Especificação de Requisitos de Software

Projeto Museu Gengar

Produzido por Adler M. Cachuba

Instituição IFPR

Paranavaí, PR

Data de Inicio: 17/05/2021

Sumário *(gerado automaticamente)*

Histórico de Revisões

Nome	Data	Razões para as mudanças	Versão

1. Introdução

1.1 Objetivo

O objetivo do software é fazer um sistema de gerenciamento de peças de um museu considerando as seções que as peças estão através de um API para ser consumida por dispositivos móveis, levando em consideração dois níveis de acesso para o sistema.

1.2 Convenções do Documento

Os requisitos funcionais serão identificados através do código
iniciado pela sigla “RF-”, seguido da prioridade (ALTA, MÉDIA, BAIXA, OPCIONAL), e os requisitos não-funcionais serão identificados por meio de um código iniciado pela sigla “RNF”.

1.3 Público-alvo e Sugestões de Leitura

Todos aqueles que desejam implementar o sistema, e principalmente para pessoas que tenham interesse no uso dessa ferramenta como meio científico, donos de museu e empreendedores que expõem suas obras, assim como entidades governamentais que disponibilizam a sociedade as entradas nos museus.

1.4 Escopo do Projeto

Todo museu possui obras que estão em determinadas sessões, existe um diretor, um cuidador, e os visitantes, e para tudo estar em ordem quando o visitante chegar existe um sistema por trás disso. O projeto pretende adicionar funcionalidades como o cadastro de sessões e peças, definir se a obra está ou não em exibição, e a transferência de sessão das obras.

1.5 Referências

Como gerir um museu : manual prático / [edição e coordenação Patrick J. Boylan]. -- Brodowski, SP : Associação Cultural de Apoio ao Museu Casa de Portinari ; São Paulo : Secretaria da Cultura do Estado de São Paulo, 2015

1.6 Trabalhos Relacionados

A quem possa interessar a leitura do livro Como gerir um museu: manual prático, realizado com apoio do Museu Casa de Portinari.
https://www.sisemsp.org.br/wp-content/uploads/2013/12/Como%20Gerir%20um%20Museu.pdf

2. Descrição Geral

2.1 Perspectiva do Produto

A ideia do sistema é construir uma API em Laravel, é uma ideia de sistema única e independente, para ser utilizada em museus, sem fins financeiros, para o gerenciamento das obras de artes que ficam disponíveis em diversos lugares, como universidades, ou museus do próprio governo.

2.2 Funcionalidades do Produto

As funcionalidades serão as seguintes, como descreve o diagrama de caso de uso na Figura 1:

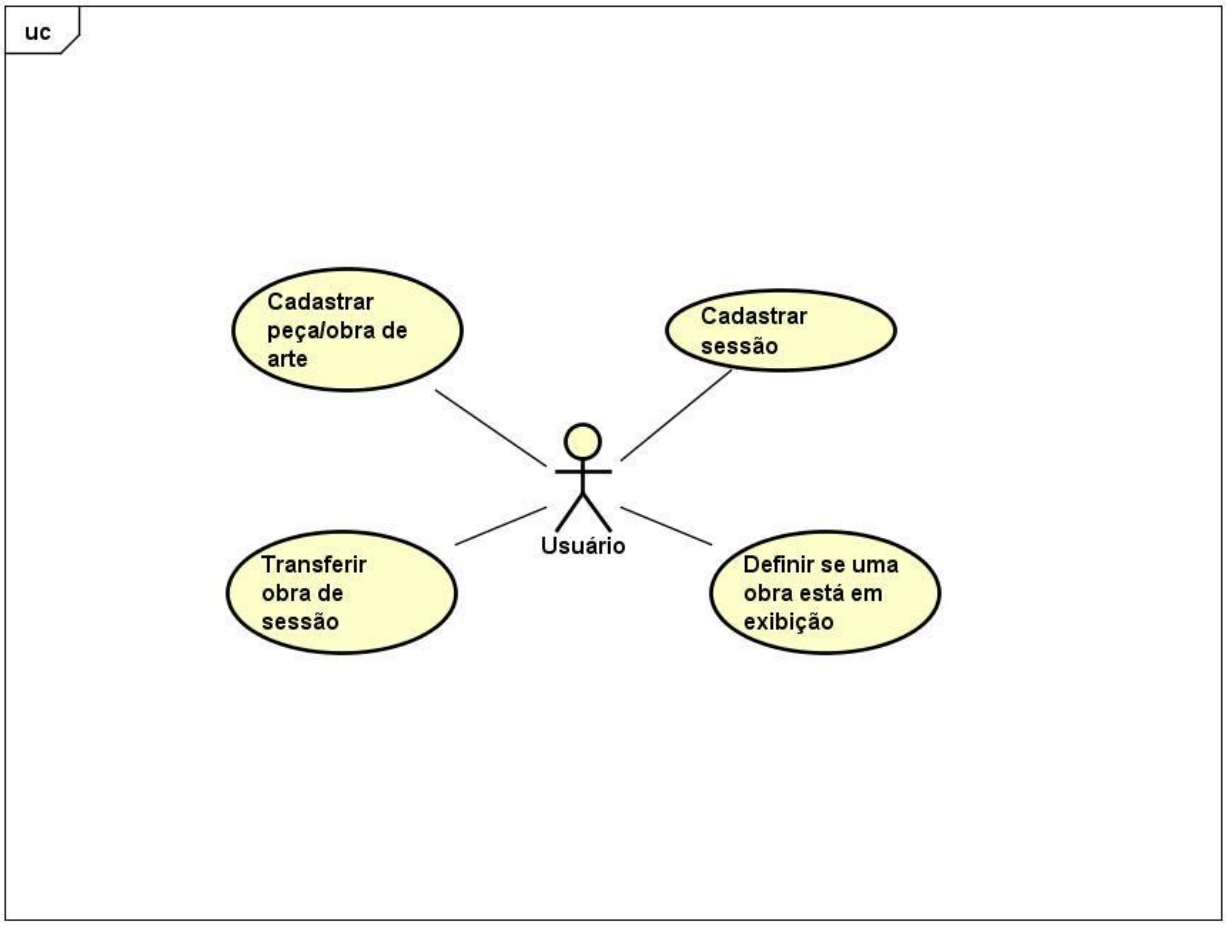


Figura 1. Diagrama de caso de uso.

1. Cadastro de peças

Aqui é realizado o cadastro das peças, no qual é considerado o nome, se ele é ativo ou não e a quantidade.

2. Cadastro de sessão

Toda peça é uma obra de arte, e é exposta em uma sessão, cada sessão tem um nome apenas.

3. Definir se está ou não em exibição

Assim como dito, tem um atributo nas peças que diz se ele está ativo ou não, e é possível realizar essa modificação.

4. Transferência de sessão para as peças.

As peças podem ser transferidas de sessão, mas somente se estiverem em exibição.

2.3 Perfis e Características dos Usuários

Os perfis de usuários são donos de museus, e aqueles que precisam de um sistema para controlar as peças que estão em exibição, como galerias de artes, quadros, entre outros.

2.4 Ambiente Operacional

A API será desenvolvida em Laravel, e posteriormente será disponibilizada no site heroku (https://www.heroku.com/) que hospeda gratuitamente sua API. Para o banco de dados, será o MySql.

O mais correto para evitar travamentos e lentidão é instalar o banco de dados no mesmo servidor onde está o servidor de aplicação, evitando gastos desnecessários.

- Prós: Fácil acesso, conexão rápida com front end;
- Contras: Mais uma função para o "servidor" realizar, o que demanda um planejamento melhor do orçamento visando uma máquina melhor.

Se necessário uma página web futuramente, praticamente todos os provedores possuem o mesmo custo da hospedagem, no valor de R$ 9.99 mensal, porém tem servidores melhores (máquina virtual melhor, processador, memória, link de internet, rota de Link), e outros com máquinas padrões (Processador single core - 1gb ram - link 50mb ou inferior).

- Prós: Somente se escolher um servidor com boa reputação (Visando Configuração);
- Contra: Depende da viabilidade financeira, pois se o investimento for baixo, o equipamento será o padrão.

Custos de hospedagem: Custo aproximadamente R$ 9,99 mensal online.

- Prós: Acessibilidade em qualquer lugar, somente acesso via internet;
- Contra: Custo mensal ou anual, para deixar site online.

