LUIZ FELIPE BAPTISTA SERAPIÃO

ATLAS DE ANATOMIA VETERINÁRIA 3D:

REFATORAÇÃO

LAVRAS – MG
2023

LUIZ FELIPE BAPTISTA SERAPIÃO

ATLAS DE ANATOMIA VETERINÁRIA 3D: REFATORAÇÃO

Relatório Técnico apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel.

APROVADA em 28 de novembro de 2023.

Dra. Ana Paula Piovesan Melchiori	UFLA
Dr. Gregório Corrêa Guimarães	UFLA
Dr. Bruno de Abreu Silva	UFLA

gov.br
Documento assinado digitalmente
ANA PAULA PIOVESAN MELCHIORI
Data: 18/12/2023 10:41:05-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Ana Paula Piovesan Melchiori

Orientador (a)

LAVRAS – MG

2023

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Serapião, Luiz Felipe Baptista.
Atlas digital de anatomia veterinária 3D : Refatoração / Luiz Felipe Baptista Serapião. - 2023.
60 p.

Orientador(a): Ana Paula Piovesan Melchiori.

TCC (graduação) - Universidade Federal de Lavras, 2023.
Bibliografia.

1. Software Educacional. 2. WebGL. 3. Computação Gráfica. I. Melchiori, Ana Paula Piovesan. II. Título.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu forças para prevalecer nesta etapa da minha trajetória.

À minha família, José, Rita e Ana Carolina, que me ofereceram amor, conforto, confiança e suporte durante toda a minha vida, até mesmo nos momentos mais difíceis.

Aos meus amigos, companheiros de república, Vitor Dourado, Vitor Cazumbá, Selwinn e Giuliano. Pela amizade e companheirismo, numa casa onde reinava a alegria e risada.

À minha amiga de infância, Gabriela, que me deu suporte emocional e conselhos ao longo de toda essa fase.

À minha orientadora, professora Ana Paula Piovesan Melchiori, que me guiou nas últimas etapas do meu curso, e no desenvolvimento deste trabalho.

Ao professor Bruno de Abreu Silva, por ter oferecido ajuda e atenção, com sugestões para o trabalho, assim como orientação na empresa júnior.

RESUMO

O espaço educacional se adapta e se renova constantemente, sendo notável atualmente, com o avanço tecnológico, especialmente computacional, e as iniciativas dos órgãos educacionais para integrarem esta nova realidade em seu espaço. Por iniciativa de docentes e discentes da Universidade Federal de Lavras, o Atlas Digital de Anatomia Veterinária 3D, conhecido como ADAV 3D, foi desenvolvido a este mesmo fim. Neste projeto, ADAV foi recriado, de forma que as necessidades das equipes que o mantém sejam atendidas, assim como, para que o sistema esteja preparado para futuras ampliações, sem a necessidade de refatorar o mesmo num futuro próximo novamente. O presente trabalho apresenta os desafios enfrentados na manutenção da ferramenta, suas limitações técnicas que a levaram a ser reconstruída, as necessidades identificadas pela equipe do projeto, assim como o seu desenvolvimento para alcançar os seus objetivos, adicionando um banco de dados com api ao sistema, uma nova aplicação WebGL dinâmica e reativa, com suporte a mobile, assim como áreas de cadastro, edição e revisão das submissões dos modelos. O sistema foi desenvolvido com bibliotecas front-end como ReactJS, onde se pode criar uma aplicação web inteiramente com HTML, CSS e JavaScript. A fim de criar uma aplicação WebGL, foram utilizadas bibliotecas como Three.js, React Three Fiber e React Three Drei, que permitem renderizar e manipular o modelos 3D de ossos criados em Blender cadastrados no sistema. Por fim, a aplicação possui uma API desenvolvida com Django Rest Framework, que é um framework Python completo para desenvolvimento de API assim como integração com banco de dados SQLite.

Palavras-chave: WebGL. React Three Fiber. Three.js. Django Rest Framework. Computação Gráfica. Anatomia Veterinária; Blender. Software Educacional.

ABSTRACT

The educational space adapts and innovates constantly, being noticeable currently, with the technological advancements, specially in computing and the initiatives of educational agencies to integrate this new reality in its space. By initiative of students and professors of The Federal University of Lavras (UFLA), the Digital Atlas of Veterinary Anatomy 3D, also known as ADAV 3D, was developed for this purpose. In this project, ADAV was rebuilt, to address the necessities of the adav team, as well, to make it prepared for future expansions, avoiding another refactoring of this project in the near future. This present document presents the challenges faced in the maintenance of the legacy tool, it´s technical limitations, why it was required a complete refactoring, the necessities identified by the project team, as well as its development to reach its goals, adding a database with api to the system, a new dynamic and reactive WebGL application with mobile support, model register, edit and review components. The web system was completely developed in HTML, CSS and JavaScript. To create a WebGL application, it was used librarys such as ThreeJS, React Three Fiber and React Three Drei, allowing to render and manipulate 3D models created on Blender and registered on the database. Finally, the application has an API developed in Django Rest Framework, a complete python framework for development of APIs as well integration with SQLite databases.

Keywords: WebGL. React Three Fiber; Three.js. Django Rest Framework. Computer Grafics; CG. Veterinary Anatomy. Blender. Educational Software.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

3D	Tri-dimensional
ADAV	Atlas Digital de Anatomia Veterinária
API	Interface de Programação de Aplicação
CRA	Create React App
CSS	Folha de Estilo em Cascata
DAC	Departamento de Computação Aplicada
DCC	Departamento de Ciência da Computação
DMV	Departamento de Medicina Veterinária
DRF	Django Rest Framework
HTML	Linguagem de Marcação de Hipertexto
R3F	React Three Fiber
UFLA	Universidade Federal de Lavras
UI	Interface de usuário

LISTA DE ABREVIATURAS

Drei	React Three Drei
Engine	Engine de desenvolvimento de jogos
React	ReactJS
Three	Three.js

SUMÁRIO

1. INTRODUÇÃO

Neste Capítulo, é apresentado a contextualização do surgimento do projeto ADAV 3D e seu objetivo, seus problemas que deram origem ao presente trabalho, assim como os objetivos do projeto.

1.2. Contextualização

Acompanhar as mudanças tecnológicas e culturais é um desafio enfrentado pelo meio acadêmico e educacional desde seus primórdios, e um de seus mais recentes vem sendo o desafio da digitalização global e como integrá-la na educação. O ADAV 3D foi uma iniciativa inovadora desenvolvida na Universidade Federal de Lavras, projeto colaborativo entre os departamentos do DMV (Departamento de Medicina Veterinária), DCC (Departamento de Ciência da Computação) e DAC (Departamento de Computação Aplicada). Apresentado no CIUFLA 2015, o poster da equipe do ADAV descreve o projeto como "O Atlas Digital 3D de Anatomia Veterinária é um aplicativo interativo que possui o objetivo de proporcionar um ensino de qualidade ao aluno, podendo também ser utilizado pelos professores como fonte de consulta...", (Caetano *et al.*, 2015, n.p.).

De acordo com Cuba *et al.* (2020), o uso de tecnologias como scanners e impressoras 3D podem gerar aplicações de apoio ao ensino. A este fim, através de escaneamento de modelagem de ossos 3D, junto a suas informações, deu-se origem à mais recente versão do ADAV 3D. O projeto consistiu de um sistema WEB com sessões 2D e 3D, o qual o usuário podia analisar e aprender sobre os ossos presentes no sistema, separados por categorias animais (Ruminantes, Carnívoros e Equinos).

O presente trabalho trata-se de um projeto de computação aplicada, afim de exercer os conhecimentos de computação e criar uma solução útil para a área de Medicina Veterinária e seus estudantes, através de um software educativo que simule o ambiente de laboratório, e permita total inspeção de um modelo de osso, assim como suas devidas informações de estudo.

Este projeto foi feito com apoio da Agência Zetta, que disponibilizou seus equipamentos para estudo e desenvolvimento do mesmo, finalizado remotamente, no notebook pessoal do autor deste trabalho.

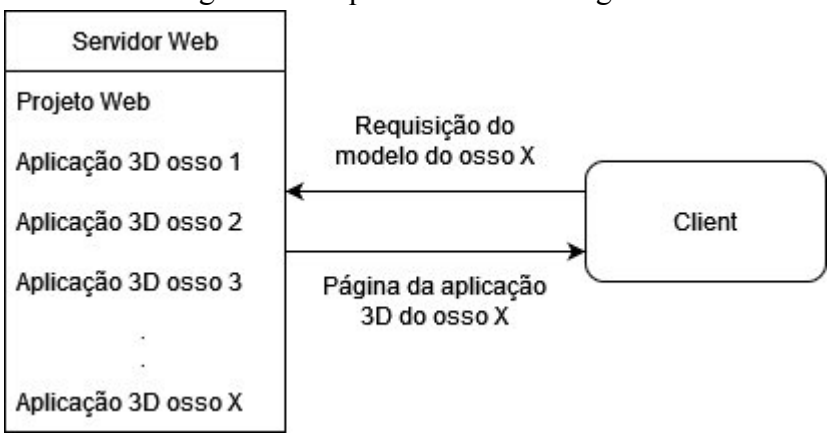
.

1.3. Definição do problema

A construção inicial do projeto foi um sucesso, porém, não pensada para futuras expansões. Isso gerava grande trabalho para as equipes de computação e veterinária sempre que o sistema expandia para receber novas categorias de animais e/ou ossos, pois era necessário atualizar seu código-fonte toda vez, já que era apenas um projeto front-end em php, sem um banco de dados para armazenar informações, junto a aplicações 3D isoladas e estáticas, precisando gerar novas aplicações para cada osso novo cadastrado no sistema.

Na figura 1, é representado a arquitetura do sistema ADAV Legado, onde um *cliente*[1] faz uma requisição de uma das X aplicações 3D armazenadas diretamente no servidor Web.

Figura 1 - Arquitetura ADAV Legado



Fonte: Do Autor (2023)

Os problemas técnicos do sistema não visavam escalabilidade, e criavam forte dependência da equipe do ADAV sob desenvolvedores disponíveis, pois não podem fazer o cadastro de um novo osso ou mesmo a edição de algum campo, já que um sistema estático não permite este tipo de controle para o usuário final, com informações presentes no próprio código do sistema e de suas aplicações. Por isso, viu-se necessário reconstruir o projeto, de forma que a equipe do ADAV tenha maior independência do DCC, assim como, possa expandi-lo sem grandes dificuldades, preenchendo apenas um simples formulário, armazenando suas informações em um banco de dados, e com uma aplicação WebGL dinâmica eliminando a necessidade de uso de *engines*[2].

---

[1] Aplicação web que permite acesso de usuários a informações de um servidor.

[2] Engine de jogos são ferramentas de desenvolvimento de jogos e aplicações 3D.

1.4. Objetivo

O projeto consiste na refatoração completa do ADAV a fim de se criar um sistema capaz de armazenar os dados de interesse para os ossos, assim como gerar uma aplicação WebGL totalmente dinâmica, sem necessidade de modificações manuais para cada osso cadastrado. O sistema deve ser totalmente reativo, que funcione em diversos tipos de aparelhos, especialmente aparelhos mobile, sendo o meio de acesso à web mais comum ao brasileiro, onde 96% dos usuários de internet no Brasil, fazem uso de aparelhos mobile, (CETIC, 2017, p.11).

1.4.1. Aplicação WebGL dinâmica e responsiva

Para que seja possível atingir o objetivo proposto, o sistema precisou adotar uma nova estratégia em como criar uma aplicação 3D. O sistema legado utilizava aplicações WebGL estáticas, feitas em engine *Unity*[3] e 'isoladas' do resto do sistema WEB, o qual apenas redireciona a pessoa para a aplicação específica do osso selecionado.

Esta abordagem demonstra-se inviável ao se considerar um sistema dinâmico, que o discente de veterinária possa cadastrar o osso e suas descrições, pois para tal, teria que também criar uma nova aplicação WebGL para toda expansão do atlas com novos ossos, criando uma dependência forte das equipes do ADAV aos discentes do DCC e DAC.

Para atacar esta problemática, foram desenvolvidos dois templates (desktop e mobile), e um renderizador 3D com todas as funcionalidades requisitadas pela equipe, escrito em JavaScript, HTML e CSS, utilizando as bibliotecas ReactJS e R3F, sendo bibliotecas de construções de front-end WEB e aplicações WebGL em React, respectivamente.

1.4.2. Sistema intuitivo e funcional

ADAV foi construído pensado para usuários não técnicos, e para tal, toma uma aproximação de design minimalista e limpa, com poucas opções por tela, sem privar o usuário das informações necessárias para aquilo que está acessando.

1.4.3. Maior controle dos orientadores do projeto

Ao dar maior controle e autonomia para a equipe de discentes do projeto, é importante que existam novas formas de controle de seus respectivos orientadores no cadastro e edição de

---

[3] UNITY, Unity, 2023. Disponível em: https://unity.com/pt Acesso em: 05/11/2023.

conteúdo do sistema, antes que seja possível visualizá-lo pelo usuário final. A este fim, toda submissão feita no sistema, seja de cadastro ou edição de algum campo, antes de ser disponibilizada ao usuário final, passa por um processo de avaliação do docente responsável, que junto das informações de submissão terá disponível seu autor, data, e poderá aprovar, reprovar ou até mesmo editar uma submissão enviada. Desta forma, o sistema se protegerá de erros e ataques vindo de usuários discentes, impedindo a entrada de informações incorretas e/ou inapropriadas ao sistema.

