

Anderson Silva Fonseca

Matrícula: 2012378

## **Ferramenta de Visualização e Marcação de Imagens 360º em Ambientes 3D**

Rio de Janeiro - RJ

Julho - 2021

# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>2</b>
<b>1.1</b>	<b>Motivação</b>	<b>3</b>
<b>1.2</b>	<b>Objetivo</b>	<b>3</b>
<b>1.2.1</b>	<b>Objetivos Específicos</b>	<b>3</b>
<b>1.2.2</b>	<b>Organização do trabalho</b>	<b>3</b>
<b>2</b>	<b>FUNDAMENTAÇÃO TEÓRICA</b>	<b>4</b>
<b>2.1</b>	<b>Imagens Equiretangulares</b>	<b>4</b>
<b>2.2</b>	<b>Modelos Digitais 3D</b>	<b>5</b>
<b>2.3</b>	<b>Rasterização</b>	<b>6</b>
<b>2.4</b>	<b>WebGL</b>	<b>7</b>
<b>2.5</b>	<b>Three.js</b>	<b>7</b>
<b>3</b>	<b>REQUISITOS E ESPECIFICAÇÕES</b>	<b>9</b>
<b>3.1</b>	<b>Requisitos funcionais e Não funcionais</b>	<b>9</b>
<b>3.1.1</b>	<b>Requisitos funcionais</b>	<b>9</b>
<b>3.1.2</b>	<b>Requisitos não funcionais</b>	<b>11</b>
<b>3.2</b>	<b>Casos de Uso</b>	<b>11</b>
<b>4</b>	<b>DESENVOLVIMENTO</b>	<b>16</b>
<b>4.1</b>	<b>Plataformas e tecnologias</b>	<b>16</b>
<b>4.2</b>	<b>Arquitetura</b>	<b>16</b>
<b>4.3</b>	<b>Diagramas de Classes</b>	<b>17</b>
<b>4.4</b>	<b>Testes</b>	<b>20</b>
<b>5</b>	<b>MANUAL DO USUÁRIO</b>	<b>23</b>
<b>5.1</b>	<b>Requisitos necessários para hospedagem</b>	<b>23</b>
<b>5.2</b>	<b>Requisitos mínimos</b>	<b>23</b>
<b>5.3</b>	<b>Apresentação do Sistema</b>	<b>23</b>
<b>5.4</b>	<b>Câmeras e Imagens 360º</b>	<b>24</b>
<b>5.5</b>	<b>Modelos</b>	<b>25</b>
<b>5.6</b>	<b>Navegação</b>	<b>27</b>
	<b>REFERÊNCIAS</b>	<b>29</b>

# 1 Introdução

O desgaste das instalações industriais é um problema comum que podem ocorrer em qualquer tipo de estrutura com o passar do tempo. O problema fica ainda mais sério quando essas estruturas se encontram em lugares de difícil acesso ou com alto nível de periculosidade, por exemplo embarcações e plataformas marítimas. Fatores como corrosão, afrouxamento de parafusos, trincas, rompimentos, etc, ou a combinação destes fatores podem comprometer uma estrutura permanentemente.

Avaliar o quanto comprometido uma estrutura possa estar é uma atividade custosa, muitas das vezes é necessário um ou mais especialistas experientes; equipamentos especializados; custos de logística e transporte e a própria segurança dos funcionários envolvidos. Estas e outras dificuldades podem acarretar em uma avaliação lenta e cansativa. Estruturas que já apresentam um alto grau de risco podem se agravar ainda mais durante este período de tempo.

Para auxiliar os especialistas a avaliarem o problema de uma estrutura são usados como auxílio: representações 3D; fotografias antigas e recentes; fotografias equiretangulares ( $360^\circ$ ); filmagens do local usando drones, etc. Porém, muitas das vezes, estes recursos são usados de maneira desassociada com os outros recursos. Por exemplo, fotografias não possuem uma referência de sua localização dentro de uma representação 3D. Isto por sua vez, dificulta ainda mais o trabalho de avaliação.

Diante da situação apresentada, este trabalho propõe o desenvolvimento de uma aplicação que venha facilitar a realização de avaliações eficiente em estruturas.

## 1.1 Motivação

Para auxiliar uma avaliação eficiente, que diminua os riscos, custos e o tempo dos especialistas são necessários o uso de tecnologias capazes de mensurar e visualizar o nível de deterioração de uma estrutura. Projetos desse gênero são desafiadores, envolvendo redes neurais, processamento de imagem, sistemas de visualização e marcação, entre outras.

## 1.2 Objetivo

Desenvolvimento de uma aplicação computacional que possibilite a visualização de uma fotografia equiretangular e um modelo digital 3D de uma estrutura marítima simultaneamente. Permitindo que seus usuários consigam estimar a posição exata da fotografia e a verdadeira grandeza de uma possível patologia dentro de uma embarcação.

### 1.2.1 Objetivos Específicos

- Leitura e visualização de imagens equiretangulares;
- Leitura e visualização de modelos 3D;
- Criar uma aplicação que consiga abrir modelos e imagens hospedados em um servidor externo.
- Criação de um cenário com múltiplas imagens associadas a um modelo 3D;
- Garantir ao usuário meios de interação com o cenário.
- Construir um ambiente Web acessível para os usuários necessário para consumir a aplicação.

### 1.2.2 Organização do trabalho

Este trabalho está organizado da seguinte maneira:

No capítulo Fundamentação teórica são apresentados os conceitos necessários para a compreensão deste trabalho, tais como breves explicações das tecnologias utilizadas e dos algoritmos implementados. No capítulo Requisitos e Especificações são apresentados as modelagens do sistema, a quais foram durante todo o processo de desenvolvimento. No capítulo Desenvolvimento apresenta-se a arquitetura e tecnologias do sistema, como o sistema é organizado e como está estruturado. Também são realizados testes para validar o funcionamento da aplicação. Por fim, no capítulo Manual do usuário, apresenta-se uma breve introdução dos sistema e como usar as funcionalidades disponíveis. Contém informações de instalação e requisitos necessários.

## 2 Fundamentação Teórica

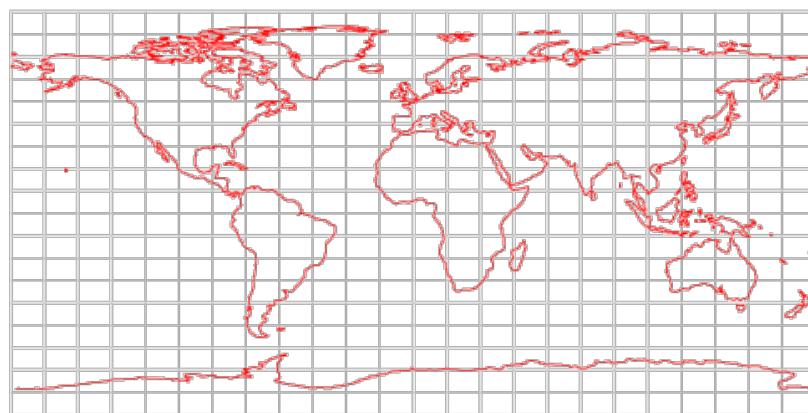
Neste capítulo introduzimos os conceitos fundamentais para a contextualização do trabalho. Cada seção é descrita de maneira breve, com a finalidade de introduzir ou relembrar conceitos conhecidos.

