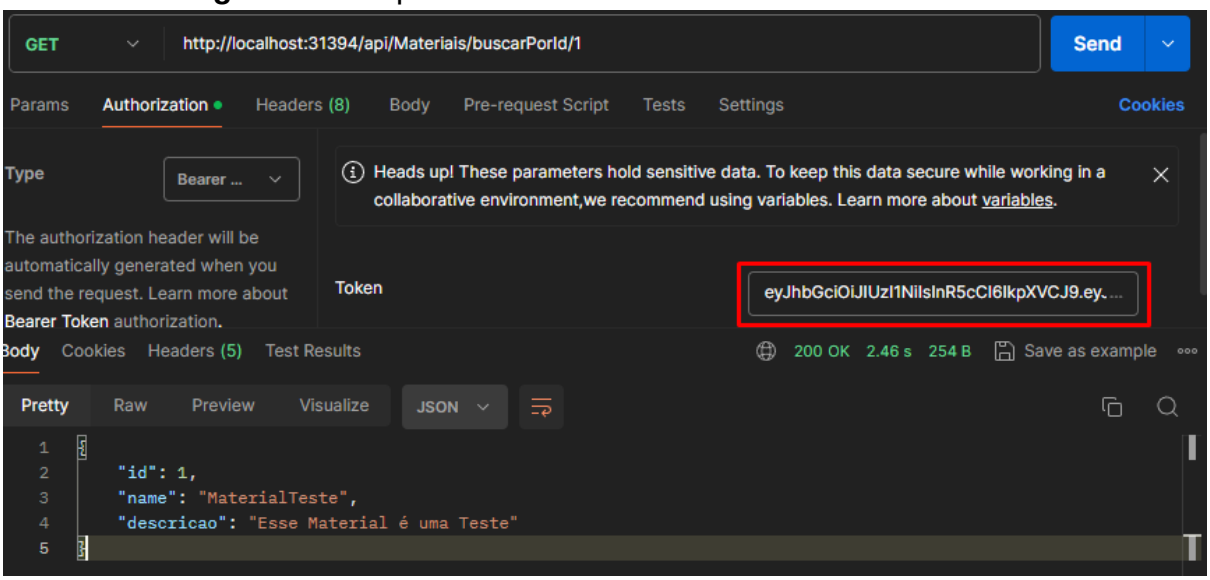
Para consultar um material previamente cadastrado na aplicação por meio da API, é necessário fornecer os seguintes dados:

Id: int, campo onde o id único do material que deseja ser buscado é imputado.

O caminho de acesso para realizar esta operação é: **api/Materiais/buscarPorId/id**

A figura 11 a seguir representa a bem-sucedida busca do material fornecendo o id citado acima.

Figura 11 - Captura de Tela Postman Material BuscaPorId.



Fonte: Elaborada pelo autor (2024).

Observa-se a utilização do token de autenticação mencionado anteriormente para acessar o método de busca, o qual retorna o material identificado pelo ID 1, solicitado por meio do parâmetro na URL. Além disso, é possível analisar no retorno

os atributos previamente cadastrados, que incluem o ID, o nome e a descrição do material encontrado durante a busca.