1.4.4. Compatibilidade com modelos previamente escaneados

Houve extenso trabalho no escaneamento, geração e adequação dos modelos 3D dos ossos para o sistema anterior. Para evitar a perda de trabalho, o novo sistema foi feito de forma a adequar-se ao trabalho já desenvolvido, com apenas pequenas adequações que devem ser feitas sobre os modelos para que possam funcionar de forma correta no novo sistema, sem a necessidade de gerá-los novamente. As adequações que precisam ser feitas devem também ser simples o suficiente para um discente com pouco conhecimento na ferramenta *Blender* possa fazê-los sem grande esforço ou investimento de tempo.

1.5. Justificativa

O sistema ADAV, quando originalmente desenvolvido, demonstrou-se ser um sistema de sucesso para Universidade Federal de Lavras. A equipe do projeto trabalhava ativamente em sua expansão e escaneamento de novos ossos. Assim como, o sistema tinha uso para fins de estudos por nossa universidade, e teve até adoção de universidades externas em seu ensino. O sistema implementado com sua aplicação WebGL era intuitivo e inovador, e contribuiu muito para o ensino.

Com seus problemas técnicos e dissolução da equipe de desenvolvedores original, o sistema começou a apresentar os sinais de um sistema legado, com aplicações quebradas, estáticas e sistema inacessível por autenticação de um banco de dados que não mais existia.

Sendo seu maior problema os desafios técnicos enfrentados, é interessante para a equipe do ADAV a refatoração do sistema, com tecnologias modernas e modular, facilitando recrutamento de novos discentes para a equipe de desenvolvimento.

É interessante também para a equipe do DMV, capaz de corrigir e adicionar novos ossos sem necessidade do envolvimento de desenvolvedores para novos cadastros ou correções.

O sistema também é de interesse do DMV, para uso em salas de aula, tendo acesso a

uma extensiva quantidade de ossos registrados no sistema para seu ensino.

Por fim, o sistema tem grande potencial para a UFLA como um todo, expandir departamentos parceiros do sistema, assim como quais tipos de modelos são registrados no mesmo, sendo uma base sólida para um sistema de ensino 3D.

2. CONCEITOS E TECNOLOGIAS UTILIZADAS

Neste Capítulo, são documentadas as ferramentas e tecnologias utilizadas no desenvolvimento do projeto, junto a seus conceitos, com um breve estudo de cada ferramenta, seu propósito e uso.

As Subseções deste Capítulo são divididos por suas áreas computacionais, sendo elas "Computação gráfica", "Desenvolvimento Web" e "Gestão de Projeto".

2.1. Computação gráfica

A computação gráfica é a área computacional de estudo de processamento e cálculos na criação de imagens e modelos, que em termos menos técnicos, significa processar e criar uma saída visual para o usuário final. Segundo Conci (2019), esta área é subdivida em três grandes áreas de estudo:

- **Análise de Imagens – AI:** Área de estudo de imagens matriciais, focada em análise para obtenção de dados e informações.
- **Processamento de Imagens – PI:** Área de estudo de imagens matriciais, focada em manipulação de imagens, como mudança de cores e uso de filtros.
- **Síntese de Imagens – SI (ou CG):** Área de estudo de imagens vetoriais, focada em produção de imagens sintéticas a partir de descrição matemática dos objetos.

O projeto desenvolvido se enquadra em um trabalho SI, que através de software de modelagem, é produzido um modelo de osso 3D, que se trata de um objeto vetorial constituído de polígonos, permitindo através de cálculos matemáticos, a criação de um objeto que pode ser inspecionado de todos os ângulos em aplicações 3D.

2.1.1. Software de modelagem 3D

O software adotado pela equipe ADAV para modelagem 3D é o *Blender*[4], que é um software de código aberto e gratuito, utilizado para criação de conteúdo 3D. Ele oferece uma ampla gama de recursos, incluindo modelagem, animação, renderização, composição, edição de vídeo, simulação de partículas e muito mais. O Blender é conhecido por sua comunidade

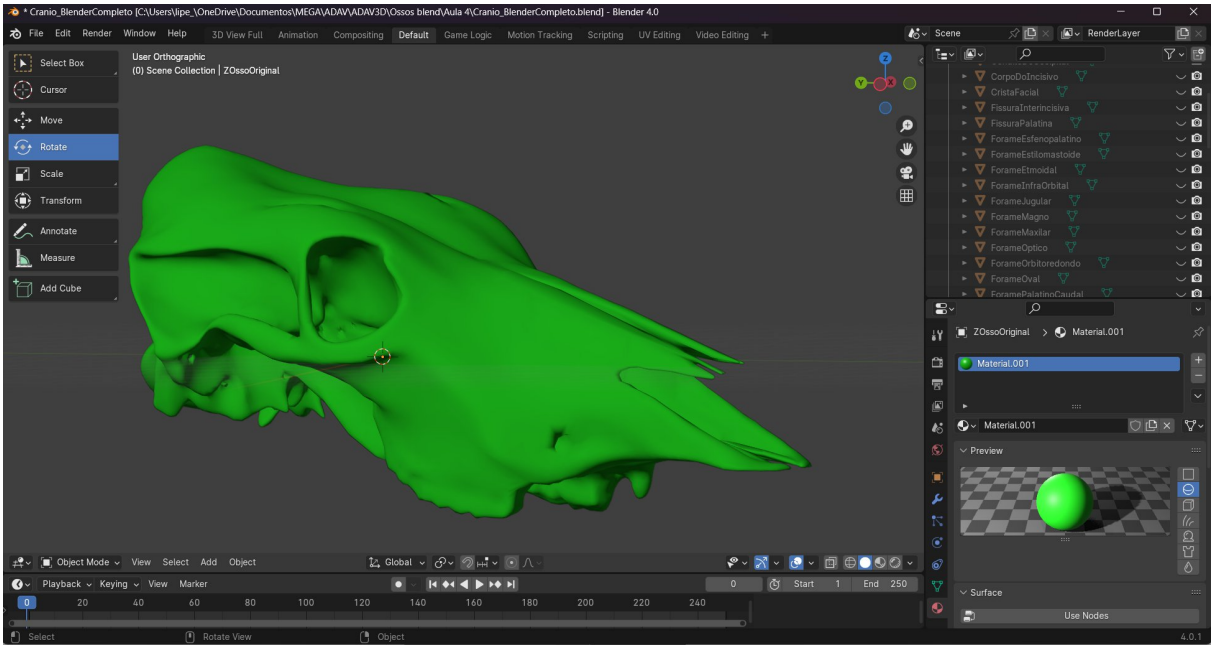
[4] BLENDER, Blender, 2023. Disponível em: https://www.blender.org/ Acesso em: 05/11/2023.

ativa e por ser uma ferramenta poderosa e versátil, sendo utilizada tanto por profissionais quanto por entusiastas. Sua interface pode ser inicialmente desafiadora para iniciantes, mas sua flexibilidade e recursos robustos o tornam uma escolha popular para uma variedade de projetos, desde animações simples até produções cinematográficas complexas.

Junto a equipamentos de scanners, a equipe do ADAV 3D fez a modelagem de 4 aulas do seu sistema legado, dando um total de 42 modelos de ossos salvos no *Google Drive*[5] do projeto, desenvolvidos e modelados no Blender, com seus acidentes sobrepostos ao modelo do osso manualmente através de *meshs*[6], por uma equipe especializada em modelagem 3D.

Um exemplo de seu uso pela equipe pode ser visto nas figuras 2 e 3, com o modelo do crânio da aula 4, módulo ruminantes, onde seus acidentes:

Figura 2 - Crânio sem acidentes

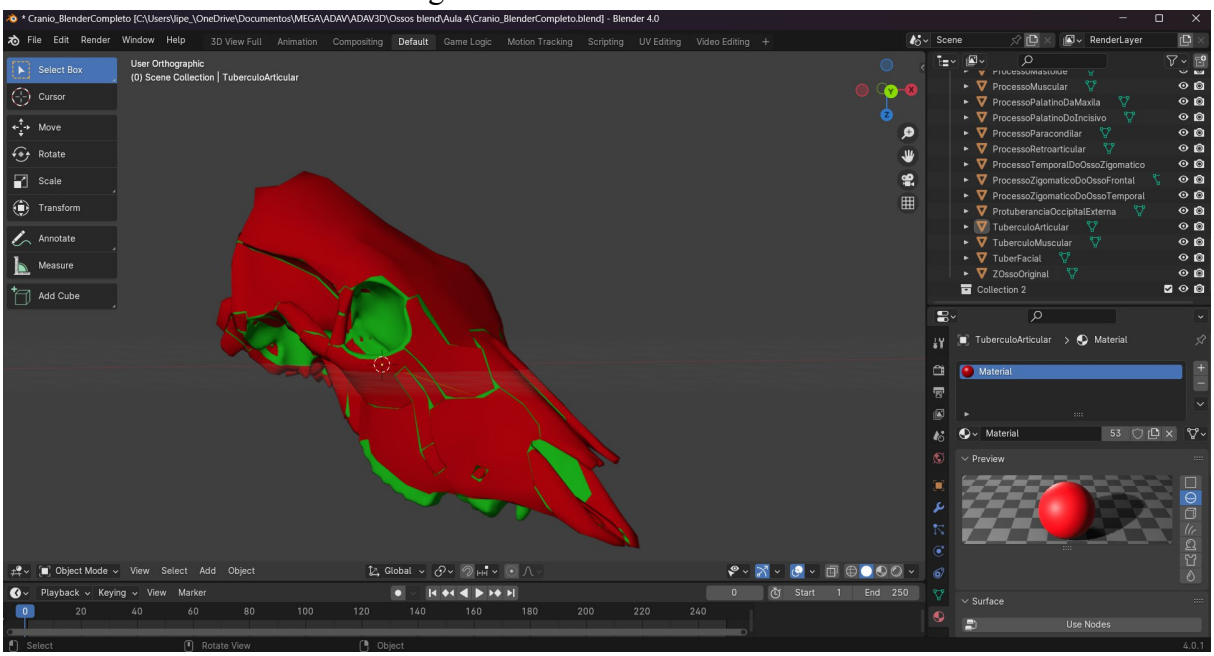


Fonte: Do Autor (2023)

[5] GOOGLE DRIVE, Google. Disponível em: https://www.google.com/intl/pt-br/drive/about.html?offset=480 Acesso em: 05/11/2023.

[6] Malha gráfica composta de vértice, geometria e fragmento.

Figura 3 - Crânio com acidentes



Fonte: Do Autor (2023)

2.1.2. Bibliotecas 3D utilizadas

O *Three.js*[7] é uma biblioteca JavaScript para desenvolvimento de aplicações de realidade virtual em navegadores web. Em seu repositório no *Github*, equipe do Three.js (2023) descreve que seu objetivo é criar uma biblioteca 3D de uso geral, fácil de utilizar, leve, com suporte a múltiplos navegadores. Projetada para ser flexível e intuitiva, Three.js abstrai muitos detalhes complexos de programação gráfica 3D, permitindo aos desenvolvedores criar facilmente cenas tridimensionais, animações e efeitos visuais diretamente no navegador. Ela utiliza WebGL para renderização eficiente e suporta a criação de ambientes 3D interativos, jogos e simuladores. Além disso, possui documentação extensiva e inúmeros projetos e bibliotecas criando novas abstrações, ou até mesmo integrando o projeto em outras bibliotecas. No caso da biblioteca utilizada no projeto, React Three Fiber[8] (R3F), que é uma biblioteca que integra a biblioteca Three.js com o ecossistema React, possibilitando a criação de elementos 3D de forma declarativa e reativa, aproveitando as capacidades do React para construir interfaces de usuário interativas com gráficos tridimensionais.

---

[7] THREE.JS., Three.js, 2023. Disponível em: https://threejs.org/ Acesso em: 08/11/2023

[8] REACT THREE FIBER, **React Three Fiber Docs**, React Three Fiber, 2023. Disponível em: https://docs.pmnd.rs/react-three-fiber/getting-started/introduction Acesso em: 02/11/2023

Por fim, também foi feito o uso de *React Three Drei*[9] (Drei), que é uma biblioteca adicional construída sobre o R3F, projetada para simplificar ainda mais o desenvolvimento de aplicações WebGL em React. Ela fornece uma variedade de componentes, utilitários e abstrações para facilitar tarefas comuns ao trabalhar com R3F.

2.2. Desenvolvimento Web

O desenvolvimento web é comumente separado em duas categorias, o desenvolvimento front-end, que se trata do desenvolvimento das páginas acessadas pelo usuário final, incluindo toda sua interface e conteúdo, assim como estilizações e rotas. E o back-end, sendo as engrenagens do sistema, área de desenvolvimento de APIs e Bancos de dados.