### 2.1 Imagens Equiretangulares

Durante o uso da aplicação serão necessárias o uso de imagens equiretangulares para visualização do cenário. Imagens equiretangulares, também conhecidas como projeções equiretangulares ou panoramas 360º são imagens que conseguem capturar uma fotografia do ambiente em sua volta. Estes panoramas são bastante utilizados em aplicações de Realidade Virtual ou Aumentada, pois conseguem levar para o usuário a sensação de imersão de estar dentro de uma fotografia ou de um vídeo. ([EIRIS; GHEISARI; ESMAEILI, 2018](#))

Baseando-se em coordenadas geográficas, uma projeção equirectangular mapeia as linhas verticais dos meridianos em linhas retas verticais em um espaço plano e as linhas de latitude em linhas horizontais do espaço plano. ([WEISSSTEIN, 2021](#))

Figura 1 – Projeção equirectangular do mundo



Fonte: [Weisstein \(2021\)](#)

Existem diversos usos para imagens equiretangulares, uma das suas utilizações mais conhecidas é seu uso para navegações imersivas que possui um video ou imagem como fundo. Por exemplo, softwares como Google Maps ([GOOGLE, 2021](#)) apresentam uma experiência de navegação entre ruas da cidade usando imagens 360º; um projeto com engenheiros e profissionais de realidade virtual tornaram possível a realização de tour

por imagens 360 que mostram interiores de ruas e de edifícios históricos de uma região associados a um modelos CAD, áudios ou sinalizações 2D. ([CôTé et al., 2013](#))

Figura 2 – Exemplo de imagem equiretangular



Fonte: Próprio Autor

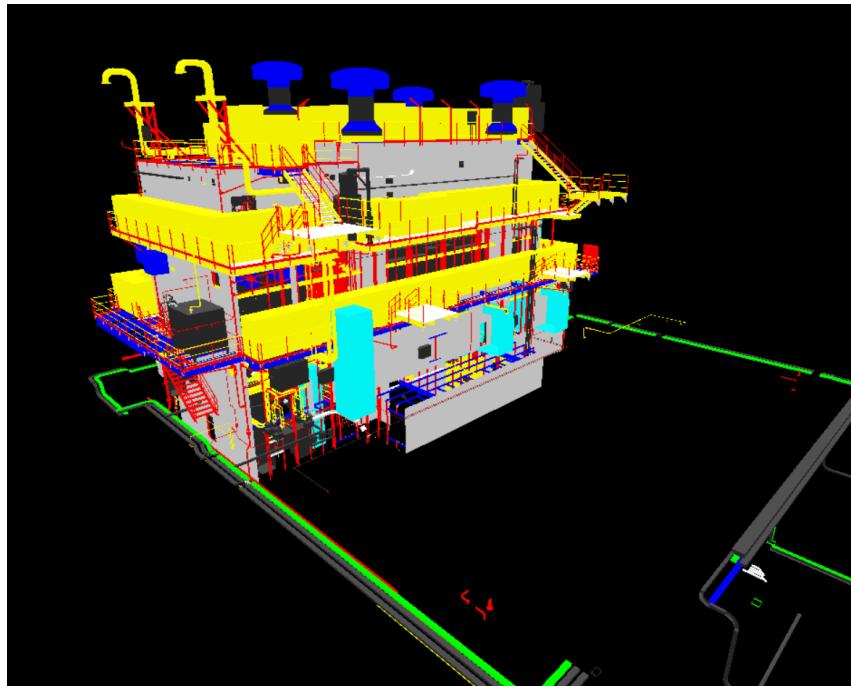
## 2.2 Modelos Digitais 3D

Modelos digitais 3D são representações de um objeto em conjunto de pontos em um espaço tridimensional. Cada ponto pode estar conectado por um conjunto de triângulos, linhas, superfícies curvas, etc.

Esses modelos são capazes de representar estruturas reais ou fictícias. Por exemplo, engenheiros e designers usam softwares de modelagem para construir uma representação 3D de edifícios e móveis; artistas 3D e desenvolvedores de jogos modelam cenários completamente tridimensionais para proporcionar ao usuário uma imersão a um cenário fictício.

Para representar estes modelos, usa-se diversos formatos de arquivos capazes de armazenar informações de uma superfície 3D: STL, OBJ, GLTF, PLY, etc. Normalmente são armazenadas as coordenadas de um objeto, informações de conectividade entre os vértices e muitas vezes informações sobre as normais. Dependendo das informações que são guardadas nesses arquivos, eles podem ter um alto custo para armazenamento. ([ZHANG, 2012](#))

Figura 3 – Modelo 3D de uma estrutura



Fonte: Próprio Autor

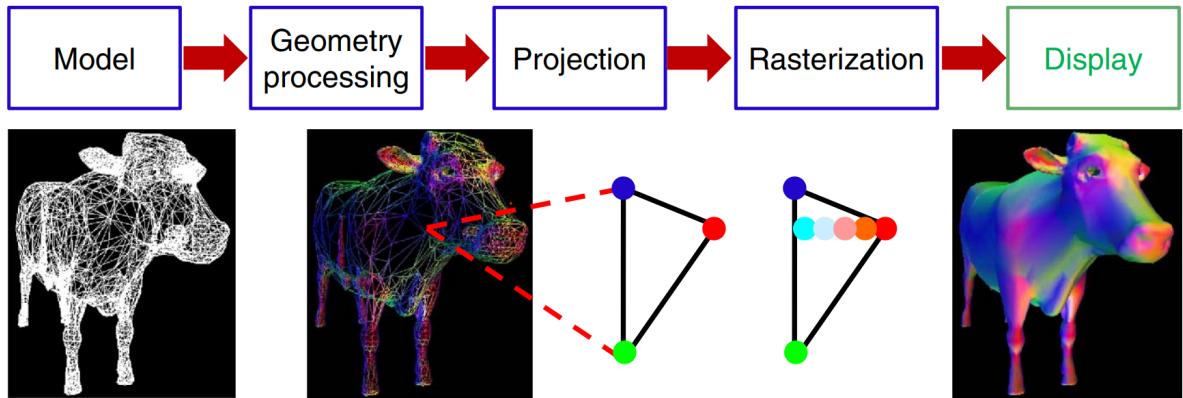
## 2.3 Rasterização

Para visualizar um cenário tridimensional em um ambiente bidimensional usa-se um método conhecido na computação gráfica como renderização. O resultado final da renderização resulta em um *frame* de tamanho predeterminado, onde cada pixel passa por uma série de cálculos para determinar o que seria visível para o usuário em uma determinada posição e iluminação. Por se tratar de uma tarefa custosa, são desenvolvidos algoritmos específicos de renderização e/ou até mesmo usados hardwares dedicados.

Um algoritmo conhecido como *Z-Buffer* ou rasterização de polígonos é um dos mais comuns e usados no processo de renderização. Ele é nativamente implementado em bibliotecas baseadas em OpenGL ([KHRONOS GROUP, 2021](#)).

Neste algoritmo, a superfície de um modelo 3D é tipicamente representada por um conjunto de vértices conectadas por triângulos. Inicialmente estas representações são submetidos a processos de transformações, iluminação, recorte e projeção para que no fim sejam mapeados para um espaço 2D. A partir dos triângulos projetados são calculados os seus fragmentos, pixels com informações de cor, profundidade e textura, que são submetidos a um mapeamento de profundidade. Em outras palavras, os fragmentos que tiverem um valor de profundidade menor são eleitos para compor o *frame* resultante. ([GATTASS, 2013](#))

Figura 4 – Pipeline de Renderização

Fonte: [Zhang \(2012\)](#)

## 2.4 WebGL

WebGL é uma API de renderização 3D projetada para o funcionamento em navegadores Web. Ele é derivado do OpenGL, mas seu contexto foi feito para funcionar em conjunto com o contexto de visualização do HTML Canvas. O recurso HTML Canvas permite o uso de diversas APIs para renderização, por padrão ela usa o *CanvasRenderingContext2D*, um contexto de visualização simples para desenhos 2D no HTML. A API WebGL é implementada a partir de outro contexto chamado *WebGLRenderingContext* disponível dentro do HTML Canvas. As funções da API estão disponíveis para uso usando Javascript.