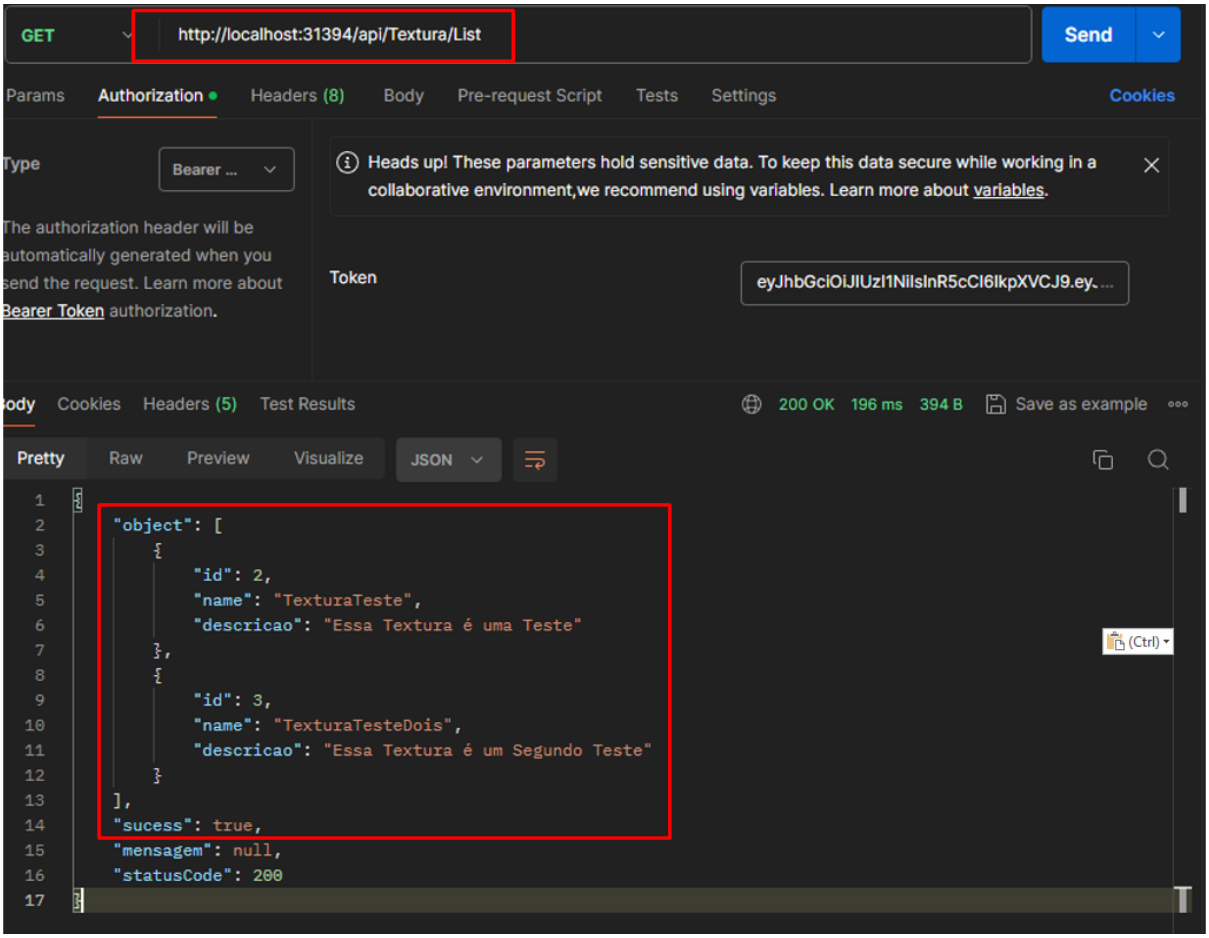
3.4 Listagem de Texturas

Para consultar e listar as texturas previamente cadastradas na aplicação por meio da API, não é necessário fornecer nenhum parâmetro adicional.

O caminho de acesso para realizar esta operação é : /api/Textura/List

A figura 12 representa a bem-sucedida busca das texturas, retornando uma lista com todas as texturas cadastrados na base de dados.

Figura 12 - Captura de Tela Postman Listagem Texturas.



Fonte: Elaborada pelo autor (2024).

A análise da figura 12 revela que, dentro da variável "object", estão contidos os dados das texturas previamente cadastradas no banco de dados. O retorno inclui as variáveis padrão das Texturas, tais como o ID único, o nome e a descrição,

apresentados em formato de lista. Além disso, observa-se a variável "success" com o valor "true", confirmando o êxito da operação realizada.

3.5 Deleção de Textura

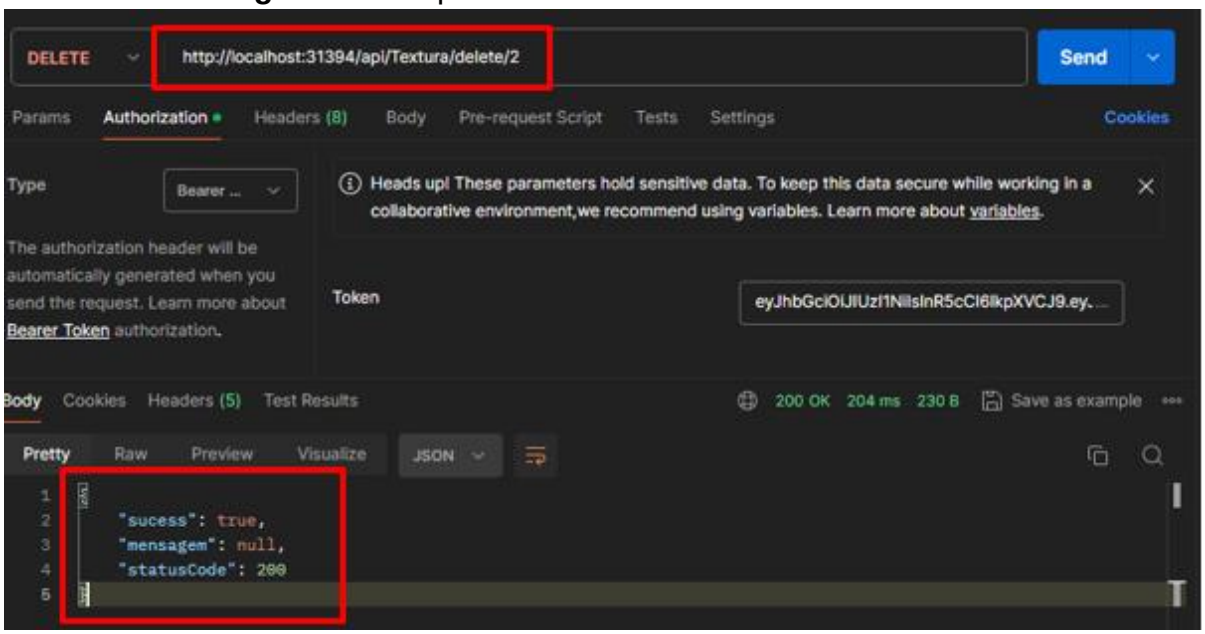
Para deletar uma textura previamente cadastrada na aplicação por meio da API, é necessário fornecer os seguintes dados:

Id: String, campo onde o id único da textura que deseja ser deletada é imputado.

O caminho de acesso para realizar esta operação é : /api/Textura/delete/id

A figura 13 representa a bem-sucedida deleção da textura, após a execução do método de exclusão.