2.2.1. Biblioteca front-end web

O ReactJS, ou simplesmente *React*[10], é uma biblioteca JavaScript desenvolvida pelo Facebook para construir interfaces de usuário interativas e eficientes. Focado em criar componentes reutilizáveis, o React permite o desenvolvimento de UIs declarativas, onde as alterações no estado da aplicação automaticamente atualizam a visualização correspondente. De acordo React (2023), “O React permite que você crie interfaces de usuário a partir de peças individuais chamadas de componentes.”. Isso resulta em um desenvolvimento mais eficiente, através do uso inteligente de componentes comuns ao sistema, acelerando o processo de desenvolvimento e facilitando manutenções.

Junto ao React, foi adotado o uso de Vite, no lugar de CRA (Create React App, previamente ferramenta padrão para criação de aplicações React, descontinuada) e Babel. Vite é uma ferramenta de construção de aplicações web rápida e moderna para JavaScript. Desenvolvida para facilitar o desenvolvimento, o Vite oferece um ambiente de desenvolvimento extremamente ágil, com recarregamento rápido, suporte a módulos ESM (ECMAScript Modules), e uma arquitetura centrada em recursos como Vue.js e React. De acordo com Vite (2023), “ Vite é uma ferramenta de construção de projetos de frontend que se destina a oferecer uma experiência de desenvolvimento mais rápida e leve para projetos de web modernos.”.

O contexto da aplicação, assim como sua lógica de chamadas asíncronas à API, foi feito

---

[9] DREI, React Three Drei, 2023. Disponível em: https://github.com/pmndrs/drei Acesso em: 07/11/2023.
[10] REACT, React, 2023. Disponível em: https://pt-br.react.dev/ Acesso em: 08/11/2023.

utilizando de Redux Toolkit[11], que é uma biblioteca que simplifica e otimiza o desenvolvimento de aplicações Redux em React. Ele oferece um conjunto de utilitários que ajudam a reduzir a quantidade de código *boilerplate* e facilitam tarefas comuns, como a criação de fatias do estado da aplicação, a definição de redutores e a gestão de ações. No site oficial do Redux Toolkit (2023), a equipe afirma que a ferramenta "inclui vários utilitários para simplificar casos de uso comuns em configuração de *store*, criação de redutores e lógica de atualização imutável." É uma biblioteca que permite o desenvolvimento da lógica central da aplicação web com menos código.

2.2.2. Banco de dados

O banco de dados da aplicação foi construído em *SQLite*[12], sendo este "uma biblioteca de linguagem C que implementa um pequeno, rápido, auto-suficiente, de alta confiabilidade, com todas funcionalidades, mecanismo de banco de dados SQL." (SQLITE, 2023), ou seja, se trata de um banco da dados relacional, que destaca-se por sua estratégia de armazenamento de dados, diferente de outros bancos de dados que armazenam suas informações em um servidor, já que o SQLite faz o armazenamento de todas suas informações em um único arquivo.

2.2.3. Framework backend e API

O *framework* utilizado na construição da API desenvolvida, junto da modelagem do banco de dados descrito no Subcapítulo anterior, foi o *Django Rest Framework*[13] (DRF). O DRF é uma extensão do Django para facilitar a criação de APIs *RESTful* em Python. Ele fornece recursos como serializadores para converter dados, funcionalidades de rotas baseadas em classes e funções, autenticação flexível, roteamento automático, e documentação interativa.

---

[11] REDUX TOOLKIT, Redux Toolkit, 2023. Disponível em: https://redux-toolkit.js.org/ Acesso em: 08/11/2023.
[12] SQLITE, SQLite, 2023. Disponível em: https://www.sqlite.org/index.html Acesso em: 06/11/2023.
[13] DJANGO, Django Rest Framework, 2023. Disponível em: https://www.django-rest-framework.org/ Acesso em: 07/12/2023.

2.3. Gestão de Projeto

As tarefas desenvolvidas para o projeto foram documentadas utilizando a ferramenta web *shortcut*[14], que permite o cadastro, compartilhamento, pontuação e atribuições de tarefas, junto a nomes, descrições, prioridades e anexos. Que foram salvas em diferentes períodos de desenvolvimento, chamadas na ferramenta de *Epics*[15] que foram utilizadas como *Sprints* para a equipe de desenvolvimento. Os conceitos de Sprints[16] e Epics, junto a metodologias ágeis, utilizadas pela equipe de desenvolvimento, são mais aprofundadas no Capítulo de "Metodologia"**.**

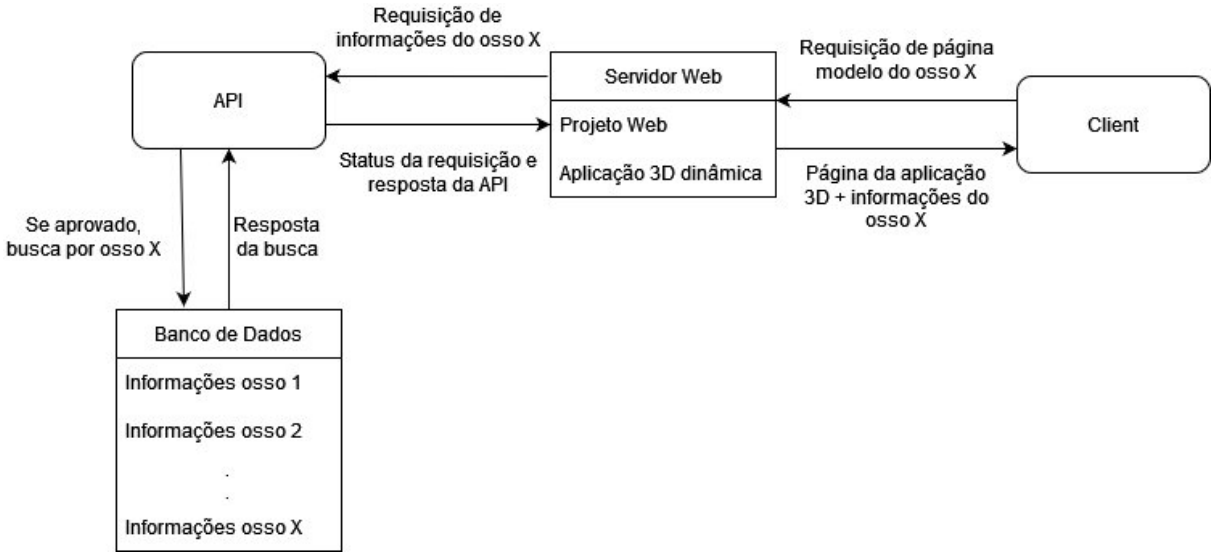
[14] SHORTCUT, Shortcut, 2023. Disponível em: https://www.shortcut.com/ Acesso em: 06/11/2023.
[15] Termo utilizado pela ferramenta shortcut para sinalizar "sprints"
[16] Termo utilizado no framework Scrum, utilizado para descrever um período de tempo fixo para desenvolvimento de atividades

3. METODOLOGIA

Neste capítulo, é documentada a metodologia utilizada para o desenvolvimento do projeto. Este projeto foi desenvolvido pelo autor deste trabalho, como desenvolvedor Full-Stack e líder do projeto, auxiliado por um desenvolvedor bolsista pela PROMAD, Lucas Gabriel Pereira Moreira, que trabalhou no desenvolvimento do design das novas páginas para o projeto, assim como no desenvolvimento front-end. O projeto foi desenvolvido em React, com suas tecnologias padrão (HTML, CSS, JavaScript), com API desenvolvida com DRF, banco de dados em SQLite, e por fim, a aplicação WebGL foi desenvolvida utilizando a biblioteca gráfica Three.js, integrada em React, através do React Three Fiber (ou R3F).

Na figura 4, é apresentado a arquitetura utilizada para o desenvolvimento do novo ADAV, com a inclusão de uma nova aplicação 3D, capaz de requisitar dados dos modelos de ossos 3D dinamicamente:

Figura 4 - Arquitetura ADAV Refatorado



Fonte: Do Autor (2023)

3.1. Metodologia ágil

O projeto foi desenvolvido seguindo os doze princípios de metodologia ágil, descritos no *Manifesto para Desenvolvimento Ágil de Software* (BECK et al., 2001):

1. Priorizar satisfação do cliente, através da entrega contínua de software com valor agregado.
2. Mudança de requisitos são bem vindas em qualquer etapa de desenvolvimento.
3. Entregas frequentes de software funcional, com menor intervalo de tempo possível.
4. Times de negócios e desenvolvimento devem trabalhar em conjunto durante todo o projeto.
5. Desenvolva projetos ao redor de indivíduos motivados.
6. Forma mais eficaz de comunicação é através de uma conversa face a face.
7. Medida de progresso principal é software funcional.
8. Desenvolvimento sustentável, que preserve ritmo constante de desenvolvimento indefinidamente.
9. Agilidade é obtida através da atenção contínua da boa técnica e design.
10. Simplicidade, ou descomplicação no desenvolvimento é essencial.
11. Equipes auto-organizáveis geram as melhores arquiteturas, requisitos e design.
12. Em intervalos regulares, equipe se reune e determina como se tornar mais eficaz.

3.1.2. Scrum Simplificado

Para desenvolvimento deste projeto, a equipe de desenvolvimento fez uso de metodologias ágeis, através do *framework*[17] Scrum, definido na obra *The Scrum Guide*. De acordo com o guia, "Scrum é um framework simples que ajuda pessoas, times e organizações a gerar valor através de soluções adaptáveis para problemas complexos", (SCHWABER; SUTHERLAND, 2020, p.4).

Devido a relativa baixa complexidade do projeto desenvolvido, o Scrum foi adaptado de forma simplificada, em que os dois desenvolvedores eram membros do Time Scrum, e com as funções de *Product Owner* e *Scrum Master* atribuídas ao autor do trabalho.

[17] Estruturas, podendo conter diretrizes, convenções, ferramentas e bibliotecas para desenvolvimento de software.

As tarefas do *backlog* do projeto foram separadas em três *sprints*, cada uma contendo um mês de duração. Durante as sprints, o time Scrum repassava *dailys* de forma asíncrona, para acompanhamento do caminhar do projeto e as tarefas de cada membro. Uma vez por semana, a equipe reunia-se em reunião, chamada *weakly*, para discutir a sua performance naquela semana, trabalhar em conjunto em tarefas não completadas da semana, e se reajustar, estendendo tempo de desenvolvimento de tarefas ainda não completadas e atribuindo novas tarefas ao time.

O guia define os termos utilizados da seguinte forma:

- **Scrum Master:** Responsável por preservar as regras Scrum, assim como garantir o seu devido uso pela equipe de desenvolvimento.
- **Product Owner:** Responsável por maximizar o valor gerado pelo time Scrum.
- **Backlog:** Ou Product Backlog, é uma lista do que é necessário ser desenvolvido para melhorar o produto, fonte principal de tarefas do time Scrum.
- **Sprint:** Períodos fixos de tempo, de um mês ou menos, com tarefas, pontuações e time atribuido.
- **Daily:** Reunião breve, de no máximo 15 minutos, feita de forma diária, afim de inspecionar o progresso das atividades sendo desenvolvidas.
- **Weakly:** Não expresso no guia, adaptação da equipe para possuir uma reunião síncrona para trabalho em conjunto semanalmente, assim como utilizada para eventos de *Review* e *Retrospective* do Scrum.
- **Review:** Processo ao fim de uma Sprint, para analisar resultados e planejar futuras adaptações.
- **Retrospective:** Análise de performance da sprint e maneiras de melhora-la.

4. DESENVOLVIMENTO

A fim de organizar a documentação das etapas de desenvolvimento do projeto, as mesmas foram separadas em Capítulos que abordam separadamente a etapa desenvolvida, assim como seu procedimento. Pela extensão da etapa do “Estudo da aplicação ADAV3D”, o mesmo foi separado em Subseções, abordando as necessidades de estudo de seus modelos, aplicação WebGL e a solução desenvolvida para o novo projeto.

4.2. Identificação de requisitos

O projeto tem como maior objetivo resolver as problemáticas identificadas pelos orientadores do projeto ao longo dos anos ao utilizar e desenvolver o projeto original, então para recolhimento de seus primeiros requisitos, esta etapa foi feita através entrevistas *one on one*, onde foi feita a entrevista de cada orientador individualmente e recolhido a maior quantidade de informações possível dos problemas enfrentados pela equipe, assim como capturar requisitos desejáveis para o sistema.