## 2.5 Three.js

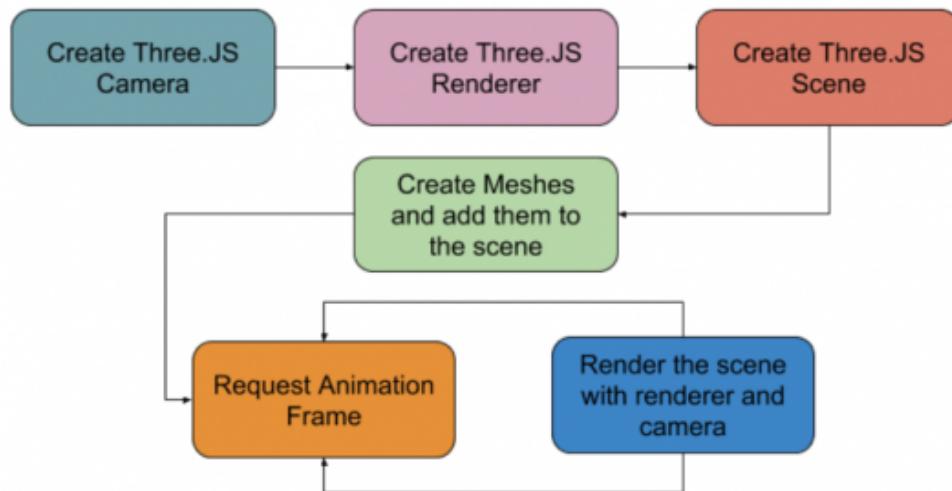
Three.js é uma biblioteca para navegadores escrita em Javascript usada para criação e animação de cenário gráficos 3D. Ela possui compatibilidade com diversas API gráficas disponíveis no HTML Canvas, como WebGL, WebGPU, SVG e CSS3D. ([CABELLO, 2021](#)) Nesta biblioteca é possível encontrar exemplos de criação de câmera, animações, estruturas geométrica, objetos, carregadores de dados, funções algébricas, luzes, câmeras, etc.

O pipeline sugerido para exibição de um ambiente 3D usando a biblioteca pode ser implementado da seguinte forma:

- Criação de uma câmera.
- Criação de um renderizador, por exemplo pode ser usado o WebGL;
- Criação de um cenário, que receberá os objetos e a câmera.

- Criação ou importação de objetos dentro do cenário. No Three.js estes objetos são considerados como malhas;
- Por fim, ela entra em loop de renderização, onde a API escolhida é usada para exibir as geometrias enviadas para o cenário.

Figura 5 – Pipeline de Renderização do Three.js



Fonte: [Intel Open Source \(2019\)](#)

# 3 Requisitos e Especificações

Neste capítulo, apresentamos os requisitos e especificações da aplicação proposta. As modelagens apresentadas nesta seção foram usados como base em todo o processo de desenvolvimento.

## 3.1 Requisitos funcionais e Não funcionais

Os requisitos nesta aplicação são categorizados em relação a prioridade e esforço. Eles apresentam uma breve descrição, uma prioridade essencial, importante ou desejável e um esforço baixo, médio ou alto.

### 3.1.1 Requisitos funcionais

O sistema é constituído dos seguintes requisitos funcionais descritos abaixo:

#### 1. Leitura e carregamento de uma imagem 360º no cenário.

<b>Descrição</b>	O usuário poderá escolher uma imagem 360º disponível e carrega-lo para o cenário atual.
<b>Prioridade</b>	Essencial
<b>Esforço</b>	Alto

#### 2. Leitura e carregamento de um modelo 3D no cenário.

<b>Descrição</b>	O usuário poderá escolher um modelo 3D disponível e carrega-lo para o cenário atual.
<b>Prioridade</b>	Essencial
<b>Esforço</b>	Alto

#### 3. Alteração no posicionamento da câmera

<b>Descrição</b>	O usuário poderá alterar o posicionamento da câmera, com a finalidade de transitar pelo cenário.
<b>Prioridade</b>	Essencial
<b>Esforço</b>	Médio

#### 4. Alteração no posicionamento da textura da Imagem 360º

<b>Descrição</b>	O usuário poderá girar a textura em volta de si, com a finalidade de ajustar o posicionamento da textura.
<b>Prioridade</b>	Essencial
<b>Esforço</b>	Médio

#### 5. Posicionamento de câmera usando teclas e mouse

<b>Descrição</b>	O usuário poderá alterar o posicionamento da câmera, usando controles do teclado para movimentação e mouse para orientação.
<b>Prioridade</b>	Desejável
<b>Esforço</b>	Médio

#### 6. Adição de múltiplas câmeras

<b>Descrição</b>	O usuário poderá adicionar outras câmeras, com outras imagens com textura e posicionamentos diferentes.
<b>Prioridade</b>	Desejável
<b>Esforço</b>	Médio

#### 7. Alternância entre múltiplas câmeras

<b>Descrição</b>	Em cenários com múltiplas câmeras, ele pode alternar sua visualização entre elas.
<b>Prioridade</b>	Desejável
<b>Esforço</b>	Médio

#### 8. Visualização de estatísticas

<b>Descrição</b>	O usuário pode visualizar as informações de status da aplicação, incluindo informações do cenário da atual e informações do esforço computacional
<b>Prioridade</b>	Desejável
<b>Esforço</b>	Baixo

### 3.1.2 Requisitos não funcionais

O sistema é constituído dos seguintes requisitos não funcionais descritos abaixo:

#### 1. Multiplataforma

<b>Descrição</b>	O sistema poderá ser executado em diferentes sistemas operacionais, tais como Linux e Windows, necessitando somente de um browser.
<b>Prioridade</b>	Desejável
<b>Esforço</b>	Médio

#### 2. Robustez

<b>Descrição</b>	O sistema deverá ser estável, ou seja, não deverá travar ou fechar inesperadamente.
<b>Prioridade</b>	Essencial
<b>Esforço</b>	Alto

#### 3. Internet

<b>Descrição</b>	Por se tratar uma aplicação Web é necessário o uso de internet para acessar o sistema.
<b>Prioridade</b>	Essencial
<b>Esforço</b>	Médio

## 3.2 Casos de Uso

Nessa subseção é apresentado o diagrama de casos de uso, exibida na Figura 6. Os casos de uso são baseadas nos requisitos do sistema sua descrição detalhada pode ser nas tabelas abaixo.

#### 1. Escolher Imagem 360°

Descrição	
<b>Ator:</b>	O usuário
<b>Descrição sucinta:</b>	Escolher uma imagem 360° disponível
<b>Pré-condições:</b>	Estar com o sistema aberto
<b>Pós-condições:</b>	Redireciona o usuário para o Caso de Uso "Criar Câmera"
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a página da aplicação;</li> <li>2. O usuário escolhe a opção "Adicionar Câmera";</li> <li>3. O sistema exibe diversas imagens 360° para serem carregadas;</li> <li>4. O usuário escolhe uma das imagens e aperta o botão "Selecionar Imagem";</li> <li>5. O sistema inicia o processo de criação de Câmera (Veja o Caso de Uso "Criar Câmera").</li> </ol>
<b>Cenário alternativo:</b>	Se a conexão com o servidor estiver instável a listagem de imagens não será exibida.

## 2. Criar câmera

Descrição	
<b>Ator:</b>	O usuário
<b>Descrição sucinta:</b>	Criação de uma câmera no cenário
<b>Pré-condições:</b>	Ter escolhido uma imagem. (Veja o Caso de Uso "Escolher Imagem 360°")
<b>Pós-condições:</b>	Exibe a imagem 360° na tela
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O sistema cria uma nova configuração para a câmera tendo a imagem escolhida como textura;</li> <li>2. O sistema exibe a imagem 360° na tela, junto com as informações relacionadas a câmera.</li> </ol>
<b>Cenário alternativo:</b>	Se a conexão com o servidor estiver instável a imagem não será carregada e a câmera não será criada.