Figura 13 - Captura de Tela Postman Deleta Textura.



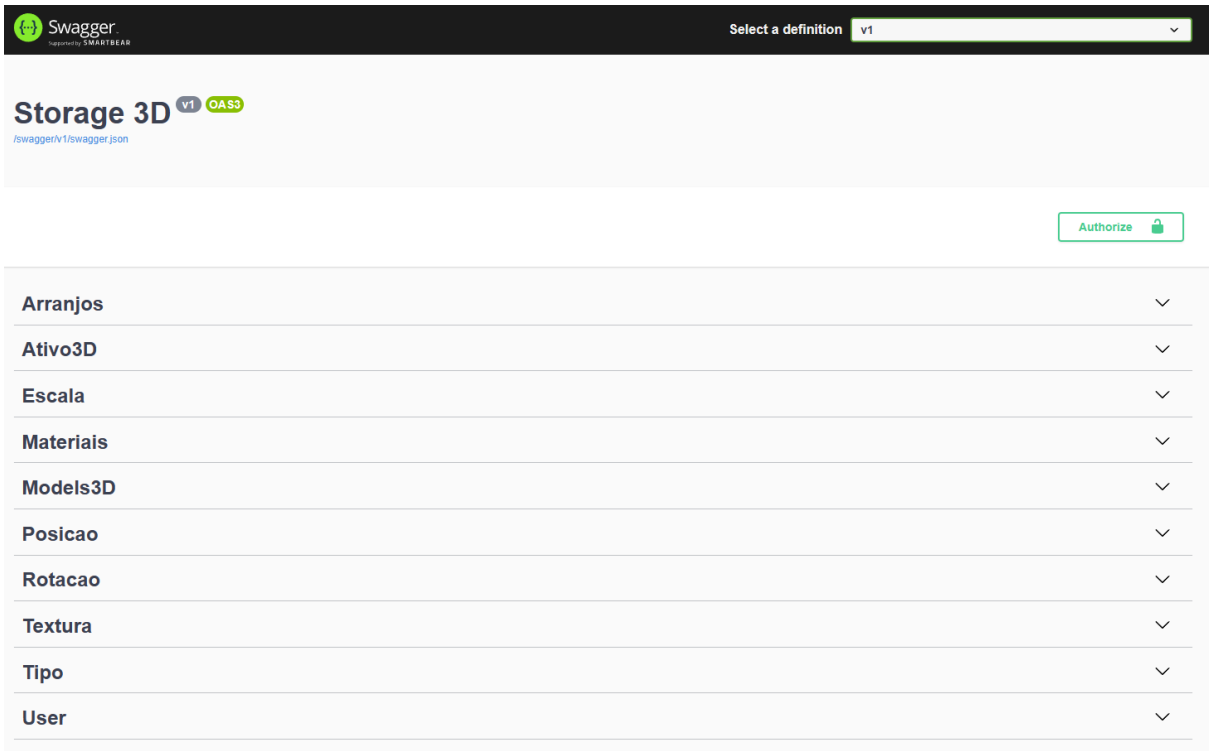
Fonte: Elaborada pelo autor (2024).

Na Figura 13, é possível observar o ID único "2" sendo passado na URL do método de exclusão. Logo abaixo, encontra-se o retorno da chamada, no qual a variável "success" apresenta o valor "true". Esse resultado indica que a operação foi realizada com êxito e que a textura associada ao ID único 2 foi devidamente excluída da base de dados.

3.6 Documentação

A documentação de uma API é essencial para garantir uma integração eficiente e precisa de seus serviços em aplicações desenvolvidas por terceiros. Através de uma documentação clara e completa, os desenvolvedores podem compreender facilmente como consumir os recursos oferecidos pela API, eliminando assim possíveis erros e reduzindo o tempo de desenvolvimento. Além disso, uma boa documentação permite que os usuários finais tenham uma experiência satisfatória ao utilizar a API, aumentando a confiabilidade e a adoção da mesma. Por consequência, a documentação é um fator fundamental para o sucesso de uma API, pois ela garante a comunicação efetiva entre os sistemas e os usuários, facilitando a integração e o uso dos serviços oferecidos.

Figura 14 - Captura de Tela Swagger.



Fonte: Elaborada pelo autor (2024).

De acordo com a Figura 15, é perceptível a presença de um botão "Authorize" que, ao ser acionado, requisitará o token Bearer JWT. Abaixo desse botão, estão listados em seções separadas os conjuntos de rotas que têm entidades em comum.