Custo com domínio anual: Custo de 99,98 por ano, no registro BR ou .com

- Prós: Pagamento 1 vez ao ano;
- Contra: Não realizando pagamento, ou não opção do registro em domínios pagos, somente será possível em domínios "diferentes", no caso os gratuitos, .tw .te.ti, etc.

2.5 Restrições de Projeto e Implementação

As restrições deste projeto estão relacionadas ao tamanho das obras que serão apresentadas, pois não é levado em consideração os critérios que serão necessários para poder apresentar, como tamanho da área de restrição, tamanho da obra de arte e se é necessário algum equipamento a mais para aquela obra não ser depreciada.

2.6 Documentação do Usuário

Para a documentação do usuário, o Swagger terá as seguintes funcionalidades:

- Cadastro de peças.
- Definir se está ou não em exibição.
- Transferir de sessão.

Porém o Swagger não está completo, então o cadastro de sessão não está disponível no Swagger, mas está funcionando perfeitamente via postman, assim como a alteração e deleção de sessões.

2.7 Pressupostos e Dependências

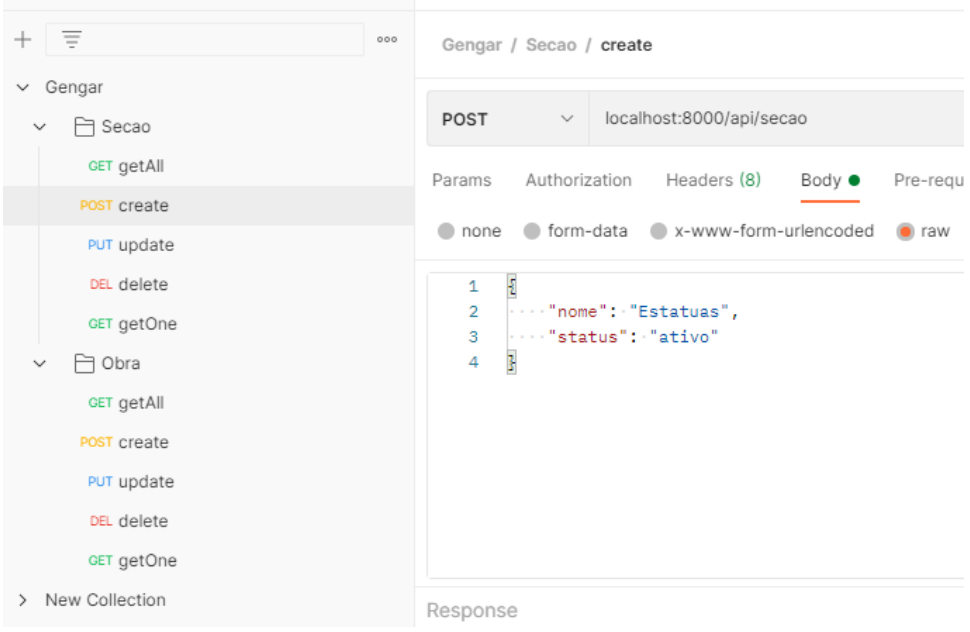
1. É pressuposto que todas as peças que forem cadastradas precisam pertencer a uma seção, e que toda peça é uma obra de arte.
2. O sistema é dependente do banco de dados estar operando perfeitamente sem nenhum erro.
3. O servidor precisa ser compatível com a quantidade de acessos e requisições, visando melhor performance.

3. Funcionalidades do Sistema

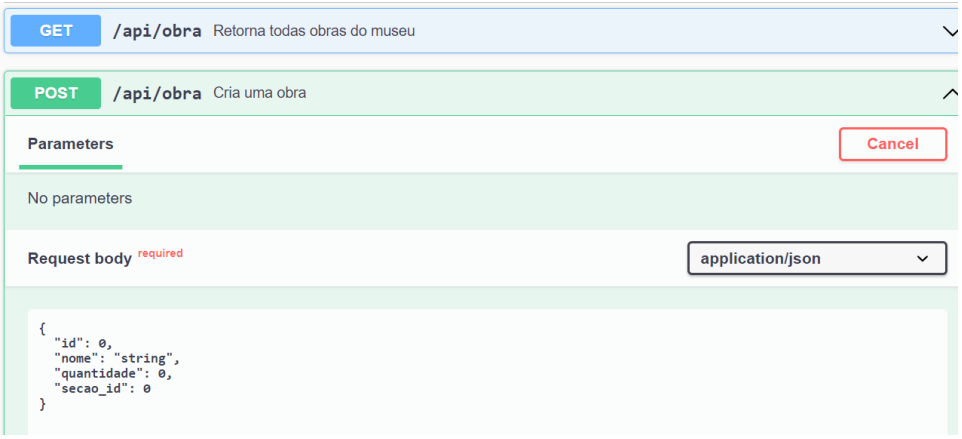
A seguir será especificado as funcionalidades que o sistema irá atender.