## 3. Reposicionar câmera

<b>Descrição</b>	
<b>Autor:</b>	O usuário
<b>Descrição sucinta:</b>	Reposiciona a posição da câmera
<b>Pré-condições:</b>	Ter pelo menos uma câmera criada. (Veja o Caso de Uso "Criar câmera")
<b>Pós-condições:</b>	Altera a posição da câmera a posição no cenário
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O usuário seleciona uma câmera;</li> <li>2. O usuário expande as "Configurações da Câmera Atual";</li> <li>3. O usuário escreve as posições da câmera;</li> <li>4. O sistema altera a posição da câmera no cenário.</li> </ol>
<b>Cenário alternativo:</b>	<ol style="list-style-type: none"> <li>1. O usuário seleciona o cenário;</li> <li>2. O usuário pode utilizar os controles do teclado para atualizar a posição da câmera;</li> <li>3. O sistema altera a posição da câmera no cenário.</li> </ol>

#### 4. Reposicionar textura

<b>Descrição</b>	
<b>Autor:</b>	O usuário
<b>Descrição sucinta:</b>	Reposiciona a imagem 360º dentro da câmera
<b>Pré-condições:</b>	Ter pelo menos uma câmera criada. (Veja o Caso de Uso "Criar câmera")
<b>Pós-condições:</b>	Mostra uma a imagem 360º com uma nova rotação
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O usuário seleciona uma câmera;</li> <li>2. O usuário expande as "Configurações da Câmera Atual";</li> <li>3. O usuário altera a rotação da imagem;</li> <li>4. O sistema altera a rotação da textura na câmera.</li> </ol>
<b>Cenário alternativo:</b>	Nenhum

#### 5. Alternar entre câmeras

<b>Descrição</b>	
<b>Autor:</b>	O usuário
<b>Descrição sucinta:</b>	Seleciona uma câmera para ser visuzalizada
<b>Pré-condições:</b>	Ter pelo menos uma câmera criada. (Veja o Caso de Uso "Criar câmera")
<b>Pós-condições:</b>	Alternar entre as câmeras criadas, mostrando a visualização da câmera selecionada;
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O usuário seleciona uma câmera criada;</li> <li>2. O sistema alterna as câmeras, alterando a visualização do cenário.</li> </ol>
<b>Cenário alternativo:</b>	Nenhum

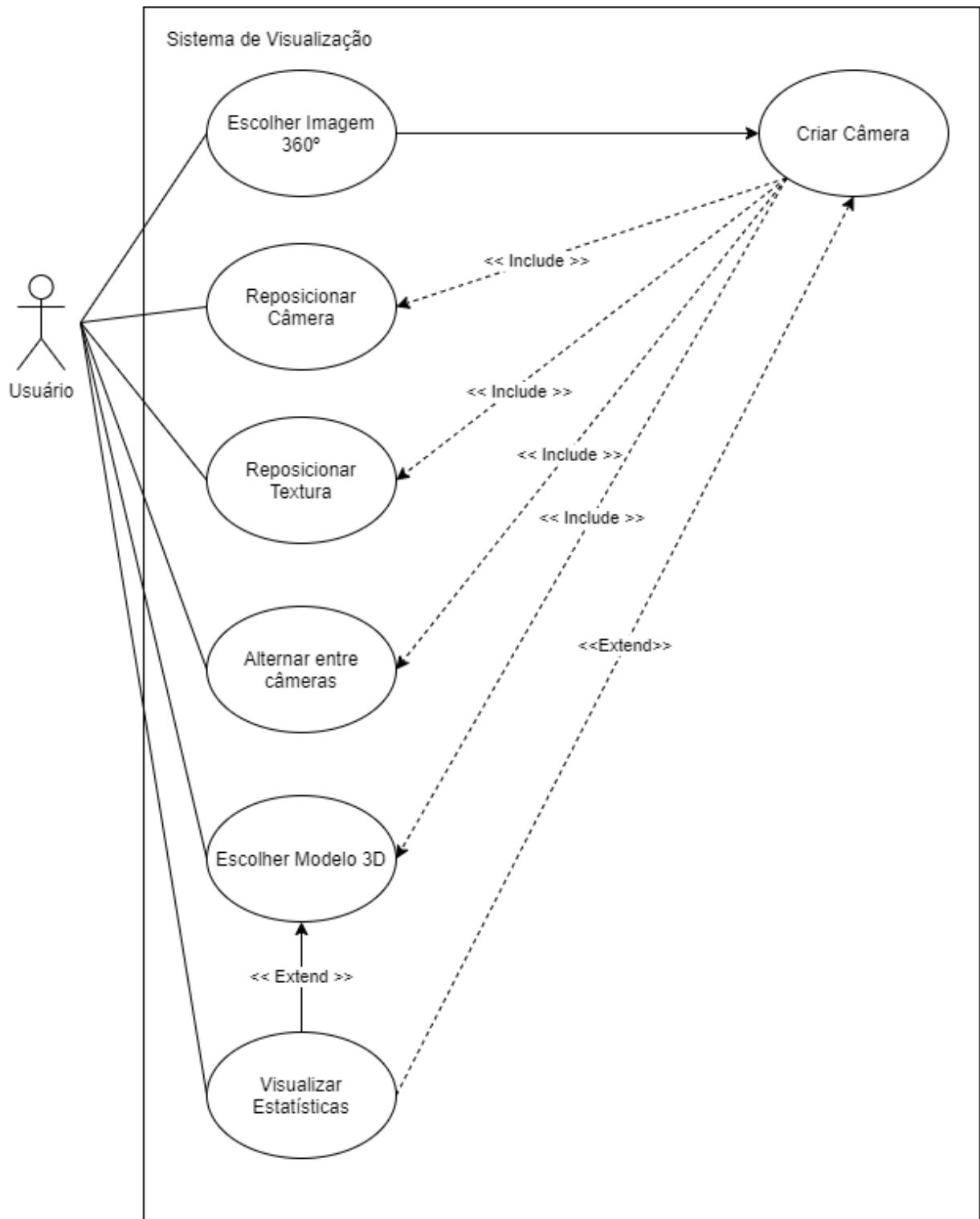
#### 6. Escolher modelo 3D

<b>Descrição</b>	
<b>Ator:</b>	O usuário
<b>Descrição sucinta:</b>	Seleciona uma câmera para ser visualizada
<b>Pré-condições:</b>	Ter pelo menos uma câmera criada. (Veja o Caso de Uso "Criar câmera")
<b>Pós-condições:</b>	Altera entre as câmeras criadas, mostrando a visualização da câmera selecionada
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O usuário escolhe a opção de "Escolher Modelo";</li> <li>2. O sistema exibe diversos modelos 3D para serem carregados;</li> <li>3. O usuário escolhe um dos modelos e aperta o botão "Selecionar Modelo";</li> <li>4. O sistema exibe o modelo 3D no cenário</li> </ol>
<b>Cenário alternativo:</b>	Se a conexão com o servidor estiver instável a listagem de imagens não será exibida.

## 7. Visualizar Estatísticas

<b>Descrição</b>	
<b>Ator:</b>	O usuário
<b>Descrição sucinta:</b>	Exibe as estatísticas
<b>Pré-condições:</b>	Estar com o sistema aberto
<b>Pós-condições:</b>	Exibe as informações de desempenho da aplicação
<b>Cenário principal:</b>	<ol style="list-style-type: none"> <li>1. O usuário acessa a página da aplicação;;</li> <li>2. O sistema exibe as informações de desempenho do cenário.</li> </ol>
<b>Cenário alternativo:</b>	Se a conexão com o servidor estiver instável estas informações podem não ser exibidas

Figura 6 – Diagrama de Casos de Uso



Fonte: Próprio Autor

# 4 Desenvolvimento

Nesta capítulo é explanado o processo de desenvolvimento da aplicação. Apresentando neste as tecnologias usadas até sua arquitetura.

## 4.1 Plataformas e tecnologias

A aplicação foi desenvolvida usando o conceito de cliente-servidor. No servidor a linguagem escolhida para desenvolvimento foi Python 3.8 com a biblioteca Flask. Foram usados a IDE Visual Studio Code da empresa Microsoft para suporte ao desenvolvimento. A instalação do Python usou o interpretador Anaconda para instalação de bibliotecas e execução do ambiente do desenvolvimento.