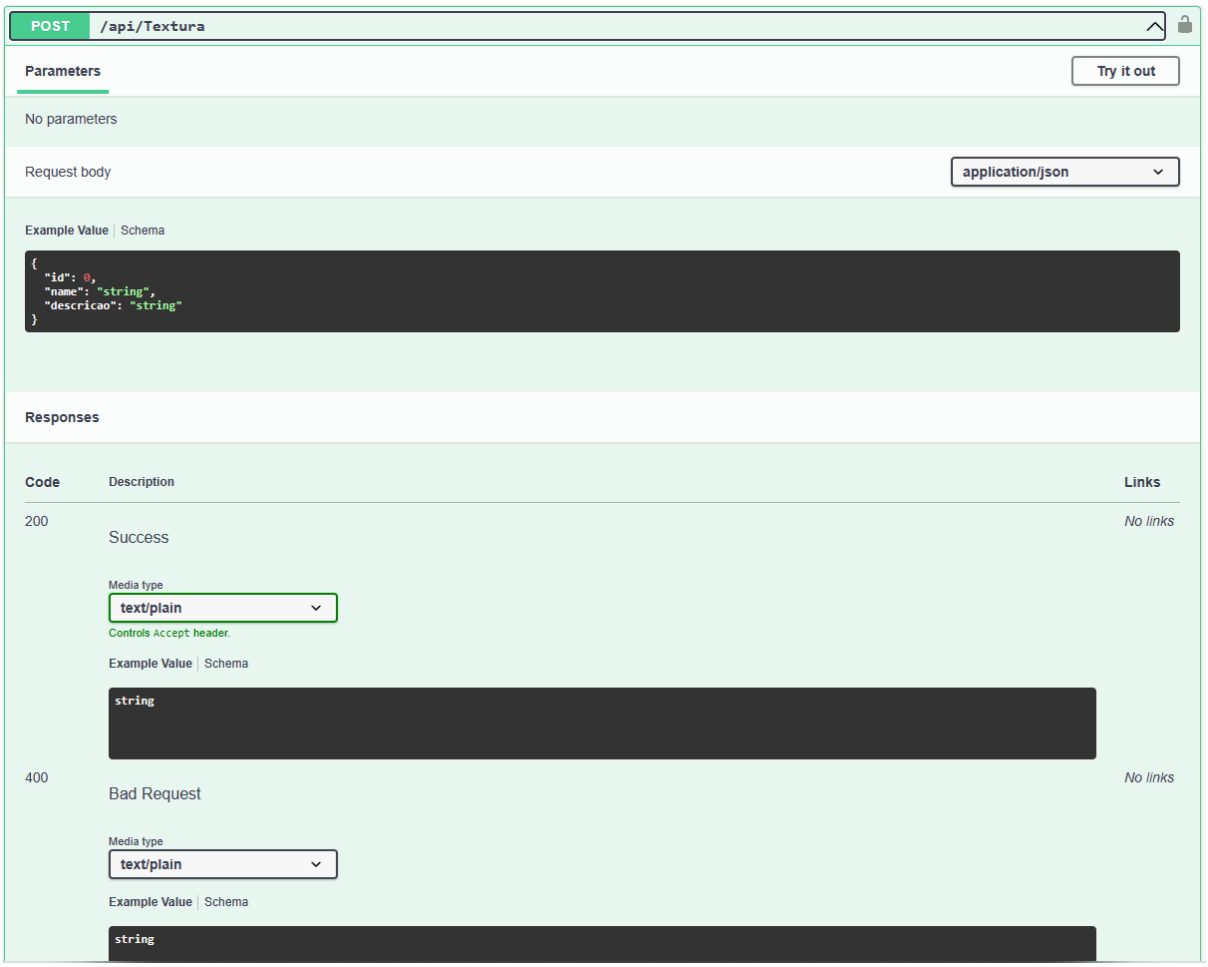
Ao clicar em qualquer uma dessas seções, elas se expandirão, exibindo as rotas disponíveis conforme ilustrado na imagem.

Figura 15 - Captura de Tela Swagger Métodos.

Fonte: Elaborada pelo autor (2024).

De acordo com a figura 16, é possível observar os métodos REST existentes para cada objeto, assim o usuário consegue identificar com facilidade as possibilidades de interação com as ferramentas proporcionadas pela API, também pode se notar que as rotas estão expostas para realizar a chamada para as mesmas, ao clicar em uma chamada específica é expandido uma tela detalhando sobre o item selecionado, conforme imagem abaixo.

Figura 16 - Captura de Tela Swagger Método Expandido.



Fonte: Elaborada pelo autor (2024).

De acordo com a Figura 16, é possível visualizar de maneira detalhada os requisitos para o método de cadastro de texturas. Ao analisar a parte superior da imagem, observa-se o caminho do método **"/api/Textura"**, precedido pelo seu tipo de chamada, que neste caso é **POST**, uma vez que se trata de um método destinado ao cadastro, conforme as convenções do **Representational State Transfer (REST)**. Logo abaixo, encontra-se a área denominada **"Request Body"**, responsável por exibir os parâmetros que a chamada pode receber em seu corpo. Os parâmetros incluem **"Name"** e **"Descrição"**, sendo imprescindível que ambos os valores sejam informados para que a operação seja realizada com sucesso.

Por fim, é importante destacar os possíveis códigos de status HTTP que podem ser retornados pela chamada. O código **200** indica que a requisição foi bem-sucedida e que o servidor retornou a resposta esperada. O código **400**, por sua vez, sinaliza uma solicitação malformada, indicando que houve um erro na sintaxe da requisição ou que os dados fornecidos não atendem aos requisitos necessários. A apresentação

desses códigos de status, juntamente com seus possíveis retornos, facilita a compreensão e a integração dos usuários com a API.