Os requisitos identificados podem ser vistos no Quadro 1:

Quadro 1 - Requisitos Funcionais

#RF	Nome RF	Descrição RF
01	Tipos de usuários	O sistema possui 3 tipos de usuários, sendo eles usuário administrador, aluno e visitante (não autenticado)
02	Cadastrar categoria	O sistema deve ser capaz de cadastrar novas categorias, as quais apenas terão um nome.
03	Cadastrar animal	O sistema deve ser capaz de cadastrar novos animais, animais serão separados em nome e espécie.
04	Cadastrar osso	O sistema deve permitir o usuário cadastrar novos ossos, que possuirão título, subtitulo, descrição, foto, modelo3D, conjunto de vistas e acidentes do mesmo.
05	Editar categoria	O sistema deve permitir a edição dos campos de uma categoria
06	Editar animal	O sistema deve permitir a edição dos campos de um animal
07	Editar osso	O sistema deve permitir a edição dos campos de um osso.
08	Análise de submissão	Toda submissão de cadastro ou edição deve passar por uma análise manual de um orientador responsável.
09-01	Aplicação WebGL	O sistema deve implementar uma aplicação WebGL 3D, capaz de renderizar os ossos modelados.
09-02	Controles de câmera	A câmera da aplicação deve ser capaz de girar ao redor do osso 360°.
09-03	Captura de hover/click em acidentes	Aplicação 3D deve ser capaz de capturar o nome do acidente ao ter a ação de hover ou click em cima do mesmo.
09-04	Seleção de vista do osso	O aplicativo deve possuir um seletor de vistas cadastradas para o osso. (ângulo de visão do modelo 3d)
09-05	Campos de informações	Aplicação possuirá campos de informações do osso, contendo: título, subtitulo, descrição do osso, descrição da vista selecionada.
09-06	Aplicação dinâmica	Deve ser feito apenas duas aplicações WebGL (desktop e mobile) capazes de recolher as informações de um banco de dados e alterarem seus campos/modelos dinamicamente.

Fonte: Do Autor (2023)

Além das funções necessárias para fazer o sistema funcionar, também existem vários requisitos não funcionais recolhidos destas reuniões, acima de tudo, sobre a experiência de usuário no uso do sistema e sua simplicidade, assim como reatividade. Estes requisitos podem ser observados na Quadro 2:

Quadro 2 - Requisitos não funcionais

#RNF	Nome RNF	Descrição RNF
01	Página de cadastro de osso intuitiva	Página de cadastro do osso possui muitos campos para serem cadastrados, assim como arquivos visuais, como foto e modelo 3D, por fim, uma ou mais vistas associadas ao osso. Para que o discente do DMV possa fazer o cadastro, a página deve ser o mais intuitiva e simples possível, com pré-visualização de imagens, modelos e vistas.
02	Identidade visual preservada	O sistema deve ser imediatamente reconhecível como ADAV, isso se diz no quesito de cores, logos, designs e experiência de usuário em geral
03	Posicionamento de botões de interesse intuitivos	Para um discente fazer um cadastro ou uma edição no sistema, deve ser imediatamente óbvio para o mesmo onde acessar as listas de categorias/animais/ossos do sistema para poder editá-los.
04	Análise simplificada	Para que um docente possa fazer uma análise de uma submissão, deve ser claro para ele quais são suas opções, oque foi submetido, por quem e quando.
05	Experiência de usuário consistente	Sistema deve ser capaz de manter uma boa experiência de usuário nos mais diversos tipos de aparelhos, exigindo que suas páginas e componentes sejam responsivos.
06	WebGL de alta performance	Aplicação 3D é feita para ser utilizada por públicos dos mais diversos tipos de hardwares, incluindo smartphones e laptops de baixa especificações.

Fonte: Orientadores ADAV (2023)

4.3. Manutenção do sistema original

Para reconstruir uma ferramenta de forma fiel ao seu objetivo, foi necessário compreender como a mesma originalmente funcionava. Porém, o sistema se encontrava inacessível, requisitando autenticação de um banco de dados que já não existia mais. Suas funcionalidades apresentavam-se de forma vaga na memória dos orientadores responsáveis. Ficou evidente que para melhor compreender a aplicação 3D que deveria ser recriada, era necessário rodar o projeto e fazer as manutenções necessárias para acessá-lo.

Ao receber acesso ao projeto original, os seguintes problemas foram identificados:

- Autenticação exigida pra acessar todas as páginas;
- Aplicações WebGL com javascript legado, utilizando de uma função não mais suportada por navegadores: console.logError;
- Havia uma aplicação WebGL estática própria para cada osso, cada aplicação tendo em média 200 mb de peso, com scripts, binários e htmls gerados por

Unity;

- Função de autenticação descentralizada, documento duplicado para cada subpasta do sistema contendo mesmo código.

Ao remover conteúdo de todos os arquivos de controle de autenticação, assim como atualizar os scripts Unity trocando console.logError por console.error, o sistema volta a funcionar e foi feita a analise de como era Atlas Digital 3D.

4.4. Estudo da aplicação ADAV 3D

Neste Seção, é documentado como foi feita a análise da aplicação ADAV3D, os procedimentos adotados em seu estudo, os problemas identificados no processo, assim como a solução gerada.

4.4.1. Avaliação de viabilidade de reaproveitamento do ADAV 3D

Ao selecionar um osso no sistema, pode ser inspecionado a aplicação 3D, representada pela Figura 5:

Figura 5 - Aplicação ADAV3D Legado

Fonte: Do Autor (2023)

Aqui se encontra presente uma aplicação que atende boa parte dos requisitos recolhidos sobre a aplicação 3D, mas possui algumas exceções chaves, que inviabilizam a reutilização assim como expansão da aplicação previamente desenvolvida, utilizando os códigos de requisitos dos quadros 1 e 2:

- **RF09-06:** Aplicação existente é estática e gerada por engine, criar uma aplicação Unity dinâmica geraria enorme retrabalho ou até refatoração completa da aplicação, e manteria a mesma desconexa do resto do sistema (impossibilitando pré-visualização de modelos 3D nas páginas de cadastro/edição, por exemplo).
- **RNF05:** De acordo com Unity Manual (2022), Unity WebGL não oferece suporte oficial para aparelhos mobile, este é um requisito chave para o sistema.
- **RNF06:** Aplicações WebGL Unity geram vários arquivos e scripts de tamanho considerável, isso impacta diretamente a performance do cliente em sua visualização, assim como do servidor em seu armazenamento e resposta de requisições.

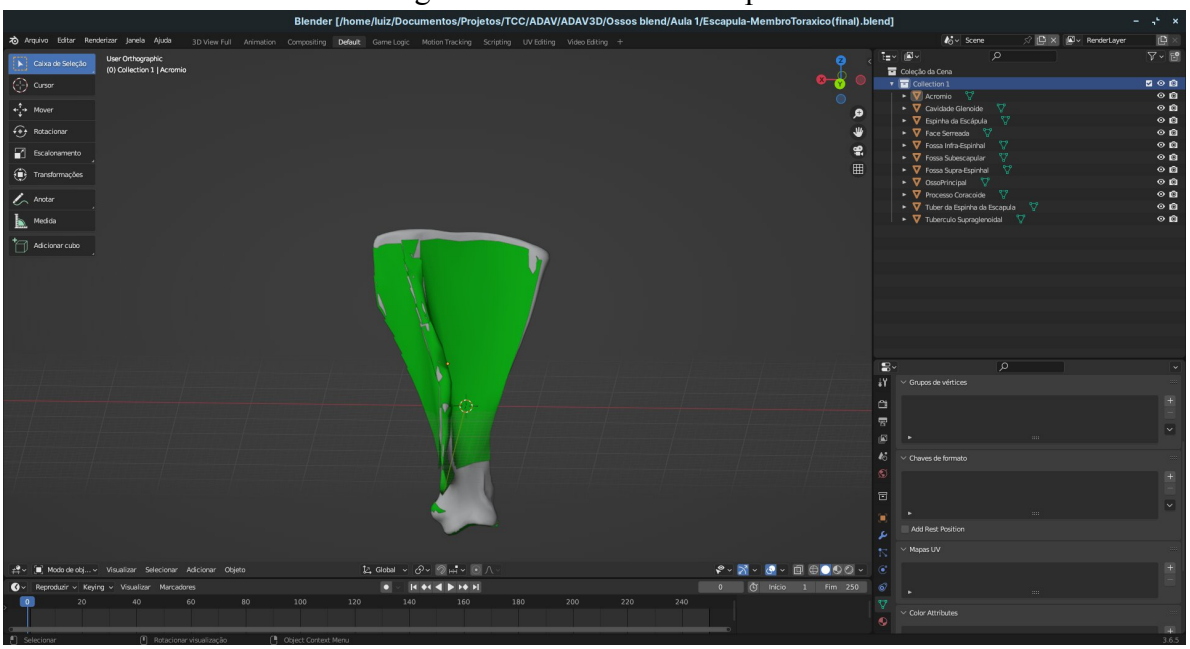
4.4.2. Estudo e adequação dos modelos 3D gerados para o projeto

Independente da viabilidade da reutilização da aplicação 3D, outro requisito chave para o sistema era o reaproveitamento dos modelos 3D de ossos previamente gerados, tendo vários já cadastrados no sistema anterior e ainda mais prontos para sua expansão, seria inviável substituir a aplicação 3D por uma que não comporte os modelos gerados.

Feito a análise do Google Drive da equipe, com todos os arquivos do projeto, incluindo seus modelos, foi confirmado que seus ossos estavam armazenados em arquivos .blend, o que significa que foram modelados e com seus acidentes adicionados utilizando a ferramenta Blender. A ferramenta utilizada para abrir o arquivo foi o Blender 3.6.5 empacotado oficialmente via Flatpak, Linux. Os modelos dos ossos são separados em seu modelo principal do osso, e meshs que o cobrem, demarcando as regiões dos seus acidentes. Todos os modelos encontram-se normalizados (centralizados no eixo da cena).

Na figura 6, é possível analisar o modelo 3D da Escápula, sua estrutura, assim como as mudanças feitas em um modelo para adequá-lo ao novo sistema 3D do projeto (abordado no Subseção seguinte).

Figura 6 - Modelo 3D Escápula



Fonte: Do Autor (2023)

A figura 6 demonstra como é inicialmente a tela ao abrir um dos modelos 3D do projeto de um osso. À direita, há uma lista contendo vários meshs pertencentes a este modelo, todos numa mesma coleção, sem uma ordem específica, com um mesh chamado de “OssoOriginal” ou “OssoPrincipal”, que representa o modelo do osso, e outros meshs na mesma coleção são seus acidentes ósseos. A primeira adequação necessária para o novo renderizador 3D, foi separar o modelo do osso original e de seus acidentes em coleções diferentes, de forma que o osso original seja o primeiro modelo do arquivo, em seguida, corrigir os nomes dos modelos de acidentes, pois é a partir destes nomes que serão capturados os nomes de acidentes impressos na aplicação 3D.

Lista de meshs já adequados em duas coleções, com osso principal na primeira, e acidentes na segunda, juntamente de nomes corrigidos pode ser vista como exemplo na Figura 7:

Figura 7 - Meshs e seus nomes adequados

Fonte: Do Autor (2023)

Na próxima etapa, é necessário garantir que todos os meshs possuam um material diferente, isso é porque estes meshs serão “pintados” na nova aplicação, e caso possuam um mesmo material, todos serão coloridos uniformemente, diferente do efeito desejado. Isso pode ser facilmente resolvido, indo de mesh em mesh, na seção material e selecionando “novo material”.

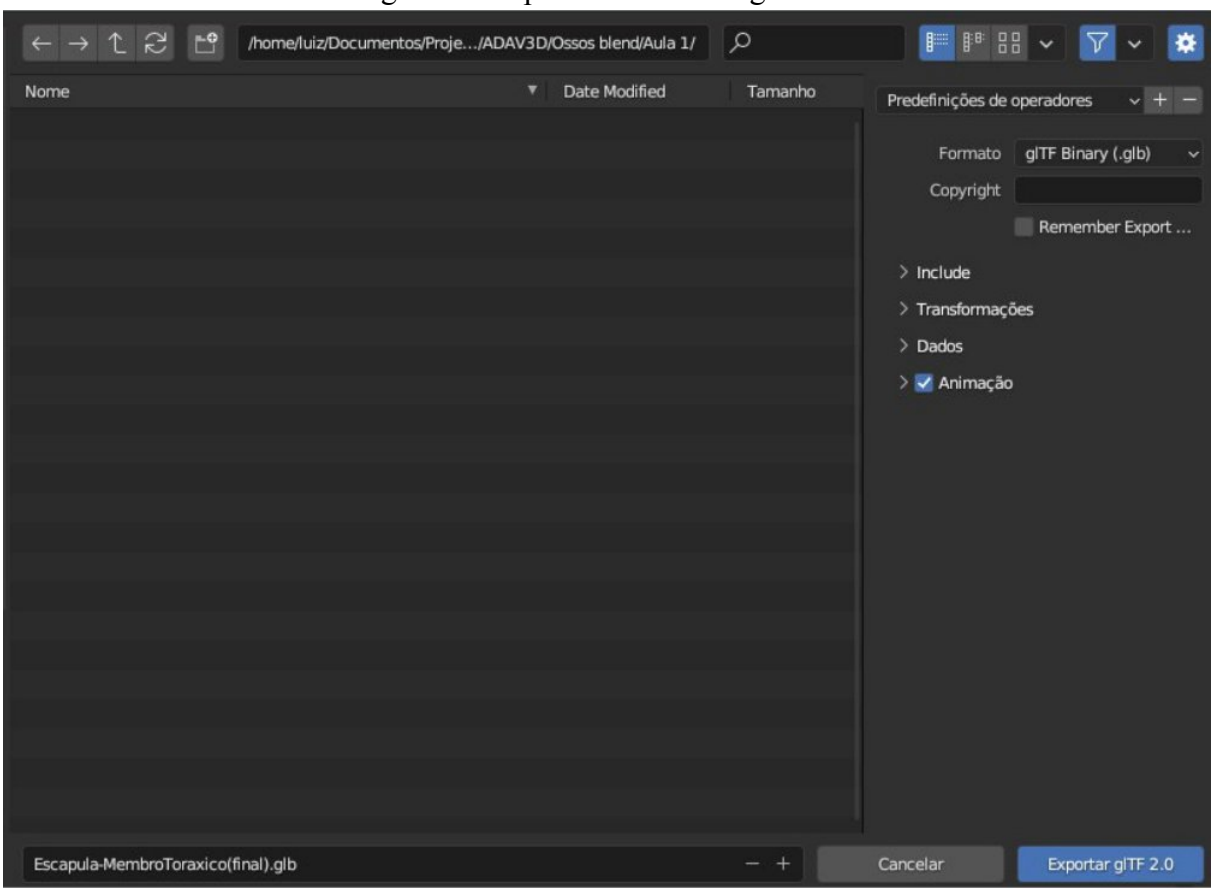
O renderizador 3D desenvolvido não é capaz de renderizar um arquivo .blend, no manual do Blender 3.6 (BLENDER, 2023), é explicado que glTF é usado para transmissão e carregamento de modelos 3D na web e aplicações nativas. glTF reduz o tamanho de modelos 3D e o tempo de execução necessário para descompactar e renderizar estes modelos.