Para a construção da visualização do cliente, foi usando o ambiente de execução Node.js, responsável por rodar o framework Angular. A Interface Gráfica de Usuário (GUI), a foi montada usando HTML5, CSS e Typerscript. Para renderização do cenário 3D foram usadas a API Gráfica do HTML5, o HTML Canvas. Por sua vez, foi usado o HTML5 Canvas encapsulado pela biblioteca Three.js que permitiu um maior controle dos elementos 3D exibidos ao usuário. Para realizar algumas funcionalidades da GUI foram criadas métodos dentro do Typerscript.

A tabela abaixo resume as tecnologias e plataformas utilizadas para o desenvolvimento do programa:

<b>Linguagem de Programação</b>	Python 3.8 e Typescript
<b>IDE</b>	Visual Studio Code
<b>Ambientes de desenvolvimento</b>	Anaconda e Node.js
<b>Bibliotecas no Servidor</b>	Flask
<b>Bibliotecas no Cliente</b>	Angular, Three.js
<b>Plataformas</b>	Windows, Linux, Mac OS X
<b>Navegadores Suportados</b>	Firefox 4+, Safari 5.1+, Opera 15+, Google Chrome 9+ e Microsoft Edge

## 4.2 Arquitetura

A arquitetura do programa usa o padrão *Model-View-ViewModel* (MVVM), que é uma variação do padrão de arquitetura *Model-View-Controller* (MVC) e é mais voltado para construção de aplicativos que usam a API RESTful. Nesse padrão as classes são divididas em três partes:

- *Model*, ou Modelo. Onde acomoda os dados do aplicativo e fornece os métodos para acessa-los sistematicamente. Neste sistema é possíveis usa-los a partir de um endpoint (url).
- *View*, ou Visão. Representa a GUI, ela mostra representações do modelo e recebe as interações do usuário. Neste sistema ele é composto principalmente pelas páginas html.
- *ViewModel*, ou Visão-Modelo. Em vez de usar a abstração de controlador, onde a lógica do serviços são implementados, nós temos a *ViewModel* que fica responsável por trabalhar os dados recebidos pelo modelo e gerar uma visualização de acordo com as entradas do usuário dentro da View, isto é, a *ViewModel* fica responsável apenas pela lógica de exibição.

A figura 7 mostra a arquitetura do sistema.

### 4.3 Diagramas de Classes

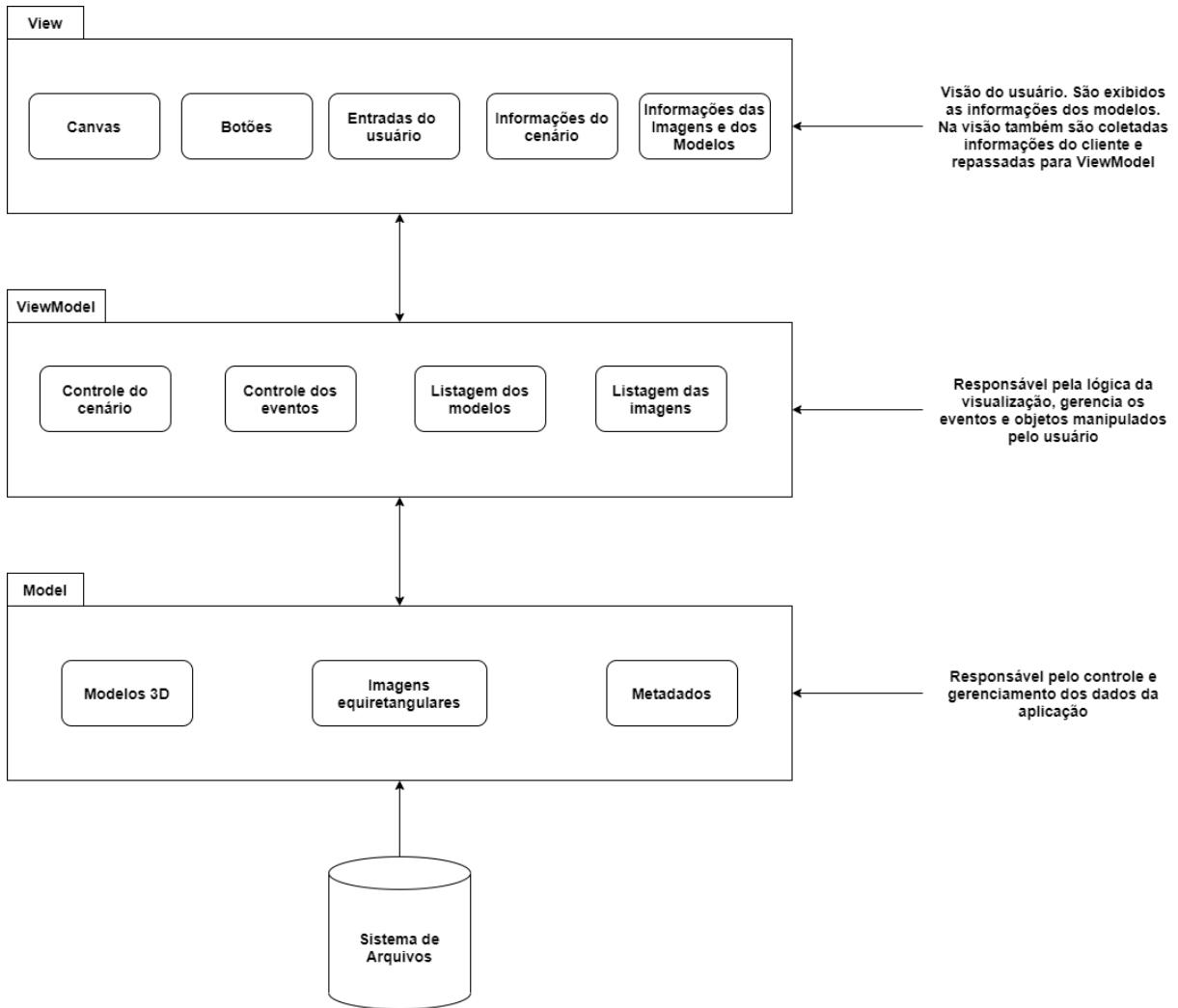
A figura 8 apresenta os diagramas de classe da aplicação. Para facilitar o entendimento, as classes em azul representam a camada da *Model*, as classes em Verde representam a camada *ViewModel*.

A biblioteca Three.js provê algumas classes necessárias para renderização do cenário na interface do usuário. As classes são citadas por fazerem parte da lógica da aplicação mas suas informações não são detalhadas na figura.

As classe *CanvasComponent* e *ScenarioTreeView* compreende toda a lógica da *ViewModel*, elas possuem funções necessárias para o controle da visualização do usuário, incluindo as informações do Cenário e gerenciamento de eventos. Para facilitar futuras exportações, estas informações são distribuídas nas classes *Scenario* e *CameraSettings*.