3.7 Considerações Finais

Em resumo, a documentação e o entendimento do fluxo e funcionamento da aplicação são componentes essenciais para o desenvolvimento e a manutenção de sistemas robustos e eficazes. Uma documentação clara e bem estruturada não apenas fornece informações vitais sobre a arquitetura, os componentes e os fluxos de trabalho da aplicação, mas também facilita a colaboração entre os membros da equipe e agiliza o processo de integração e implementação de novas funcionalidades. Compreender profundamente o funcionamento da aplicação é fundamental para tomar decisões informadas durante o desenvolvimento, aprimoramento e resolução de problemas, garantindo, assim, que a aplicação atenda às necessidades dos usuários finais de forma eficiente e confiável. Em última análise, uma documentação completa, aliada a um conhecimento sólido sobre o funcionamento da aplicação, é a chave para o sucesso contínuo do projeto e a satisfação do cliente.

4 TESTES DE FUNCIONAMENTO

4.1 Considerações Iniciais

Para realizar os testes na plataforma, foi utilizada a metodologia da caixa preta, a qual tem como premissa o foco nas funcionalidades e não em como essas funcionalidades foram desenvolvidas em nível de código. Os testes foram aplicados aos UC01, UC02, UC03 e UC04, que foram as funcionalidades da plataforma escolhidas para serem analisadas.

4.2 Teste de Caixa Preta

Um teste de caixa preta é uma metodologia de teste em engenharia de software na qual o examinador avalia a funcionalidade de um sistema sem ter conhecimento interno de sua estrutura ou implementação. Nesse tipo de teste, o foco está nas entradas e saídas do sistema, tratando-o como uma "caixa preta", onde o examinador não tem acesso ao código-fonte ou à lógica interna do programa. Em vez disso, o teste é conduzido com base nas especificações, requisitos ou documentação disponíveis. Os casos de teste são projetados para verificar se o sistema opera conforme o esperado, assegurando que as entradas produzem as saídas desejadas. O objetivo é garantir que o software atenda aos requisitos do usuário final e funcione corretamente em uma variedade de cenários, sem a necessidade de entender os detalhes técnicos internos do sistema. De acordo com Pressman (2011), os testes de caixa preta são fundamentais para identificar erros nas funcionalidades do software, assegurando que ele opere de acordo com as expectativas dos usuários e os requisitos estabelecidos.

4.3 Validação dos Casos De Teste

A verificação dos casos de teste é um passo crucial no processo, visto que garante a relevância, precisão e eficácia desses casos na identificação de problemas ou falhas no sistema.

4.3.1 Autenticação

No presente teste, o caso de uso UC01, relacionado à validação de ações de autenticação, foi avaliado. Durante o processo, todas as informações necessárias foram preenchidas para verificar seu funcionamento, resultando em um retorno de estado correto e as saídas esperadas.

Tabela 5 - Teste UC01.

Caso de uso	Condição de entrada	Classes válidas	Classes inválidas
UC01 - Autenticar	Inserir nome do usuário	1. Nome de usuário no formato de texto.	2. Nome de usuário vazio
	Inserir a senha	3. Senha no formato de texto.	4. Senha Vazia
	Enviar uma solicitação ao EndPoint de Autenticação	5. Solicitação deve chegar ao endPoint do back-end e validar os campos.	6. Enviar requisição sem os campos necessários
			7. Falhas internas da aplicação
			8. Não obter retorno da aplicação
		9. Retornar credenciais de acesso do usuário	10. Retorno com dados inválidos

Fonte: Elaborada pelo autor (2024).

Tabela 6 - Resultado do teste UC01.

Condição de entrada	Classe	Resposta

Inserir nome do usuário	1. Nome de usuário no formato de texto	O sistema aceitou o nome do usuário do tipo texto
	2. Nome de usuário vazio	O sistema rejeitou o nome de usuário vazio
Inserir a senha	3. Senha no formato de texto.	O sistema aceitou a senha do tipo texto
	4. Senha Vazia	O sistema rejeitou a senha vazia
Enviar uma solicitação ao EndPoint de Autenticação	5. Solicitação deve chegar ao endPoint do back-end e validar os campos.	O sistema valida os campos e dá continuidade na requisição.
	6. Enviar requisição sem os campos necessários	O sistema rejeitou a requisição
	7. Falhas internas da aplicação	O sistema retorna mensagem de erro padrão para falhas internas inesperadas
	8. Não obter retorno da aplicação	O sistema não retornar informações esperadas para a requisição
	9. Retornar credenciais de acesso do usuário	O Sistema retornou credenciais do usuário
	10. Retorno com dados inválidos	O Sistema retornou credenciais invalidas

Fonte: Elaborada pelo autor (2024).

4.3.2 Upload de artefatos

No presente teste, o caso de uso UC02, relacionado à validação de ações de upload de artefatos, foi avaliado. Durante o processo, todas as informações necessárias foram preenchidas para verificar seu funcionamento, resultando em um retorno de estado correto e as saídas esperadas.

Tabela 7 - Teste UC02.

Caso de uso	Condição de entrada	Classes válidas	Classes inválidas
UC02 - Upload de artefatos	Inserir informações do artefato para upload	1. Inserir informações em formato JSON, de acordo com os requisitos do artefato desejado.	2. Informações fornecidas fora do padrão JSON
			3. Informações incorretas no JSON
			4. Requisição com corpo vazio
	Enviar uma solicitação ao EndPoint do seu respectivo artefato	5. Solicitação deve chegar ao endPoint do back-end e validar os campos.	6. Enviar requisição sem os dados necessários
			7. Falhas internas da aplicação
			8. Não obter retorno da aplicação. adas
		9. Retornar mensagem padrão de sucesso	10. Retornar mensagem inesperada
			11. Não retornar nenhuma mensagem

Fonte: Elaborada pelo autor (2024).