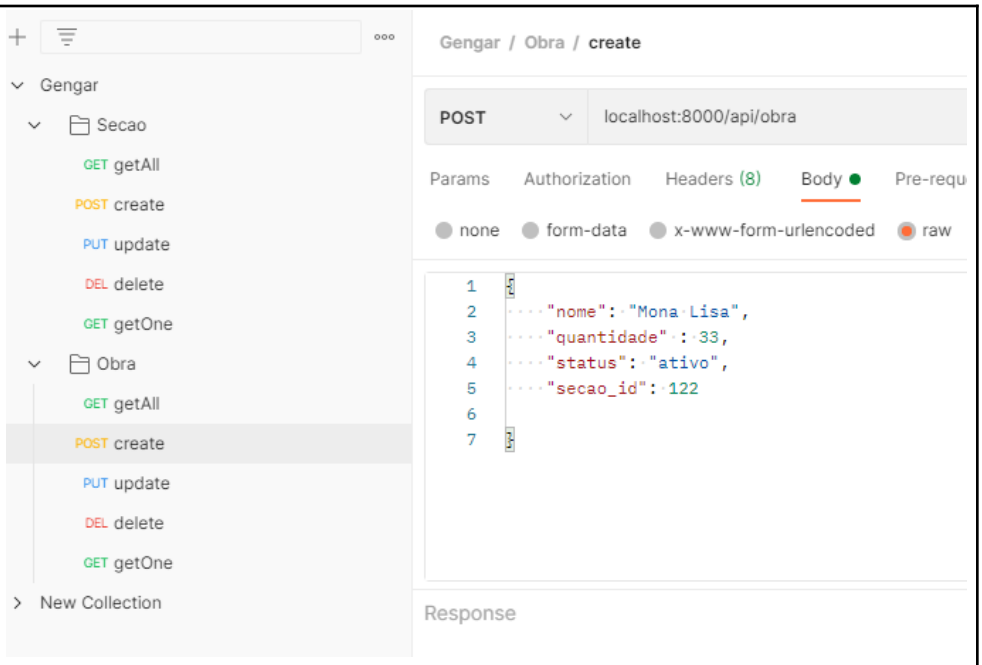
3.1 Cadastro de sessão

Pré-Condição:	Existir um espaço na exposição para sessão, postman instalado, e projeto rodando.
Garantia de sucesso:	Sessão criada no banco de dados, retornando o valor 400.

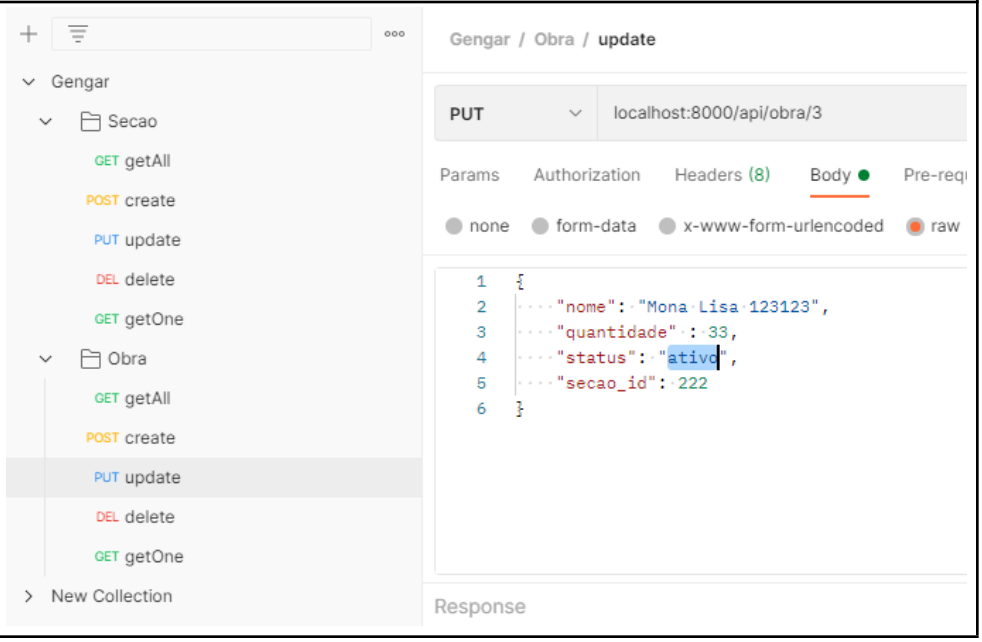
Cenário principal simplificado:	 Pelo Postman.
Cenário alternativo:	A sessão não é criada porque aconteceu um erro, e mostra o código do erro.