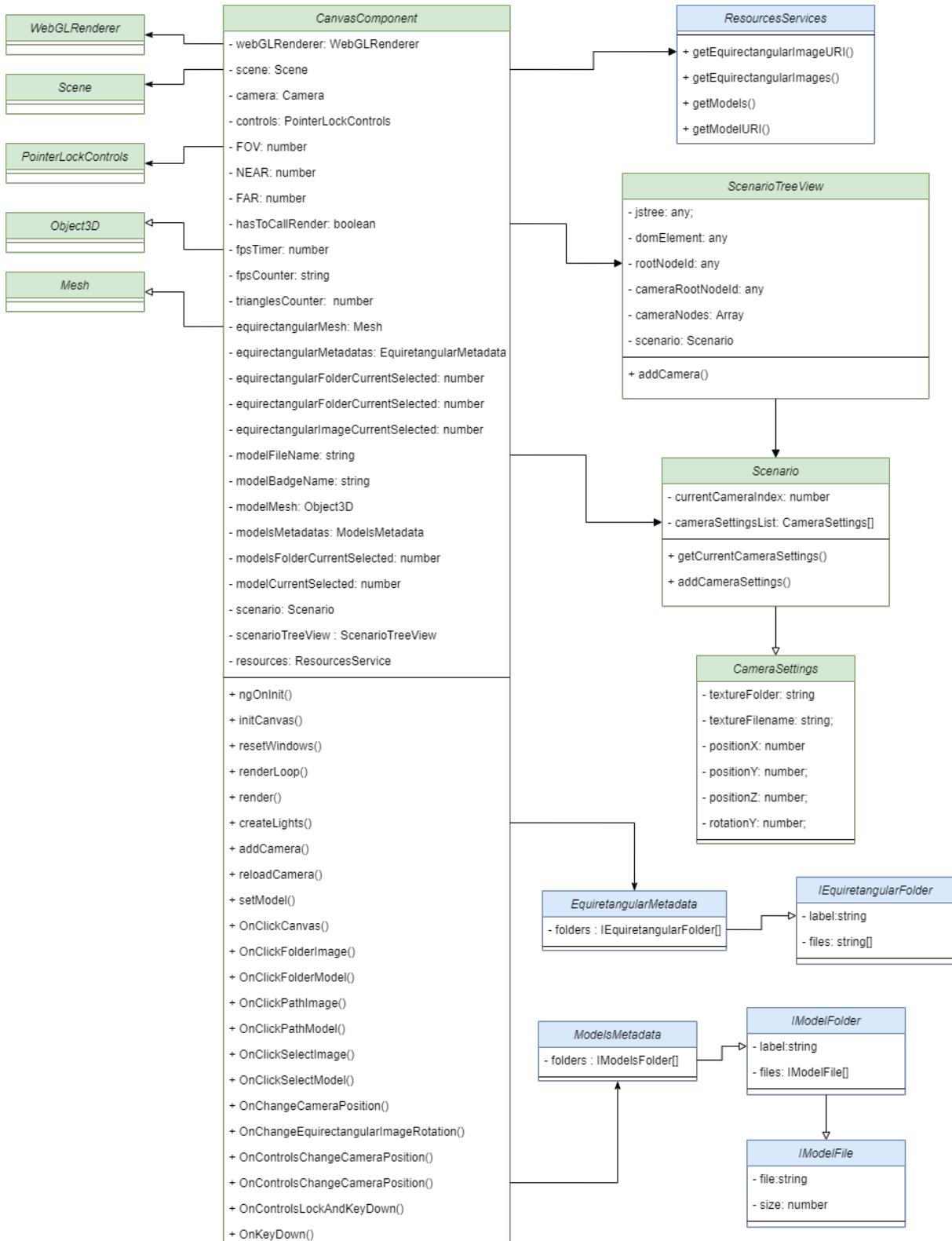
As classes *EquirectangularMetadata*, *I EquirectangularFolder*, *ModelsMetadata*, *IModelFolder* e *IModelFile* representam os dados armazenados no servidor. Elas são acessíveis a partir da classe *ResourcesServices*.

Figura 7 – Funcionamento da aplicação usando a arquitetura MVVM



Fonte: Próprio Autor

Figura 8 – Diagrama de Classes da Aplicação



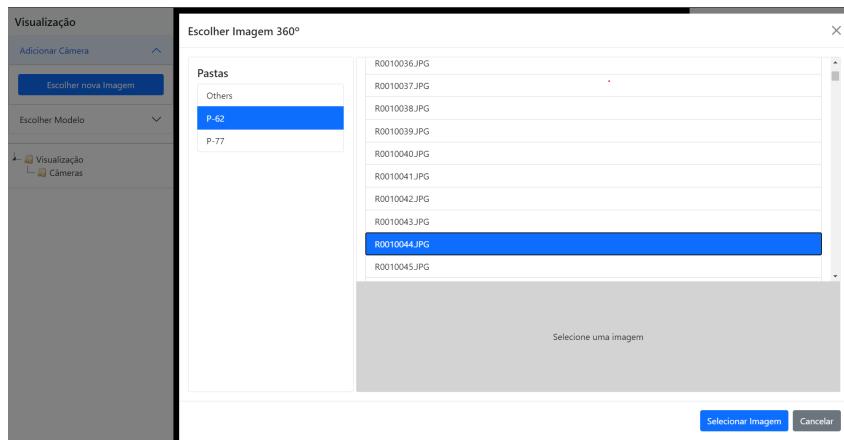
Fonte: Próprio Autor

## 4.4 Testes

Para avaliar a interface foi implementado um roteiro que explore todos os casos de usos da aplicação.

1. Clique no Botão de Adicionar Câmera
2. Escolha uma imagem 360º e Selecione (Figura 9)
3. Alterar a posição da Câmera e Rotação da imagem (Figura 10)
4. Adicionar uma nova câmera com uma imagem diferente
5. Alterar a posição da Câmera e rotação da imagem (Figura 11)
6. Alterar para a câmera criada anteriormente
7. Clicar no botão escolher Modelo
8. Escolher um Modelo e seleciona-lo (Figura 12 e 13)

Figura 9 – Tela de Escolha de Imagem



Fonte: Próprio Autor

Figura 10 – Alteração do posicionamento e rotação da câmera



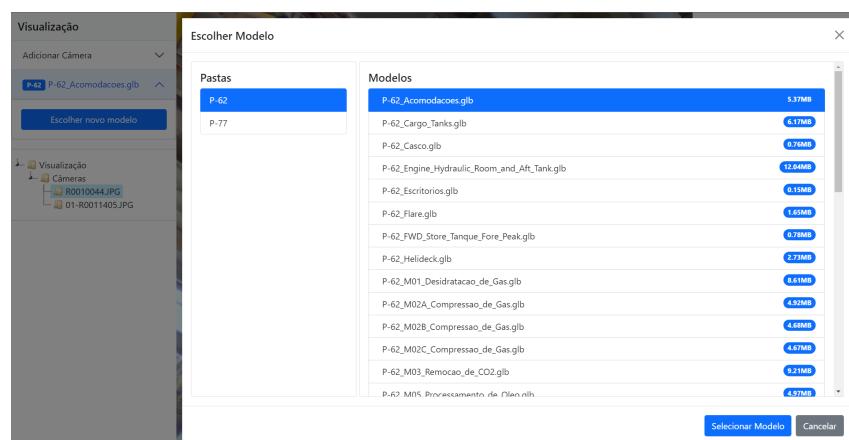
Fonte: Próprio Autor

Figura 11 – Alteração do posicionamento e rotação da segunda câmera



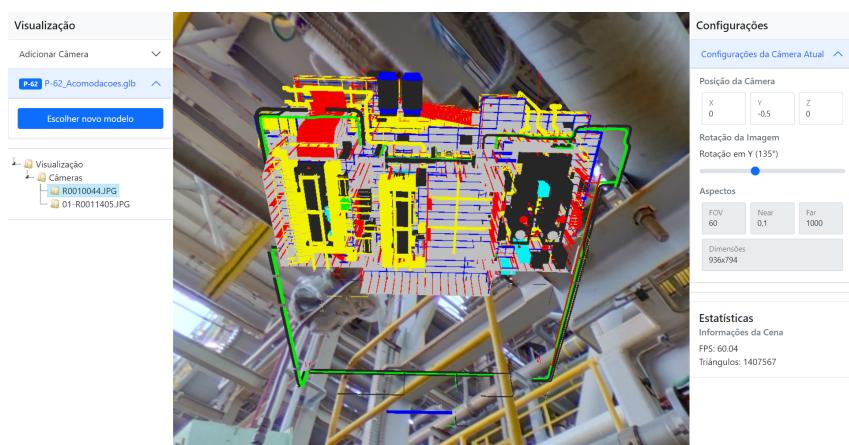
Fonte: Próprio Autor

Figura 12 – Tela de Escolha de Modelos



Fonte: Próprio Autor

Figura 13 – Resultado após a escolha do modelo



Fonte: Próprio Autor

# 5 Manual do usuário

Este capítulo apresenta o manual do usuário. É demonstrado como utilizar o programa para visualização de imagens 360º e Modelos 3D.