Tabela 8 - Resultado do teste UC02

Condição de entrada	Classe	Resposta
Inserir informações do artefato para upload	1. Inserir informações em formato JSON, de acordo com os requisitos do artefato desejado.	O Sistema aceitou as informações passadas no formato JSON
	2. Informações fornecidas fora do padrão JSON	O sistema rejeitou as informações fora do padrão JSON
	3. Informações incorretas no JSON	O sistema rejeitou o JSON com informações incorretas
	4. Requisição com corpo vazio	O sistema rejeitou a requisição com o corpo vazio
Enviar uma solicitação ao EndPoint do seu respectivo artefato	5. Solicitação deve chegar ao endPoint do back-end e validar os campos.	O sistema valida o JSON e dá continuidade na requisição
	6. Enviar requisição sem os dados necessários	O sistema rejeitou a requisição
	7. Falhas internas da aplicação	O sistema retorna mensagem de erro padrão para falhas internas inesperadas
	8. Não obter retorno da aplicação	O sistema não retornar informações esperadas para a requisição
	9. Retornar mensagem padrão de sucesso	O sistema retornou a mensagem padrão de

		sucesso
	10. Retornar mensagem inesperada	O sistema retornou uma mensagem não mapeada
	11. Não retornar nenhuma mensagem	O sistema não retornou nenhuma mensagem

Fonte: Elaborada pelo autor (2024).

4.3.3 Gerenciamento de artefatos

No presente teste, o caso de uso UC03, relacionado à validação de ações de gerenciamento de artefatos, foi avaliado. Durante o processo, todas as informações necessárias foram preenchidas para verificar seu funcionamento, resultando em um retorno de estado correto e as saídas esperadas.

Tabela 9 - Teste UC03

Caso de uso	Condição de entrada	Classes válidas	Classes inválidas
UC03 - Gerenciamento de artefatos	Inserir informações do artefato para gerenciamento	1. Inserir o identificador do artefato a ser gerenciado	2. Informações fornecidas não são do tipo INT
			3. Informações de identificação incorretas
			4.Identificação vazia
	Enviar uma solicitação ao EndPoint do seu respectivo artefato	5. Solicitação deve chegar ao endPoint do back-end e validar os	6. Enviar requisição sem os dados necessários

		campos.	7. Falhas internas da aplicação
			8. Não obter retorno da aplicação
		9. Retornar mensagem padrão de sucesso	10. Retornar mensagem inesperada
			11. Não retornar nenhuma mensagem

Fonte: Elaborada pelo autor (2024).

Tabela 10 - Resultado do teste UC03

Condição de entrada	Classe	Resposta
Inserir informações do artefato para gerenciamento	1. Inserir o identificador do artefato a ser gerenciado	O sistema aceitou o identificador
	2. Informações fornecidas não são do tipo INT	O sistema rejeitou o identificador
	3. Informações de identificação incorretas	O sistema rejeitou a identificação incorreta
	4.Identificação vazia	O sistema rejeitou a identificação vazia
Enviar uma solicitação ao EndPoint do seu respectivo artefato	5. Solicitação deve chegar ao endPoint do back-end e validar os campos.	O sistema valida a identificação e dá continuidade na requisição.

	6. Enviar requisição sem os dados necessários	O sistema rejeitou a requisição
	7. Falhas internas da aplicação	O sistema retorna mensagem de erro padrão para falhas internas inesperadas
	8. Não obter retorno da aplicação	O sistema não retornar informações esperadas para a requisição
	9. Retornar mensagem padrão de sucesso	O sistema retornou a mensagem padrão de sucesso
	10. Retornar mensagem inesperada	O sistema retornou uma mensagem não mapeada
	11. Não retornar nenhuma mensagem	O sistema não retornou nenhuma mensagem

Fonte: Elaborada pelo autor (2024).

4.3.4 Integração de Software

No presente teste, o caso de uso UC04, relacionado à validação de ações de integração de software, foram avaliadas. Durante o processo, todas as informações necessárias foram preenchidas para verificar seu funcionamento, resultando em um retorno de estado correto e as saídas esperadas.

Tabela 11 - Teste UC04.

Caso de uso	Condição de entrada	Classes válidas	Classes inválidas
UC04 - Integração de Software	Inserir informações necessárias para realizar interação com a aplicação	1. Inserir informações necessárias para interação com a	2. Informações fornecidas fora do padrão para a ação escolhida.

		aplicação	3. Informações incorretas.
			4. Requisição com corpo vazio
	Enviar uma solicitação ao EndPoint da funcionalidade escolhida.	5. Solicitação deve chegar ao endPoint do back-end e validar os campos	6. Enviar requisição sem os dados necessários
			7. Falhas internas da aplicação
			8. Não obter retorno da aplicação
		9. Retorno padrão de acordo com a funcionalidade	10. Retorno fora do padrão esperado
			11. Não retornar nada

Fonte: Elaborada pelo autor (2024).