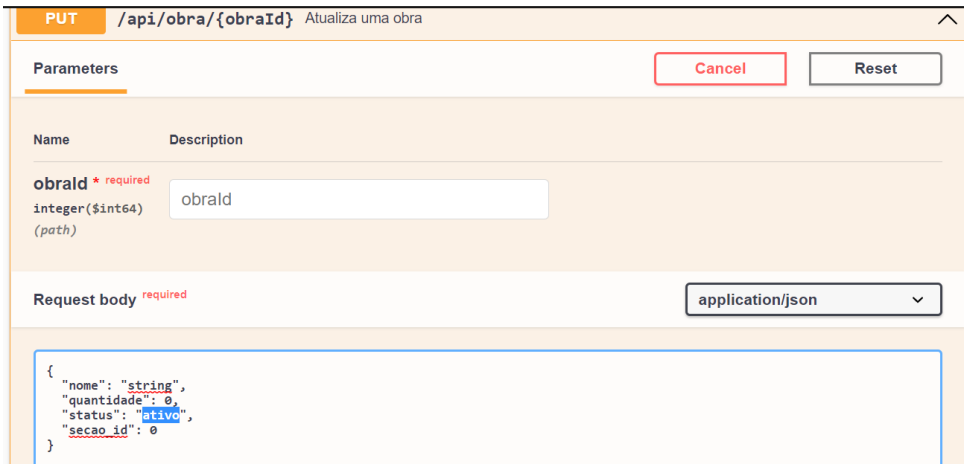
3.2 Cadastro de obra

Pré-Condição:	Existir um espaço na sessão para criar e sistema rodando.
Garantia de sucesso:	Obra criada no banco de dados, retornando o valor 400.
Cenário principal simplificado:	 Pelo swagger

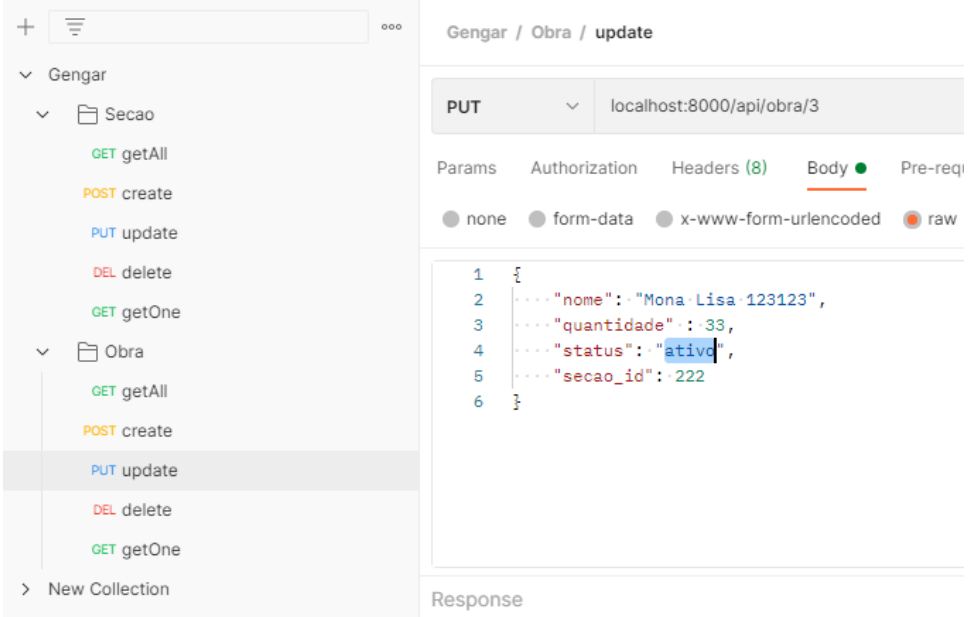
	 Pelo Postman.
Cenário alternativo:	A sessão não é criada porque já existe um nome dessa sessão no banco e o sistema exibe uma mensagem de erro.