## 5.1 Requisitos necessários para hospedagem

Para instalação e execução é necessário possuir um ambiente python 3.8 ou superior com a biblioteca Flask disponível. É necessário que a máquina esteja conectada em uma rede para que usuários de fora da máquina instalada consigam usufruir da aplicação.

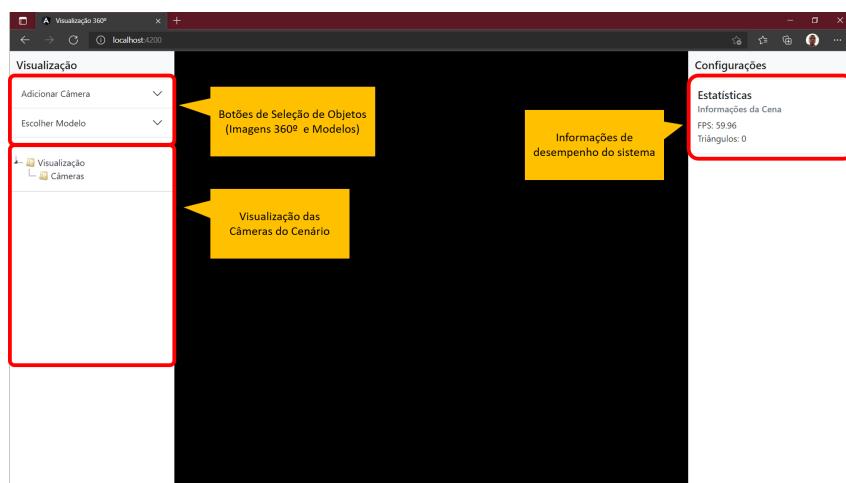
## 5.2 Requisitos mínimos

Após hospedado, o sistema encontra-se disponível através da rede, sendo acessível a partir de um navegador WEB. Para visualização e controle da interface recomenda-se possuir ao menos um mouse e teclado, visto que aplicação não foi projetada para dispositivos móveis.

## 5.3 Apresentação do Sistema

A figura 14 apresenta a interface inicial do sistema. O funcionamento de cada funcionalidade foi anexado a própria imagem para facilitar o entendimento.

Figura 14 – Tela inicial do Sistema

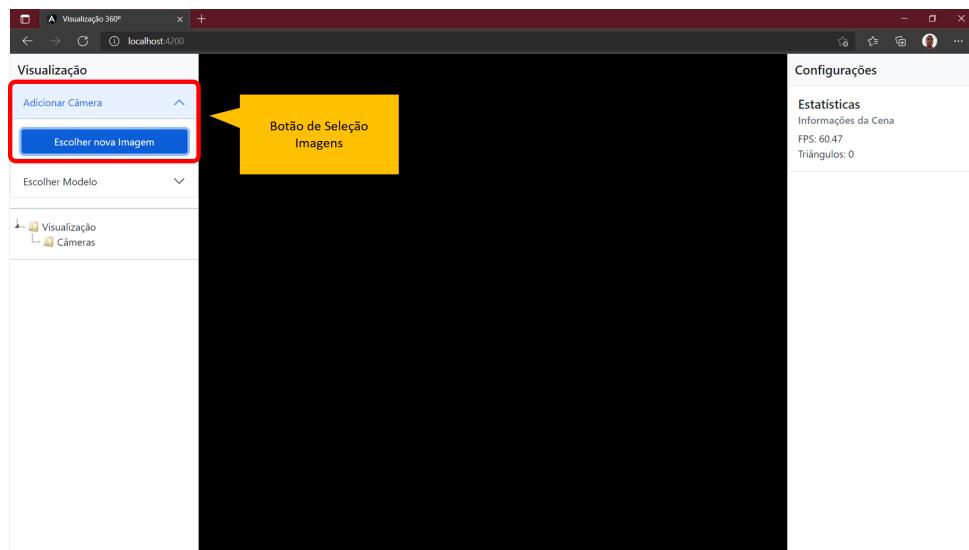


Fonte: Próprio Autor

## 5.4 Câmeras e Imagens 360°

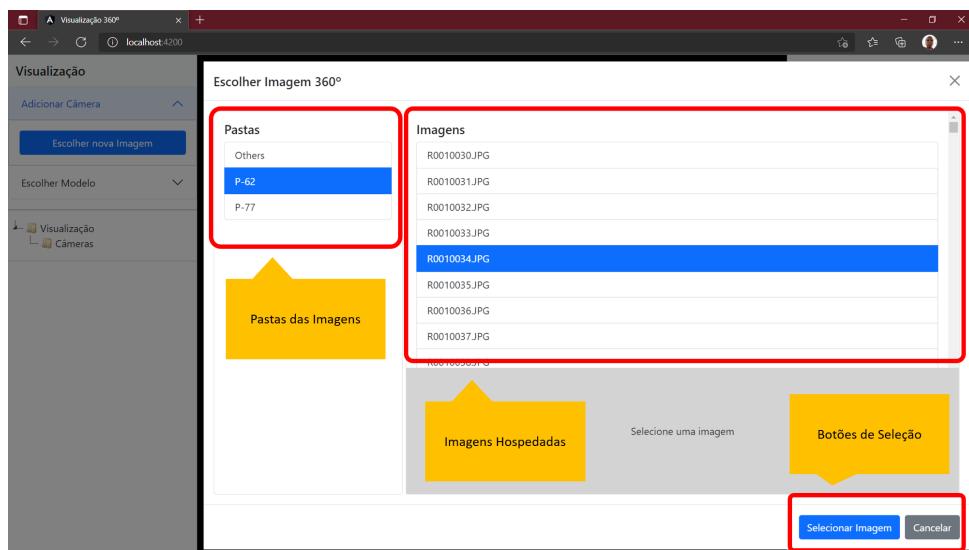
As figuras abaixo apresentam a interface relacionada as câmeras.

Figura 15 – Exibição do botão de adicionar câmera



Fonte: Próprio Autor

Figura 16 – Modal de seleção de câmera



Fonte: Próprio Autor

Figura 17 – Visualização das informações da câmera.



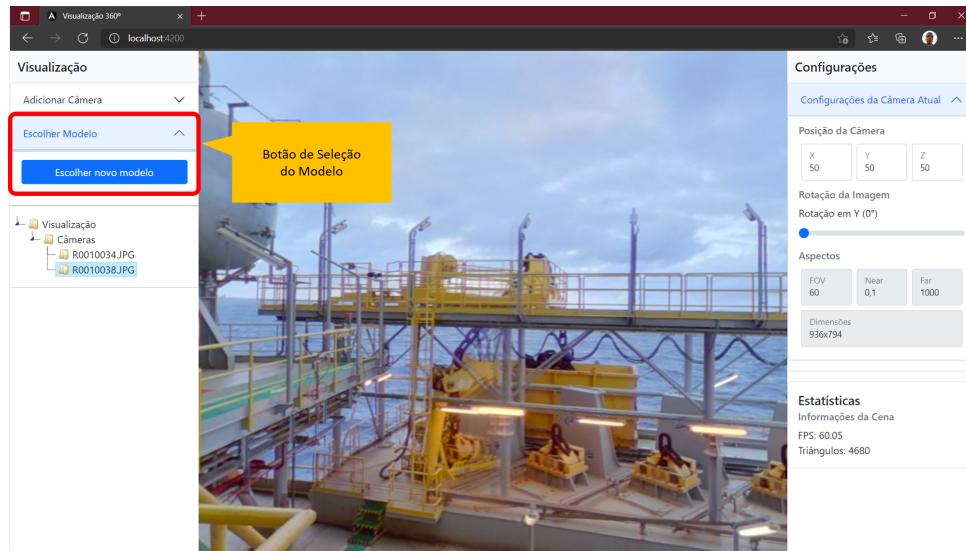
Fonte: Próprio Autor

Na figura 15 é destacado o botão de "Adicionar Câmera", ao clica-lo aparece a opção de escolher uma imagem. Ao clicar no botão de escolher imagem é exibido um *modal* de seleção de imagem, como mostrado na figura 16. As imagens são categorizadas por uma pasta e pelo seu nome de arquivo, é possível cancelar a operação ou selecionar a imagem para criar uma câmera. Ao criar uma câmera a imagem é exibida na área destacada em laranja na figura 17. As configurações ajustáveis da câmera ficam disponíveis na área destacada com a cor verde. Para adicionar mais câmeras, basta repetir o passo-a-passo das imagens. É possível alternar entre elas usando a visualização em árvore das câmeras do cenário