Tabela 12 - Resultado do teste UC04

Condição	Classe	Resposta
inserir informações necessárias para realizar interação com a aplicação	1. Inserir informações necessárias para interação com a aplicação	O sistema aceitou as informações

	2. Informações fornecidas fora do padrão para a ação escolhida	O sistema rejeitou as informações
	3. Informações incorretas	O sistema rejeitou as informações incorretas
	4. Requisição com corpo vazio	O sistema rejeitou a requisição de corpo vazio
Enviar uma solicitação ao EndPoint da funcionalidade escolhida.	5. Solicitação deve chegar ao endPoint do back-end e validar os campos	O sistema valida as informações e dá continuidade na requisição
	6. Enviar requisição sem os dados necessários	O sistema rejeitou a requisição
	7. Falhas internas da aplicação	O sistema retorna mensagem de erro padrão para falhas internas inesperadas
	8. Não obter retorno da aplicação	O sistema não retornar informações esperadas para a requisição
	9. Retorno padrão de acordo com a funcionalidade	O sistema retornou a mensagem padrão
	10. Retorno fora do padrão esperado	O sistema retornou uma mensagem não mapeada
	11. Não retornar nada	O sistema não retornou nenhuma mensagem ou objeto

Fonte: Elaborada pelo autor (2024).

4.4 Resultados

Os testes de caixa preta realizados nos cenários mencionados anteriormente atenderam às expectativas, funcionando adequadamente em todas as fases

executadas, uma vez que os cenários foram selecionados entre os mais importantes na plataforma. O sistema apresentou um desempenho adequado, sem falhas que impeçam sua utilização, demonstrando eficácia ao passar pelos testes de caixa preta.

4.4.1 Resultado do Teste UC01: Autenticação

No teste UC01, que abordou o processo de autenticação, o sistema demonstrou conformidade ao aceitar nomes de usuário e senhas no formato correto de texto e rejeitar entradas inválidas, como nomes de usuário ou senhas vazias. Ao enviar as solicitações ao *EndPoint* de autenticação, o sistema validou adequadamente os campos e deu continuidade às requisições válidas. Erros internos inesperados foram tratados com mensagens de erro padrão, e o sistema retornou credenciais corretas para usuários autenticados. No entanto, em alguns casos, o sistema retornou dados inválidos ou nenhuma mensagem, apontando a necessidade de uma maior robustez no tratamento de exceções e retorno de respostas.

4.4.2 Resultado do Teste UC02: Upload de Artefatos

O teste UC02, relacionado ao upload de artefatos, também foi bem-sucedido. O sistema aceitou as informações em formato JSON quando estavam de acordo com os requisitos, rejeitando dados fora do padrão ou com erros. Ao enviar as solicitações ao *EndPoint* correspondente, o sistema validou os campos de forma adequada, permitindo a continuidade da requisição em casos válidos. Falhas internas e erros inesperados resultaram em mensagens de erro padrão, garantindo que o sistema lidou corretamente com essas situações. Contudo, foi observado que em alguns casos, como retorno inesperado ou falta de resposta, o sistema precisaria de aprimoramentos para fornecer feedback mais consistente ao usuário.

4.4.3 Resultado do Teste UC03: Gerenciamento de Artefatos

No teste UC03, que trata do gerenciamento de artefatos, o sistema foi testado com diversas condições de entrada, como a inserção de identificadores válidos, inválidos ou vazios. O sistema aceitou corretamente os identificadores válidos e rejeitou as entradas inválidas, como identificadores incorretos ou não numéricos. As

solicitações enviadas ao *EndPoint* do artefato foram validadas de maneira apropriada, e o sistema rejeitou requisições com dados incompletos ou ausentes. O desempenho foi adequado, com o sistema respondendo corretamente às falhas internas e retornando mensagens padronizadas de sucesso. No entanto, em casos de mensagens inesperadas ou ausência de resposta, foi identificado que melhorias poderiam ser feitas para aprimorar o retorno ao usuário.

4.4.4 Resultado do Teste UC04: Integração de Software

No teste UC04, relacionado à integração de software, o sistema demonstrou conformidade ao aceitar informações necessárias para realizar interações com a aplicação, rejeitando entradas incorretas, fora do padrão ou com corpo vazio. O sistema validou os dados ao serem enviados ao *EndPoint* da funcionalidade e deu continuidade a requisições válidas, demonstrando que as integrações estão funcionando corretamente. O sistema também foi capaz de rejeitar requisições com dados insuficientes e tratou falhas internas de maneira adequada, fornecendo mensagens de erro padronizadas. Tal como nos outros testes, foi identificado que o sistema pode ser aprimorado em situações onde retornou mensagens não mapeadas ou nenhuma resposta, a fim de garantir maior clareza e feedback ao usuário.

5 CONCLUSÃO

Este estudo apresenta uma plataforma cujas principais funcionalidades incluem o gerenciamento, integração e armazenamento de artefatos 3D, abordando questões como o gerenciamento manual de artefatos, a centralização e segurança no armazenamento e a integração com outros softwares. A concepção desta proposta decorre da identificação de necessidades no âmbito da computação gráfica, aliada às potencialidades da área de sistemas gerenciais, visando uma aplicação capaz de administrar, armazenar e disponibilizar recursos para a construção de ambientes 3D.

O propósito delineado neste projeto foi concretizado por meio da implementação de uma solução computacional facilmente integrável, destinada a auxiliar na gestão de elementos relacionados à criação de ambientes virtuais tridimensionais. A solução desenvolvida permite uma administração mais eficiente e automatizada dos recursos envolvidos no processo, minimizando os impactos decorrentes dos desafios identificados. A solução capacitará os usuários responsáveis pela administração de elementos a otimizar seus procedimentos de trabalho, reduzir custos e evitar possíveis retrabalhos, contribuindo para aprimorar a qualidade do produto final e aumentar a satisfação dos usuários.