Na documentação do Three.js (2023), se diz o seguinte sobre carregar modelos 3D: “Onde possível, recomendamos usar glTF (Formato de Transmissão GL). Ambos formatos *.glb* e *.gltf* são bem suportados. Isso é porquê glTF é focado em em entrega de recursos em tempo de execução, é compacto para transmitir e carregar. Inclui meshes, materiais, esqueletos...”

Devido à sua alta compatibilidade com WebGL e até mesmo várias engines 3D, assim como arquivo único, compactado e de baixo peso, o novo renderizador possui suporte a modelos em formato glTF 2.0, como um único binário **.glb**, simplicando o processo de cadastro do osso no sistema, assim como passando todas as informações e modelos necessários por um único arquivo binário, de baixo peso para o banco de dados, pesando de 100 a 200kb.

Como Blender possui suporte a glTF 2.0, é um processo simples de se fazer, selecionando arquivo/exportar/glTF 2.0, na janela que abrir, em formato, selecionando .glb e confirmando. Janela representada na Figura 8.

Figura 8 - Exportando modelo glTF 2.0



Fonte: Do Autor (2023)

4.4.3. WebGL dinâmico e responsivo com R3F

Sabendo da estrutura dos modelos que o projeto possui e como era o funcionamento de sua aplicação 3D, houve cinco desafios para esta etapa:

1. Criar um novo renderizador 3D, preferencialmente um integrado com ReactJS e/ou Javascript, foram as tecnologias em que o novo front-end foi desenvolvido.
2. Reproduzir todas as funcionalidades da aplicação anterior , vistos em RF09-01~05.
3. Fazer uma aplicação totalmente dinâmica, que seja capaz de pegar as informações do osso, junto do seu modelo, direto da API (RF09-06).
4. Adicionar compatibilidade com dispositivos móveis (RNF05)
5. Boa performance, capaz de rodar em dispositivos móveis fracos, mas que ainda possuam navegadores com suporte a WebGL (RNF06)

Unity por ter um renderizador semi pronto poderia ser uma opção, porém suas aplicações são “isoladas” do resto do sistema, e não possui suporte oficial para aparelhos

mobile, tornando o retrabalho em adaptar a aplicação tão grande quanto desenvolver uma nova por completo.

Seguindo este caminho, buscou-se uma biblioteca Javascript muito famosa chamada Threejs, com documentação extensiva e diversos cursos disponíveis, suas cenas são escritas e manipuladas em javascript puro e não possui as limitações de hardwares suportados em WebGL que Unity possue.

Por sua forte integração com React, foi utilizado o React Three Fiber, ou R3F, que é uma integração completa de Threejs em React. A biblioteca conta também com vários componentes com abstrações inteligentes, pré-definições que fazem sentido para maior parte das aplicações, muita documentação e bibliotecas de expansão, como React Three Drei, que proporciona mais formas de controle e adiciona escolha manual de câmeras e seu posicionamento de forma mais intuitiva, permitindo criar uma aplicação 3D com Threejs, extremamente performática, em curto prazo de desenvolvimento.

Sua alta integração no ambiente front-end permitiu o desenvolvimento de aplicações que faz uso inteligente de uma cena 3D em meio a páginas e menus em HTML e CSS, assim, não mais que o necessário precisa ser renderizado na cena da nova aplicação 3D, ou seja, apenas o modelo do osso e seus acidentes são elementos presentes na cena 3D.

Para auxílio da criação da cena, foram utilizados diversos componentes de Drei, como *Enviroment*, que cria um mapa de ambiente cúbico global, podendo incluir planos de fundo pré-definidos ou a partir de arquivos customizados, além de efeito “borrar” o fundo e iluminação.

Foi utilizado a iluminação de pré-definição *sunset*, sem um plano de fundo, em conjunto de uma luz ambiente, de intensidade 0.5 do R3F. Isso garante boa iluminação para o osso, em todos os ângulos, mas ainda preservando seus detalhes, mais facilmente visualizados através de “sombras” produzidas na renderização.

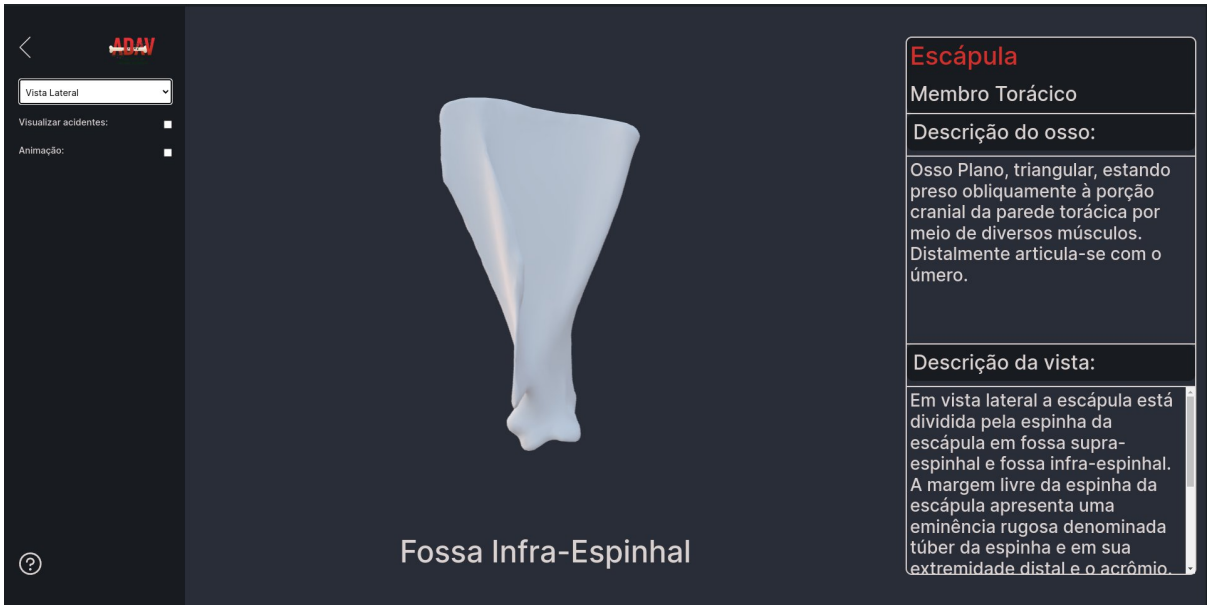
Foi usado também *OrbitControls* de Drei, onde todos os controles reagem à câmera padrão, o que da controle total da cena, assim como parâmetros para sua configuração, como alvo de nossa câmera na cena, em coordenadas cartesianas (X, Y, Z) [0, 0, 0], focando a câmera sempre ao centro da cena. Também controlado por parâmetros, as distâncias mínimas e máximas do alvo, até mesmo rotação automática e sua velocidade, todos utilizados na aplicação final do projeto.

Por fim, a câmera de perspectiva, também de Drei, escolhida por podermos usar os efeitos de *zoom,* aproximação e afastamento do modelo, graças as propriedades de

câmeras perspectivas. É também controlado seu campo de visão (FOV) em valor 75 e sua posição, extremamente importante para funcionalidade de vistas, onde uma seleção de vista faz atualização deste parâmetro.

Na figura 9, é representado a aplicação final para usuário de computadores de mesa:

Figura 9 - Aplicação ADAV 3D Desktop



Fonte: Do Autor (2023)

Para telas maiores, ou *Desktops*, a aplicação tem a cena 3D no centro, o modelo do osso é colorido com uma cor branca, igual a já utilizada no projeto anterior para os ossos, *RGB*(0.8, 0.8, 0.8). Seus acidentes são coloridos de cores aleatórias e iniciam-se invisíveis, o usuário possui total controle da câmera em 360° ao redor do centro do osso ao arrastar a mesma com o dedo ou ponteiro do mouse, assim como controle de distância (gesto de pinça ou roda do mouse), permitindo fazer o *zoom*[18] do modelo na cena.

Em sua esquerda, pode ser observado os controles da cena, como o seletor de vista, uma caixa de seleção de visualização dos acidentes, os tornando visíveis na cena, e uma caixa de seleção para animação, sendo que a aplicação inicia com cena animada da câmera girando ao redor do modelo do osso.

Já na direita, encontra-se o painel de informações, nele pode ser avistado, em ordem, nome do osso, um subtítulo (opcional), descrição geral do osso e descrição da vista atualmente selecionada. Caso a descrição exceda o tamanho da caixa de descrição, é apresentado uma barra

[18] Aproximar ou afastar câmera de uma cena, modelo ou objeto.

de rolagem, permitindo o usuário a ter acesso a toda informação, sem quebrar a aplicação.

O renderizador 3D faz parte do *Canvas* da biblioteca R3F, que já possui reatividade integrada na sua cena, que se reajusta de acordo com a altura da sua *viewport*. Isso facilitou o desenvolvimento de uma versão mobile para o sistema, assim como a criação de pré-visualizações nos diversos *modais* de cadastro e edição. Replicando este mesmo comportamento para o HTML que sobrepõe a cena 3D e imprime o nome do acidente, ou seja, se adaptando a altura da sua tela de renderização, foi desenvolvido rapidamente uma tela mobile para a aplicação, onde toda sua tela se trata apenas do novo renderizador 3D.

Por isso, para criar uma aplicação mobile sem comprometer suas funcionalidades, foi desenvolvido um menu em *gaveta*, vindo de cima, mantendo as cores escolhidas para aplicação e de fundo com efeito transparente, deixando o osso sempre visível. A aplicação Mobile pode ser vista na Figura 10:

Figura 10 - Aplicação ADAV 3D Mobile - vertical

Fonte: Do Autor (2023)

O menu de informações e controle pode ser acessado ao tocar no botão em formato de seta, no topo da aplicação. O menu possui um design simples e limpo, com seu título e subtítulo no topo, junto aos mesmos controles da aplicação presentes na sua versão *Desktop*. As descrições gerais e das vistas podem ser encontradas no mesmo local, sendo alternadas através dos botões “Informações” e “Descrição Vista”. Assim como a aplicação *Desktop*, este menu é pensado para casos de descrições maiores que sua área de visualização, permitindo arrastar o menu para ter acesso a todo seu conteúdo, isso também viabiliza seu uso na horizontal, ideal para ossos mais “compridos” ou telas menores fixas em horizontal, representado na Figura 11:

Figura 11 - Aplicação ADAV 3D Mobile - horizontal

Fonte: Do Autor (2023)

4.5. Modelagem do banco de dados

Uma aplicação dinâmica necessita de um banco de dados onde suas informações são armazenadas e consumidas. Para tal, é necessário um bom planejamento de um banco de dados, levando em consideração as necessidades da aplicação.

Além de criar um modelo capaz de relacionar os ossos aos seus respectivos animais e/ou categorias, é necessário lembrar que o sistema necessita passar por uma revisão manual por um docente responsável, e por isso foi feito o uso de duas estratégias diferentes, uma para novas submissões, outra para edições.