## 5.5 Modelos

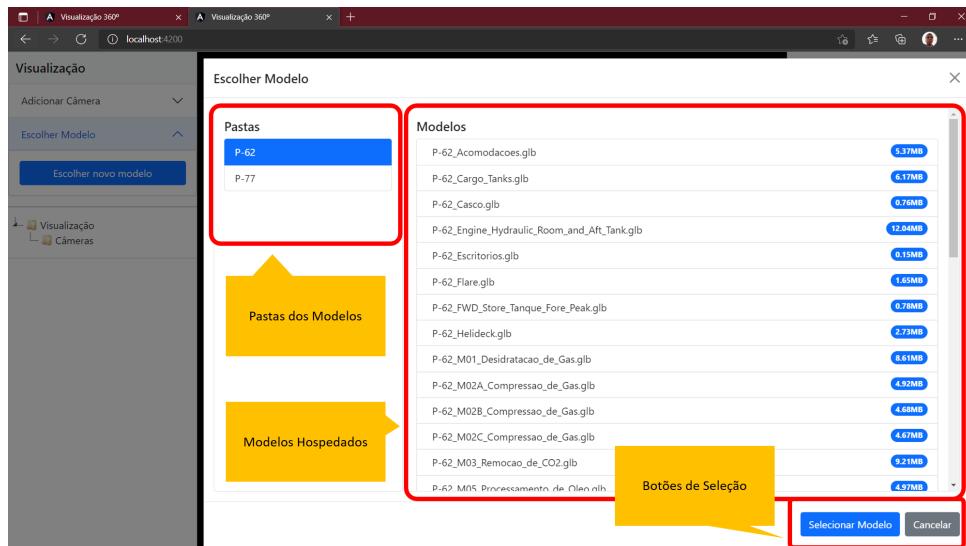
As figuras abaixo apresentam a interface relacionada aos modelos.

Figura 18 – Exibição do botão de Escolher Modelo



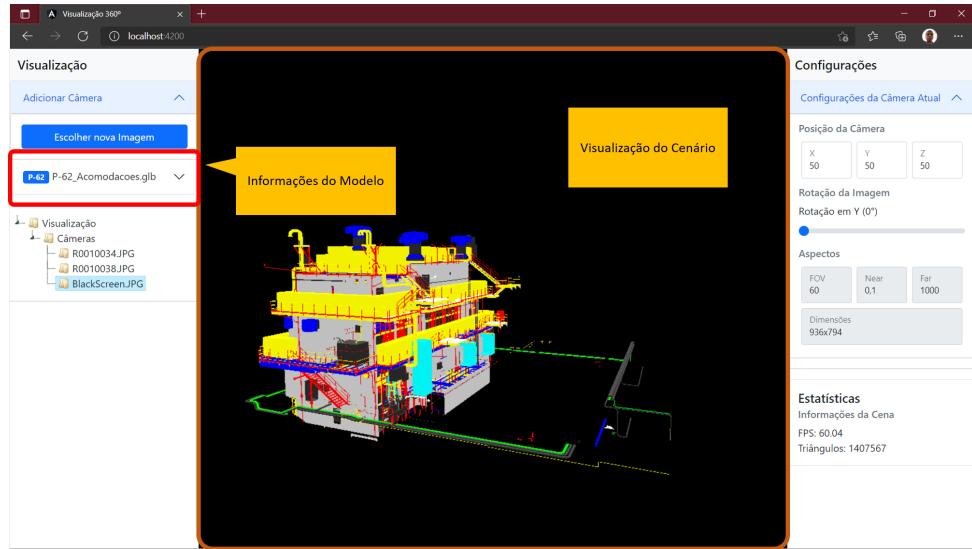
Fonte: Próprio Autor

Figura 19 – Modal de seleção de modelo



Fonte: Próprio Autor

Figura 20 – Visualização das informações do modelo.



Fonte: Próprio Autor

Assim como nas estapas de adição de câmera, é exibido uma opção de "Escolher novo modelo" (ver figura 18). Um *modal* exibe os modelos disponíveis para visualização, separados por pastas, é possível usar o botão de selecionar o modelo para carrega-lo no cenário, como observado nas figuras 19 e 20.

## 5.6 Navegação

A aplicação possui um sistema de navegação intuitivo para posicionamento da câmera. Pode-se considerar uma alternativa as opções de configuração da câmera. Para usá-la basta clicar em qualquer lugar do canvas, você receberá um aviso que o seu mouse está preso ao cenário, como visto na figura 21.

Figura 21 – Aviso de Navegação



Fonte: Próprio Autor

Os controle de posicionamento passam a ser as teclas do teclado e o mouse.

- Para ir para frente, você pode apertar W ou seta para cima;
- Para ir para trás, você pode apertar S ou seta para baixo;
- Para ir para esquerda, você pode apertar A ou seta para esquerda;
- Para ir para direita, você pode apertar D ou seta para direita;
- Para ir para cima, você pode apertar E ou espaço;
- Para ir para baixo, você pode apertar Q ou Shift esquerdo.

# Referências

- CABELLO, R. *Three.js*. 2021. Acessado em 29/06/2021. Disponível em: <<https://github.com/mrdoob/three.js/blob/dev/README.md>>. Citado na página 7.
- CôTé, S. et al. Live mobile panoramic high accuracy augmented reality for engineering and construction. In: . [S.l.: s.n.], 2013. Citado na página 5.
- EIRIS, R.; GHEISARI, M.; ESMAEILI, B. Pars: Using augmented 360-degree panoramas of reality for construction safety training. *International Journal of Environmental Research and Public Health*, v. 15, n. 11, 2018. ISSN 1660-4601. Disponível em: <<https://www.mdpi.com/1660-4601/15/11/2452>>. Citado na página 4.
- GATTASS, M. *OpenGL Render Pipeline*. 2013. Acessado em 29/06/2021. Disponível em: <[http://webserver2.tecgraf.puc-rio.br/~mgattass/LivroCG/07\\_OpenGLRenderPipeline.pdf](http://webserver2.tecgraf.puc-rio.br/~mgattass/LivroCG/07_OpenGLRenderPipeline.pdf)>. Citado na página 6.
- GOOGLE. *Google Maps*. 2021. Acessado em 29/06/2021. Disponível em: <<https://maps.google.com>>. Citado na página 4.
- INTEL OPEN SOURCE. *RENDERING IMMERSIVE WEB EXPERIENCES WITH THREE.JS AND WEBXR*. 2019. Acessado em 29/06/2021. Disponível em: <<https://01.org/blogs/darktears/2019/rendering-immersive-web-experiences-threejs-webxr>>. Citado na página 8.
- KHRONOS GROUP. *OpenGL*. 2021. Acessado em 29/06/2021. Disponível em: <[https://www.khronos.org/opengl/wiki/Main\\_Page](https://www.khronos.org/opengl/wiki/Main_Page)>. Citado na página 6.
- WEISSTEIN, E. *Equirectangular Projection*. 2021. From MathWorld—A Wolfram Web Resource. Acessado em 29/06/2021. Disponível em: <<https://mathworld.wolfram.com/EquirectangularProjection.html>>. Citado na página 4.
- ZHANG, S. Three-dimensional range data compression using computer graphics rendering pipeline. *Applied optics*, Optical Society of America, v. 51, n. 18, p. 4058–4064, 2012. Citado 2 vezes nas páginas 5 e 7.