Além disso, a aplicação incorpora recursos de segurança e autenticação para o controle de acesso e proteção dos usuários, garantindo que apenas indivíduos autorizados possam acessar a aplicação e utilizar suas funcionalidades. A documentação fornecida é intuitiva, facilitando as possíveis integrações e a utilização da plataforma.

Durante o processo de desenvolvimento, o foco foi criar uma ferramenta de alto desempenho. Após a realização dos testes de caixa preta, ficou claro que a ferramenta possui a robustez necessária para ser implementada em ambientes de produção reais. Com os testes da ferramenta demonstrando eficiência, torna-se viável oferecer aos usuários uma solução que facilita a gestão de artefatos em ambientes virtuais 3D complexos de forma mais ágil e eficaz.

5.1 Trabalhos Futuros

Uma das principais áreas de desenvolvimento futuro consistirá na elaboração de uma interface gráfica dedicada para a aplicação, a fim de ampliar a acessibilidade

dos usuários, eliminando a dependência exclusiva de chamadas HTTP via API. Esta medida visa proporcionar uma experiência mais intuitiva e acessível aos usuários, permitindo uma interação mais direta com as funcionalidades da aplicação. A implementação de uma interface gráfica eficiente não apenas simplificará o uso da aplicação, mas também poderá promover uma maior adoção por parte de um público mais amplo, incluindo usuários menos familiarizados com tecnologias de integração baseadas em API. Ademais, a introdução de uma interface gráfica sólida e intuitiva poderá contribuir significativamente para a eficiência operacional e a usabilidade geral da aplicação, elevando assim seu valor e utilidade para os usuários finais.

Outra linha potencial de desenvolvimento prospectivo deste estudo envolve a exploração e implementação de integrações entre a aplicação em foco e outras plataformas amplamente adotadas no mercado. Essa iniciativa visaria estabelecer uma gestão integrada e abrangente dos recursos, abarcando desde o gerenciamento inicial dos artefatos até os estágios avançados de construção e modelagem. A concretização dessa integração ofereceria uma visão abrangente e contínua do ciclo de vida dos projetos, possibilitando uma coordenação eficaz e uma otimização dos processos envolvidos. Adicionalmente, a interconexão com outras aplicações consagradas poderia potencializar a funcionalidade e a utilidade da presente aplicação, agregando novas camadas de complexidade e interoperabilidade ao seu conjunto de recursos. Esse aprofundamento na integração entre aplicações não apenas fortaleceria a relevância do aplicativo no setor, mas também poderia estabelecer integrações estratégicas e parcerias com outras entidades do setor, ampliando seu impacto e sua relevância no cenário tecnológico.

Em síntese, este projeto marcou o início do desenvolvimento de uma ferramenta de administração de elementos destinada a ambientes tridimensionais. Ainda há uma longa trajetória a ser percorrida para transformar esta ferramenta em um produto de excelência, capaz de satisfazer as exigências dos usuários e manter sua relevância em um cenário tecnológico em contínua evolução.

REFERÊNCIAS

CROCKFORD, D. **The application of JSON in web services**. 2006. Disponível em: http://www.json.org/json.html. Acesso em: 29 set. 2024.

DOCKER. **O que é Docker?** 2024. Disponível em: https ://www.d.com/recursos//o que -container. Acesso em: 29 set. 2024.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. 2000. Tese (Doutorado) – University of California, Irvine. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm. Acesso em: 29 set. 2024.

FLUENTVALIDATION. **FluentValidation**: Uma popular biblioteca de validação .NET. Disponível em: https ://github.com/Floco/Fluente. Acesso em: 29 set. 2024.

GONÇALVES, JC; SILVA, AF. Arquitetura de uma ferramenta para controle e rastreio de artistas de software. **Revisão por pares**, v. 9, p. 242–256, 2020. Disponível em: https ://peer.org/índice.php/diário/artigo/ver /2145 . Acesso em: 29 set. 2024.

MARCUS, A. **OAuth 2.0:** The definitive guide. O'Reilly Media, 2013.

MICROSOFT. **O que é o .NET?** 2024a. Disponível em: https ://dot.Microsoft.com /pt-br/aprender/dotnet /o que-é-ponto. Acesso em: 29 set. 2024.

MICROSOFT. **O que é o Microsoft SQL Server?** 2024b. Disponível em: https ://learn.Microsoft.com/pt-br/sql /sql-servidor/sql-servidor-202?visualizar=sql-servidor-ver. Acesso em: 29 set. 2024.

PRESSMAN, R. S. **Software engineering**: a practitioner's approach. 8. ed. New York: McGraw-Hill, 2011.

SANTOS, JP. Utilização dos artistas de contabilidade gerencial como influência nos fatores contingenciais: um estudo multicaso em clínicas médicas. **Revista Gestão e Secretariado**, v. 13, p. 1367-1618, 2018. Disponível em: http://dx.doi.org /10.7769/ges.v13i3.136. Acesso em: 29 set. 2024.

SWAGGER. **O que é Swagger**? ANO. Disponível em: https://swagger.io. Acesso em: 29 set. 2024.