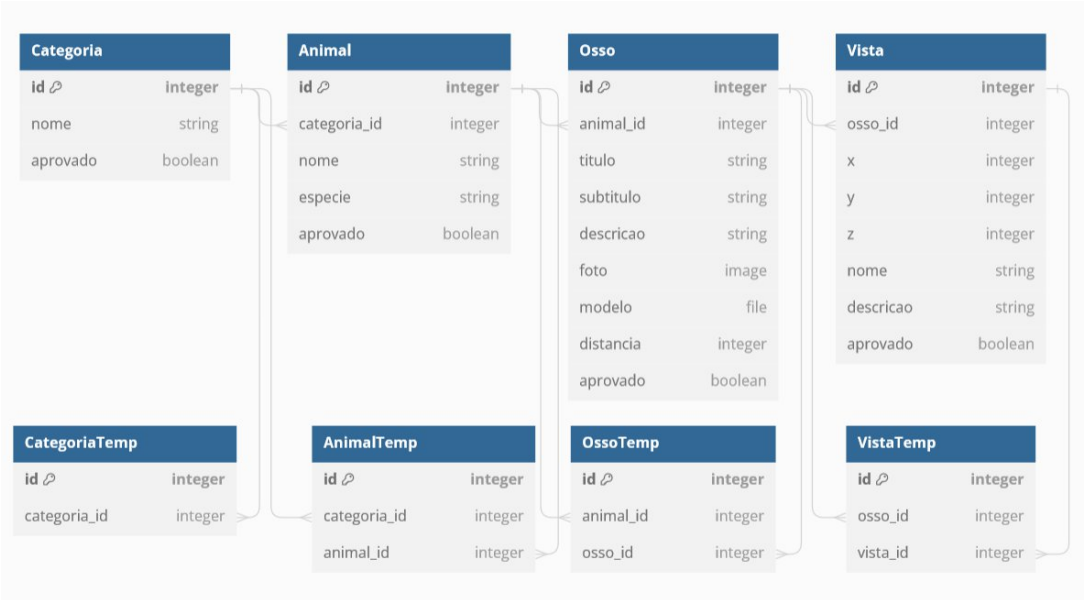
Ao fazer um novo cadastro, de qualquer uma das tabelas, seu cadastro é armazenado

diretamente na tabela principal, com seu campo “aprovado” inicialmente em “Falso”, apenas trocando para “Verdadeiro” ao ser aprovado, passando a ser selecionável na tela de listagem de ossos. Por outro lado, edições são mudanças de uma tabela, ainda não aprovadas, sendo necessário armazená-las em tabelas externas, contendo os mesmos campos, que foram nomeadas de tabelas temporárias, ou “Temp”.

Na figura 12, pode-se observar um diagrama simplificado da modelagem feita. Isso significa ausência dos campos “autor” e “data”, representando usuário e data de submissão, presentes em todas as tabelas, campo *change*[19], presente na tabela OssoTemp para descrição de mudança, e por fim, a tabela de Usuário, que foi utilizada padrão do Django, e possui ligação com todas as tabelas, através do campo “autor”.

Se abstêm da ilustração também a duplicação de campos entre uma tabela e sua respectiva tabela temporária, deixando o modelo o mais limpo e simples possível para análise.

Figura 12 - Modelagem simplificada



Fonte: Do Autor (2023)

A modelagem do banco de dados é feita através do framework utilizando para desenvolvimento do back-end e API do sistema, feita através de código em python usando estrutura de classes, com seus atributos representando os campos de uma tabela. As ligações entre tabelas também são caracterizadas por campos que recebem outras classes de modelagem, juntamente da sua configuração de restrição de integridade referencial, escolhendo qual ação

[19] Palavra em inglês com significado de “mudança”

deve ser tomada para caso de campo referenciado (tabela em ligação) seja deletada do sistema.

No fim de sua modelagem, essas tabelas são migradas pelo *framework* de desenvolvimento para um único arquivo SQLite, onde é armazenado todas as informações do sistema, exceto arquivos (modelos e fotos), que são armazenados em diretórios externos configurados pelo desenvolvedor.

Na figura 13, é exemplificado como é feita essa modelagem, usando de exemplo a classe que representa a tabela “Osso” do sistema.

Figura 13 - Modelagem de tabela em classe

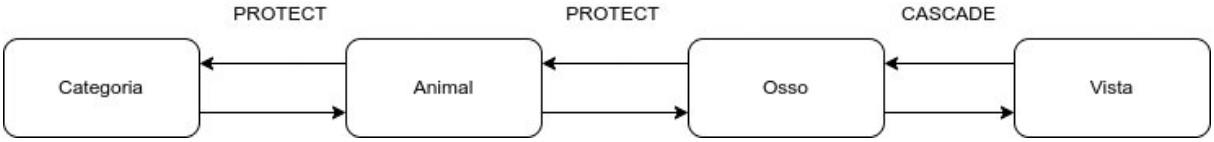
```
class Osso(models.Model):
    animal = models.ForeignKey(
        Animal,
        on_delete=models.PROTECT
    )
    titulo = models.CharField(max_length=50)
    subtitulo = models.CharField(max_length=50, default='')
    descricao = models.CharField(max_length=1024)
    distancia = models.IntegerField(default=500)
    foto = models.ImageField(upload_to=upload_imagem)
    modelo = models.FileField(upload_to=upload_modelo)
    aprovado = models.BooleanField(default=False)
    autor = models.ForeignKey(User, on_delete=models.PROTECT)
    data = models.DateField()

    def __str__(self):
        return str(self.titulo)
```

Fonte: Do Autor (2023)

A relação entre tabelas e sua configuração de restrição de integridade referencial pode ser vista na Figura 14:

Figura 14 - Configuração de integridade referencial



Fonte: Do Autor (2023)

Como visto na figura 14, todas as relações são protegidas, ou seja, caso o campo a ser removido possua relação com outra tabela, ele deve ser protegido e a remoção não acontecerá. Isso acontece para proteger o banco de dados do sistema de ataques ou erros humanos, onde a remoção de uma categoria, por exemplo, resultaria na remoção de N animais pertencentes à categoria, assim como mais M ossos por animal que foi removido, oque seria um ponto de falha crítico em sua modelagem, possibilitando perda imensa de dados em um único comando.

Por isso, para remover um Animal, por exemplo, todos seus Ossos devem ser removidos previamente, e uma categoria só pode ser removida após a remoção de todos os seus animais. Única exceção para este comportamento no sistema, é a relação de Ossos e suas vistas, apesar de serem tabelas diferentes e campos separados, na visão do usuário final, Vista é uma extensão do Osso, pertencente a ele como seu nome ou descrição, e portanto, não deve atrapalhar operações de remoção de Ossos dos administradores do sistema, e são removidas juntamente ao seu Osso referenciado nestes casos.

Esse comportamento é diferente em suas tabelas temporárias, uma tabela temporária tem como único propósito armazenar dados de edições até serem aprovados ou não, e não devem interferir ou "proteger" outras tabelas do sistema. O que significa na prática, que caso haja um osso X cadastrado no sistema, e uma edição deste mesmo osso X em sua tabela temporária, ao um administrador remover Osso X do sistema, todas suas edições aguardando análise também serão removidas, configuradas em cascata.

4.6. Desenvolvimento e configuração de uma API

O manuseamento do banco de dados modelado se faz através de uma API desenvolvida em Django Rest Framework, que é um framework python para desenvolvimento completo de APIs, assim como bancos de dados. Api desenvolvida para o sistema foi separada em quatro aplicações:

- **api** - Aplicação principal, contém os arquivos de configuração principais da API, junto das URLs raízes do sistema (que permitem acesso às outras aplicações).
- **auth –** Responsável pelas rotas e lógicas de autenticação do sistema.
- **adav –** Detém modelagem, rotas de acesso e lógica das categorias e animais, servindo como "interfaces" para outras aplicações do ADAV, possui arquivos de controle de permissões e funções comumente usadas pelo sistema.

- **ossos –** Aplicação membro de “adav”. Possui modelagem, rotas e lógicas sobre os ossos e vistas.

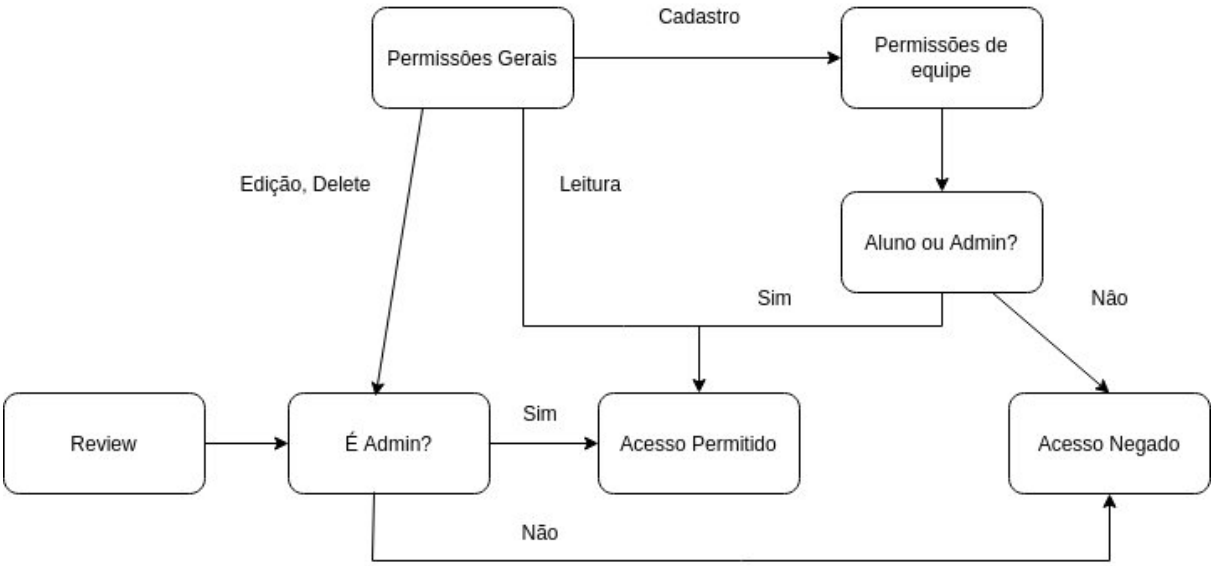
Sua lógica e rotas de autenticação foram desenvolvidos utilizando a biblioteca simplejwt, automatizando boa parte de seu desenvolvimento. A api incorpora a biblioteca drf-spectacular para geração instantânea de documentação, através de uma página “swagger”, que documenta todas as rotas presentes e seus campos.

A API foi desenvolvida com 3 tipos de usuários, sendo eles:

- **Não autenticado** – permissões “seguras”, como receber informações das tabelas (listagem de ossos, ou visão mais detalhada de um osso, por exemplo. Apenas leitura)
- **Staff** – usuário aluno, com permissões de criar novas submissões e submeter edições
- **Admin** – controle total da api e banco de dados, acesso a página de admin da api, permissões de *review*[20] das submissões dos alunos.

O diagrama da figura 15 representa as permissões descritas por usuário na lista anterior, onde rotas de tabelas principais são protegidas por “Permissões Gerais” e rotas de tabelas temporárias são protegidas por “Permissões de equipe”:

Figura 15 - Configuração de permissões da API

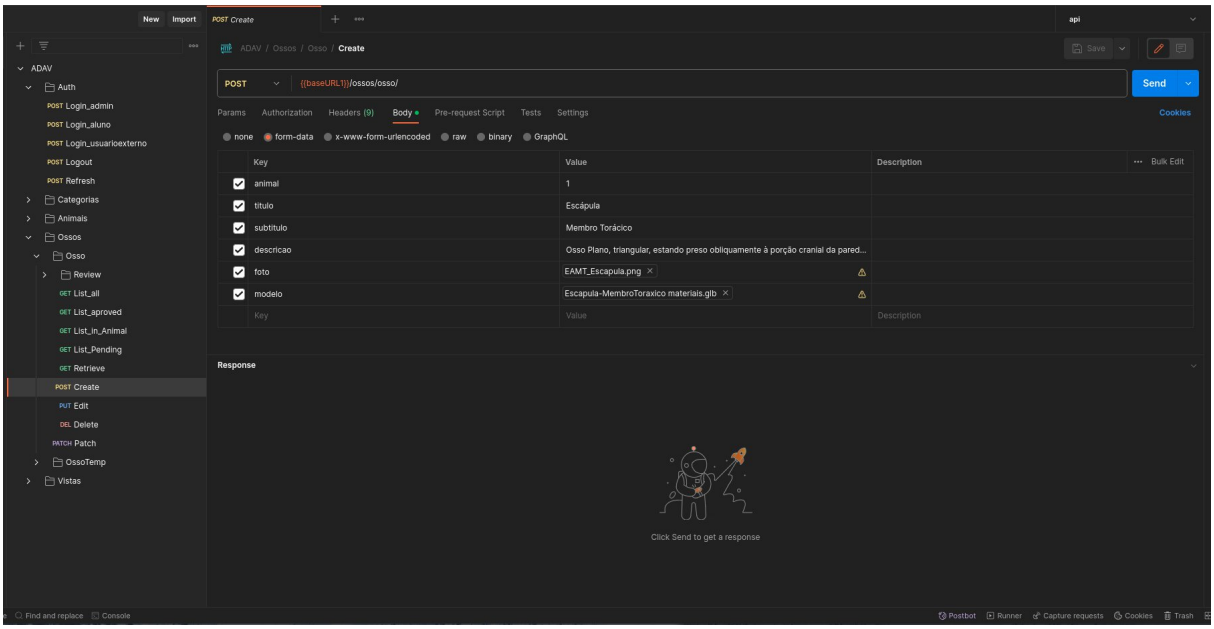


Fonte: Do Autor (2023)

[20] Palavra em inglês com significado de “análise”.