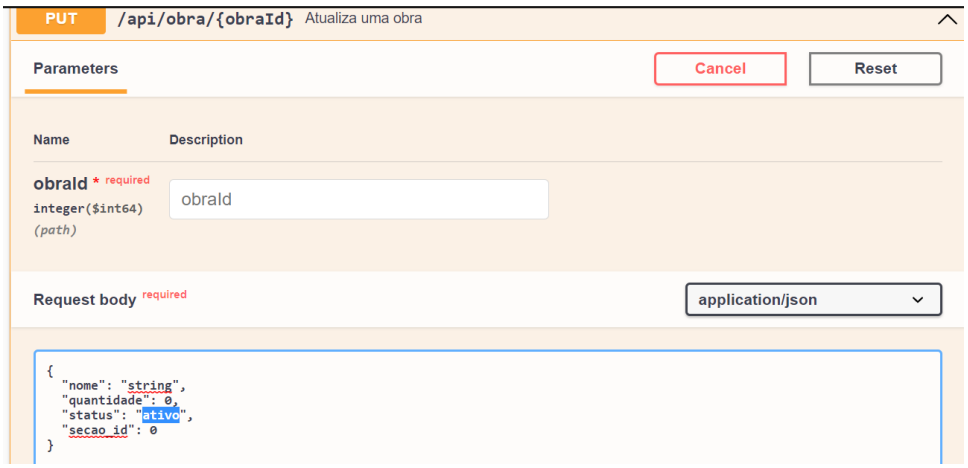
3.3 Definir se uma obra está ou não em exposição

Pré-Condição:	Existir uma obra cadastrada no banco.
Garantia de sucesso:	Obra muda o status da obra no banco de dados retornando 400.
Cenário principal simplificado:	

	Pelo Postman, informando o ID da obra na URL.  Pelo Swagger.
Cenário alternativo:	Voltar erro 500, erro de servidor, e exibe para o usuário a mensagem de erro.

3.4 Transferir de seção

Pré-Condição:	Existir uma obra cadastrada no banco.
Garantia de sucesso:	Obra muda a seção que está cadastrada no banco de dados
Cenário principal simplificado:	

	Pelo Postman, informando o ID da obra na URL.  Pelo Swagger.
Cenário alternativo:	Retorna o código de erro, pode ser por sessão inválida, ou erros no JSON.

4. Requisitos de Interface Externa

4.1 Interfaces do Usuário

O sistema terá uma interface através de um swagger, a interface irá conter opções de cadastrar, alterar e visualizar as peças e sessões.

A tela deve mostrar ao usuário as informações para poder criar, editar, deletar as obras de arte, através da inserção de JSON, com o nome da peça, qual seção ela está, e se ela está disponível ou não.