Foi feita também uma documentação manual da API e seu funcionamento utilizando a ferramenta *Postman*[21], permitindo a equipe compartilhar as rotas e exemplos de uso, assim como acelerar o processo de testes destas rotas. Na figura 16, pode ser visto a configuração desta documentação, feita em pastas representando as tabelas a serem acessadas, assim como suas possíveis rotas e ações, exemplificadas para uso da equipe de desenvolvimento:

Figura 16 - Documentação de rotas com exemplos



Fonte: Do Autor (2023)

4.7. Novo front-end, mesma identidade visual

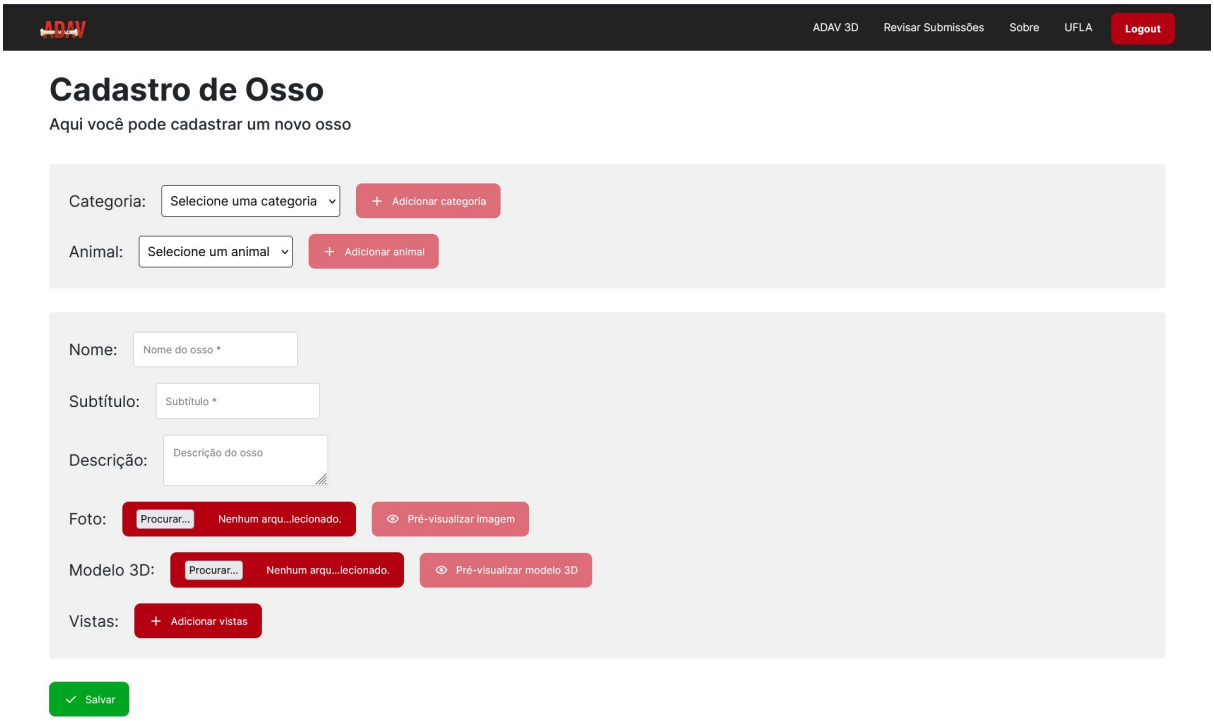
Apesar de suas limitações técnicas, ADAV possui uma boa identidade visual, e em sua reconstrução, houve foco em preservá-la, com pequenas alterações que tornasse o sistema mais intuitivo e moderno. O novo front-end foi construído totalmente em React, com Vite e JavaScript, Redux Toolkit.

Seu maior desafio veio em forma das diversas páginas dinâmicas, necessitando de readequação de sua API diversas vezes para atender as necessidades do design de experiência de usuário planejado pela equipe para sua interface.

[21] POSTMAN, Postman, 2023. Disponível em: https://www.postman.com/ Acesso em: 08/11/2023.

Em destaque, vem suas novas telas e componentes, necessários para o funcionamento do sistema, como página de cadastro e seus modais, página de *reviews* com modais próprios, e por fim, modais de edição para todas as tabelas do projeto.

Figura 17 - Página cadastro de osso



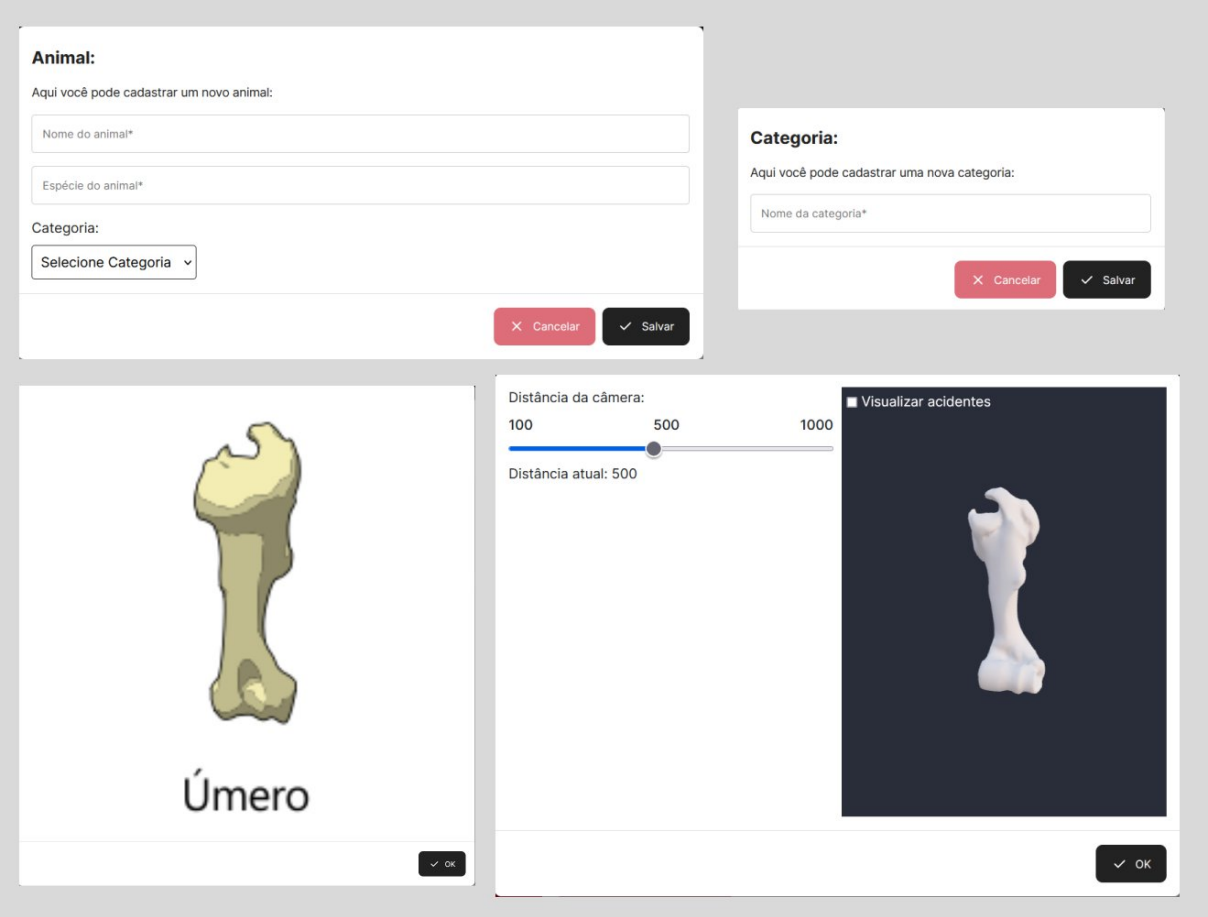
Fonte: Do Autor (2023)

Por meio da página apresentada na figura 17, a equipe consegue fazer o cadastro de um novo osso no sistema, porém é nesta mesma página que se cadastra novas categorias e animais, ao selecionar uma categoria/animal, caso vá cadastrar um osso de categoria ainda não existente, o discente pode fazer o cadastro ali mesmo, pressionando “adicionar categoria/animal”, que abrirá um dos modais respectivos da operação. Após um cadastro, o sistema atualiza automaticamente a listagem de categorias e de animais pertencentes a categoria selecionada.

Tabela Osso possui três campos de texto, sendo subtítulo opcional, dois seletores de arquivos, um para a imagem de capa do osso, mostrada na listagem de ossos em ADAV 3D, e outro para seleção do modelo 3D do osso, arquivo único .glb, como demonstrado na seção 3.4.2. Ambos possuem modais de pré-visualização, e no modal do modelo 3D, mais um campo opcional, sendo um controle deslizante para distância padrão da câmera em relação ao osso. Isso é importante para garantir que osso tem distância adequada para a seleção de vistas, assim como para sua renderização inicial, antes de qualquer zoom manual do usuário.

Os modais presentes na página de cadastro podem ser vistos na Figura 18:

Figura 18 - Modais de cadastro



Fonte: Do Autor (2023)

Por fim, é feito uso do modal de seleção de vistas, que depende da seleção de ossos, e reage a alterações na distância da câmera (todas as vistas possuem a mesma distância do centro da cena, ponto [0,0,0], onde deve estar também, o centro do osso), em todas as vistas, a câmera estará apontando ao centro da cena.

Como pode ser observado em Vistas Ortográficas pela UFRGS (2016), existem dois sistemas de projeção de vistas, americano e europeu, sendo sua única diferença a posição do plano de projeção em relação ao objeto e observador, porém, em ambos sistemas, as seis vistas ortográficas padrões permanecem sendo as mesmas:

- **VA:** Vista frontal anterior
- **VP:** Vista Frontal Posterior
- **VS:** Vista Superior
- **VI:** Vista inferior
- **VLE:** Vista lateral esquerda
- **VLD:** Vista lateral direita

O discente tem disponível um seletor com seis vistas selecionáveis, podendo selecionar de zero a seis vistas em seu cadastro, onde pode nomear cada vista selecionada e adicionar uma descrição.

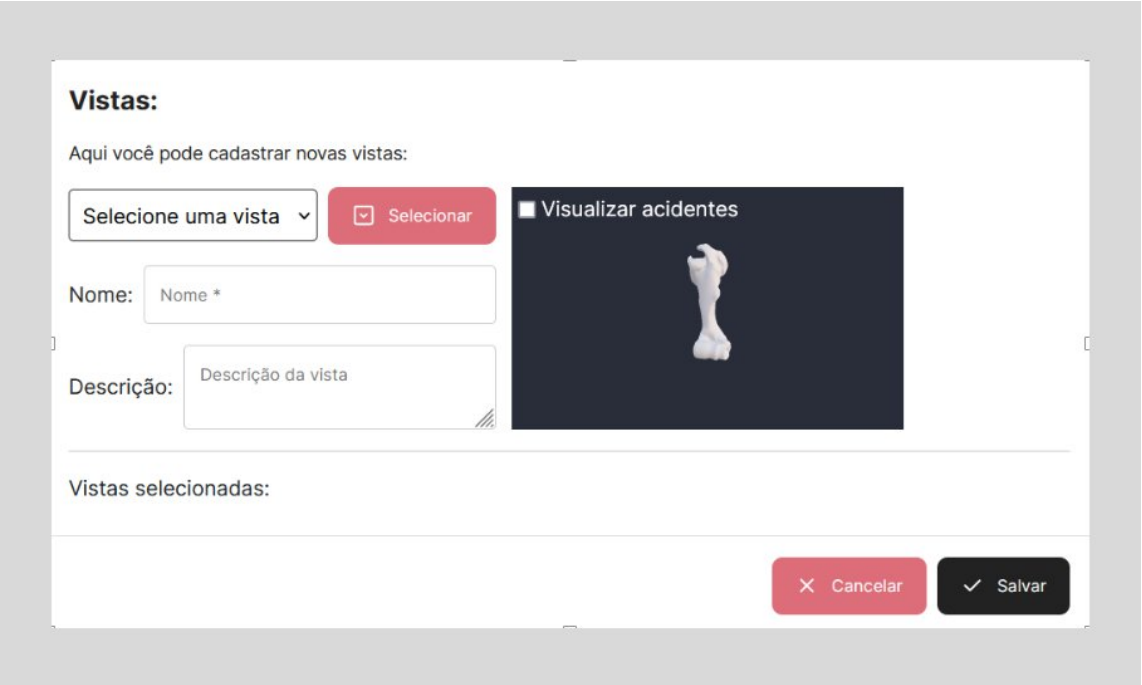
Tabela 1 - Vistas ortográficas da aplicação

Nome provisório	X	Y	Z
Vista Frontal Anterior (VA)	0	0	Distância
Vista Frontal Posterior (VP)	0	0	-Distância
Vista Superior (VS)	0	Distância	0
Vista Inferior (VI)	0	-Distância	0
Vista Lateral Esquerda (VLE)	-Distância	0	0
Vista Lateral Direita (VLD)	Distância	0	0

Fonte: Do Autor (2023)

Representadas pela Tabela 1, “Distância” representa a distância escolhida pelo discente no modal do modelo, além disso, existem seis opções de vistas no seletor de vistas, simplificados para “frente”, “trás”, “cima”, “baixo”, “esquerda” e “direita”. Ao escolher uma das vistas e preencher seus campos de nome e descrição, basta pressionar o botão “selecionar” para adicionar a vista à lista de vistas a serem cadastradas. Uma mesma vista não pode ser selecionada múltiplas vezes, e no fim, com todas as vistas do osso selecionadas, apenas pressionar “salvar”. Modal de seleção de vistas encontra-se representado pela Figura 19:

Figura 19 - Modal cadastro vistas

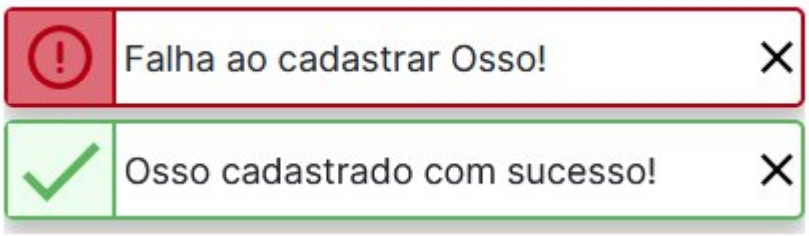


Fonte: Do Autor (2023)

Desta forma, o aluno pode de forma intuitiva selecionar o modelo, ajustar a câmera da cena, assim como selecionar suas vistas, todos numa mesma página de cadastro, como se fosse um único formulário.