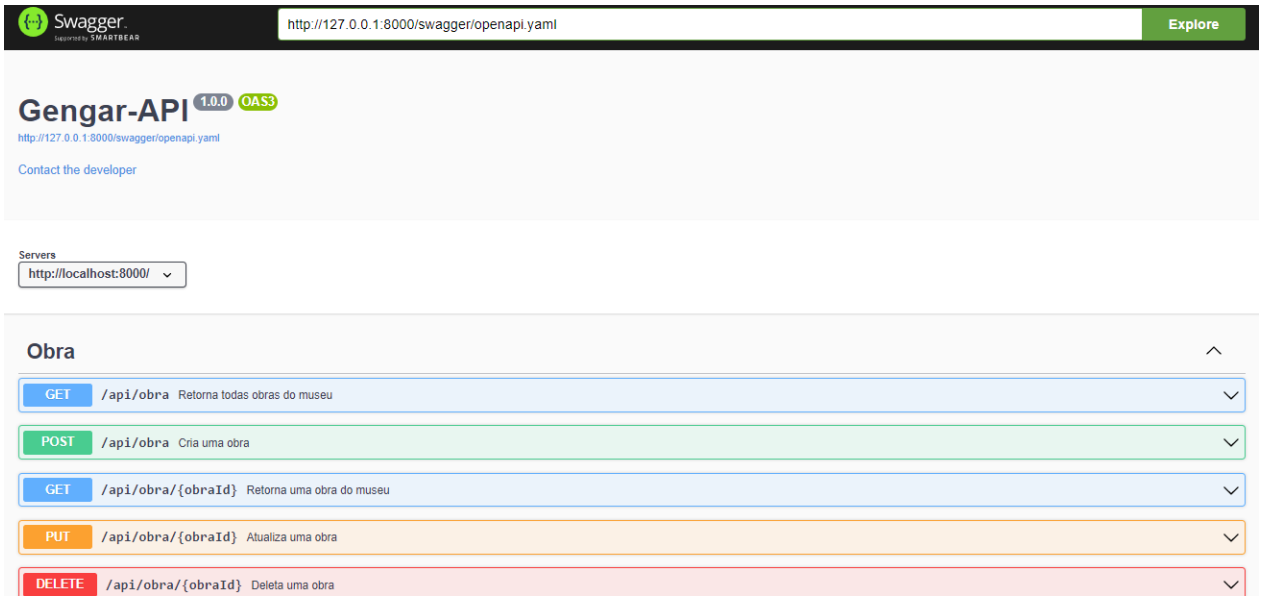


Foto 2. Foto do Swagger.

4.2 Interfaces de Hardware

Não se aplica.

4.3 Interfaces de Software

A API Rest pode ser consumida tanto pelo swagger, que é a implementação web, e também pode ser feita pelo Postman.

4.4 Interfaces de Comunicação

O servidor estará ligado a internet, o protocolo de comunicação entre a API e o software será o HTTP. Qualquer dispositivo com acesso a navegadores terá como acessar e consumir essa API pelo Swagger, ou utilizando o Postman para fazer as requisições.

5. Outros Requisitos Não-Funcionais

5.1 Requisitos de Desempenho

RNF1001 - O tempo para realizar a consulta de algum usuário não deverá passar de 30 segundos.

RNF1002 - O sistema precisa no mínimo possuir um processador Dual Core 2Ghz, 4GB RAM e 40GB disponível em disco.

5.2 Requisitos de Proteção

RNF2001 - O sistema deve validar todos os dados antes de serem inseridos no banco de dados.
RNF2002 - O sistema terá um controle e acesso restringido aos dados.
RNF2003 - O sistema poderá ter a instalação de certificados digitais.
RNF2004 - O sistema irá possuir senhas “fortes” de acesso aos perfis de administradores.
RNF2005 - O sistema deve mostrar mensagens de erro caso operações realizadas não sejam permitidas.

5.3 Requisitos de Segurança

Para os requisitos de segurança será considerado os seguintes tópicos:

- A senha dos usuários deve possuir no mínimo 8 dígitos, é necessário uma letra maiúscula e um número
- O servidor hospedado fornece um firewall garantindo a autenticidade dos dados, também fornecendo um backup das últimas 24 horas anteriores.
- Todos os dados fornecidos pelos usuários serão usados apenas pela empresa para solicitações de suporte, garantindo que nenhuma informação poderá ser acessada por outra entidade.

5.4 Atributos de Qualidade de Software

RNF4001 - O sistema não irá possuir um modo que funcione offline, em todas suas funcionalidades será necessária a utilização da internet 24/7.
RNF4002 - Um usuário deverá conseguir realizar o cadastro de uma obra em menos de 5 minutos.

6. Outros Requisitos

Apêndice A: Glossário

API(*Application Programming Interface*) é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na web.

Laravel é um framework PHP livre e open-source que facilita a criação dos componentes e da criação de uma API.

Apêndice B: Modelo de Análise

<Opcionalmente, inclua quaisquer modelos de análise pertinentes, como diagramas de fluxo de dados, diagramas de classes, diagramas de transição de estado ou diagramas de relacionamento de entidade.>

Apêndice C: Lista de Problemas

<Esta é uma lista dinâmica dos problemas de requisitos em aberto que ainda precisam ser resolvidos, incluindo "ASDs", decisões pendentes, informações necessárias, conflitos aguardando resolução e similares.>