Por fim, ao selecionar “Salvar” na página de cadastro, a requisição de cadastro é feita e aluno é alertado imediatamente caso haja sucesso ou não em seu cadastro, através de alertas desenvolvidos para o sistema, representados na figura 20.

Figura 20 - Alertas



Fonte: Do Autor (2023)

Porém, analisando pela modelagem vista na figura 12, nota-se que vistas e ossos não são parte de uma mesma tabela, e ainda mais, vistas só podem ser cadastradas para um osso

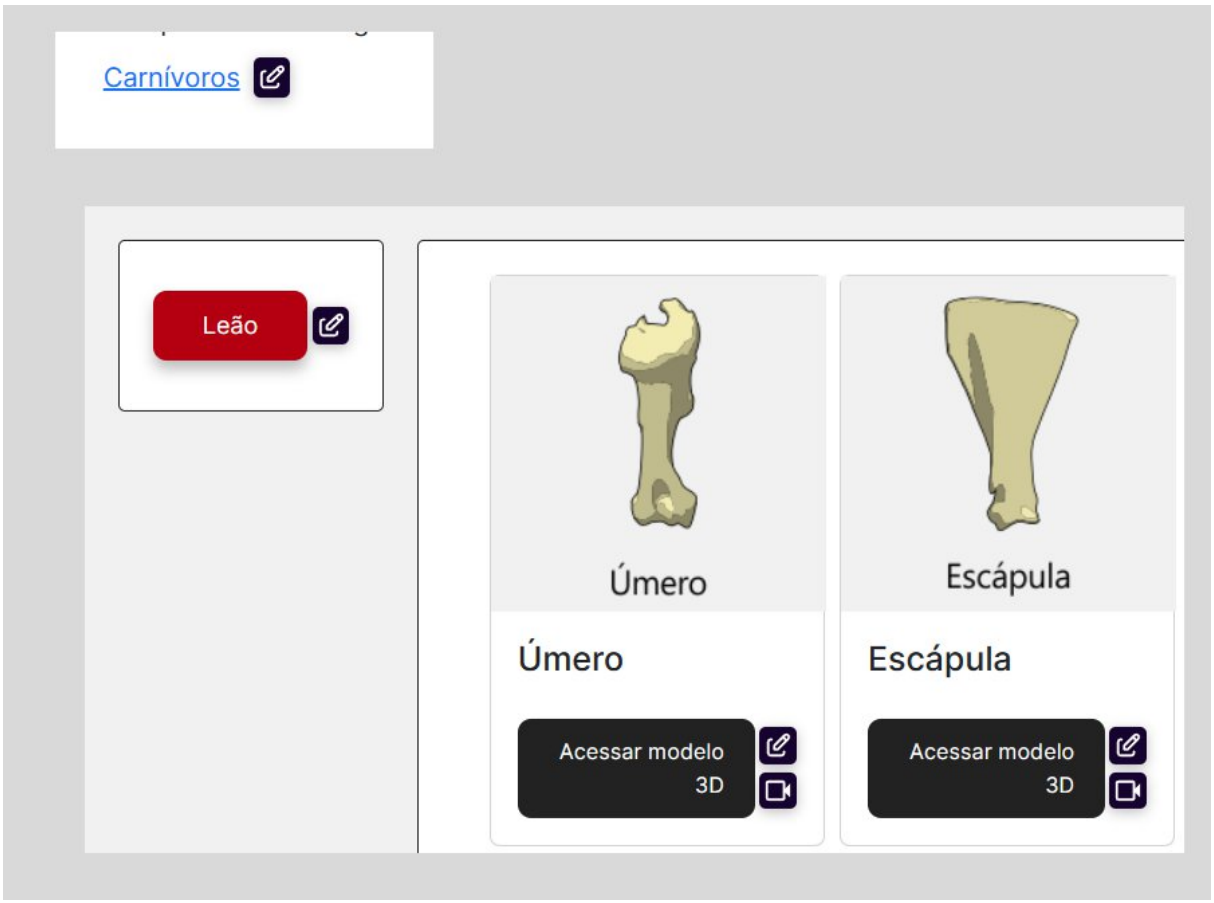
que já foi previamente cadastrado no sistema, já que possuem ligação com osso e necessitam de sua chave primária.

Para contornar este problema, são feitas duas requisições no fim do cadastro, a primeira é o cadastro do osso, com todas suas informações e arquivos. Subsequentemente, caso a primeira requisição seja efetivada com sucesso, API responde com as informações do osso cadastrado, incluindo sua chave primária, que é então usada numa segunda requisição, junto de uma lista de vistas, as quais são cadastradas uma por uma no banco de dados.

É possível que haja erros do autor da submissão, e por isso, ADAV 3D também possui a funcionalidade de submeter edições em categorias, animais, ossos e vistas já cadastrados.

Um discente membro da equipe pode encontrar um botão com ícone de um lápis numa folha, representando a edição, ao lado das listagens do respectivo campos no ADAV 3D, exceto o botão de edição de vistas, que para diferencia-lo do editor do osso, recebe um ícone de uma câmera, representados na figura 21:

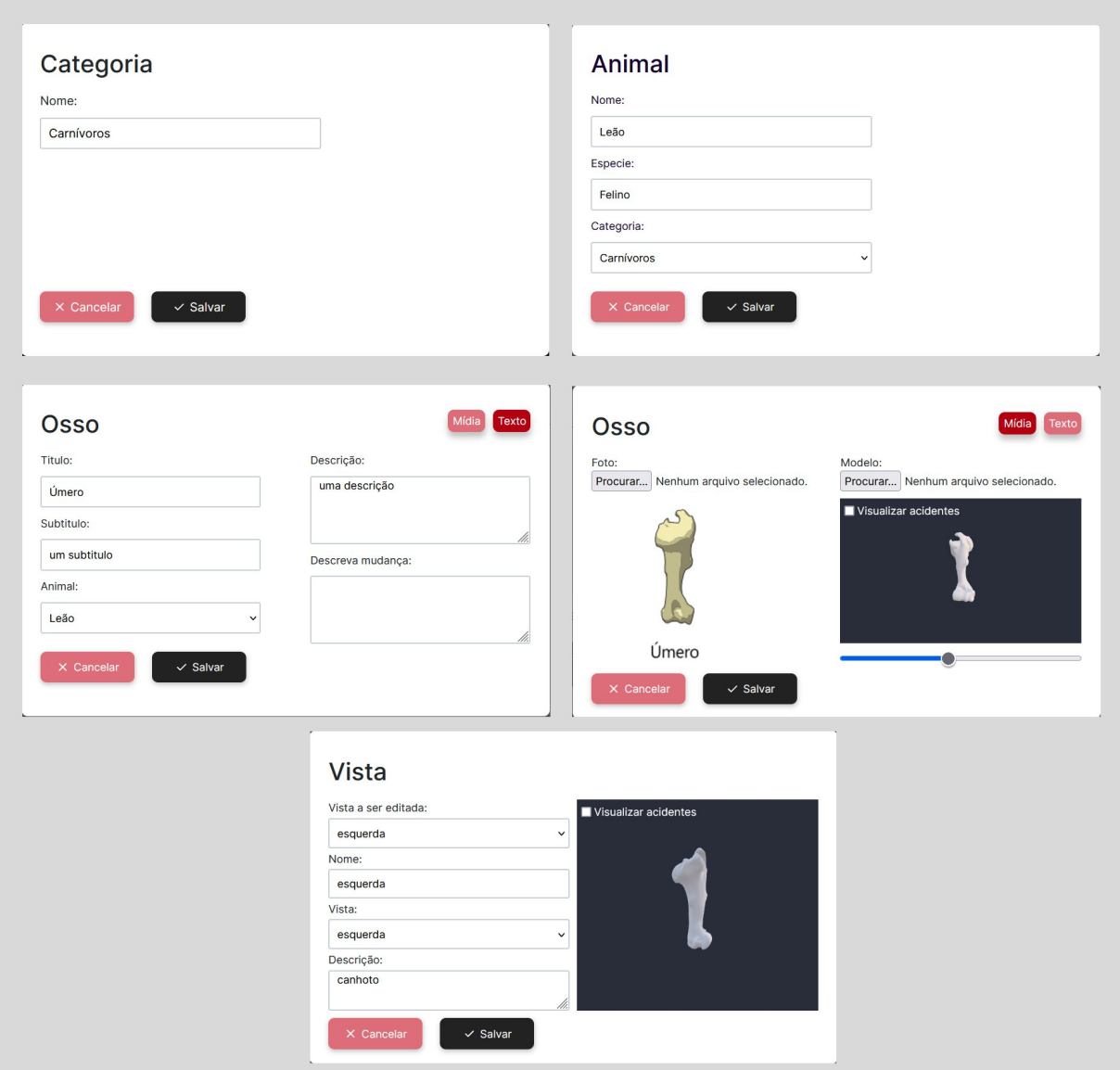
Figura 21 - Acesso a modais de edição



Fonte: Do Autor (2023)

Os modais acessíveis através destes botões, são modais de “baixo privilégio”, disponíveis para qualquer membro autenticado da equipe do projeto, é nestes modais que são feitas as edições temporárias do sistema, que na verdade são cadastros, contendo as mesmas informações da tupla editada (exemplo: informações do Leão) , apenas com as mudanças feitas pelo discente (atualização de nome e/ou espécie). Estes modais podem ser vistos na figura 22:

Figura 22 - Modais de edição



Fonte: Do Autor (2023)

Os modais da figura 22 foram desenvolvidos para serem simples e intuitivos, contendo todos os campos de uma determinada tabela. Destes modais, destaca-se o modal de ossos, pois é dividido em duas seções, "mídia" e "texto", representando os campos a serem editados em cada seção, porém ambas parte de um mesmo formulário, e submetidas juntas, em casos de múltiplas mudanças. Além disso, é o único modal que exige a declaração do discente de qual foi sua mudança, essa exigência de uma descrição da mudança vem da complexidade de análise destas edições, como por exemplo: Um discente atualizou o nome de um dos acidentes do modelo do osso, isso demandaria excesso trabalho do orientador para encontrar a mudança na submissão caso aluno não declare sua mudança.

Para o modal de vistas, o aluno pode selecionar qual vista irá atualizar em "Vista a ser editada", podendo atualizar qualquer um dos seus campos, sendo seu nome, vista ortográfica e sua descrição, sendo uma vista atualizada por vez por submissão de edição.

Relações entre tabelas também podem ser atualizadas, como exemplo, "a que categoria pertence aquele animal", através de um simples selecionador, com os campos disponíveis para atualização (como categorias, no exemplo dado).

Todas as submissões do sistema, cadastro ou edições, são então armazenadas para avaliação, que pode ser feita na página de "Revisar submissões" do orientador do projeto.

Nesta página o orientador tem as submissões organizadas por tabelas, e tipo de submissão, com a exceção das vistas, por serem vistas específicas de um osso, encontram-se em *dropdowns* na mesma seção das submissões do osso em si.

Cada *dropdown*[22] tem um contador, avisando quantas submissões existem para avaliação do orientador, podendo ser visualizadas ao selecionar a seta no canto direito do mesmo.

Na listagem de submissões, essas submissões são separadas em até dois cartões, sendo um cartão vermelho, com as informações atuais daquele campo, e um verde, representando os novos valores (em caso de cadastros novos, apenas o cartão verde com a nova submissão)

O orientador ao fazer sua revisão possui três opções, como aceitar a submissão, recusá-la, ou até mesmo, editá-la antes de avalia-la, caso ache que precisa apenas de uma mudança menor (um erro gramatical, por exemplo).

O botão de edição por seção abre um modal similar aos modais representados aos modais da figura 22, diferenciado no modal de vista, já que são avaliadas e editadas individualmente, assim como seu comportamento, que não submetem um campo novo em uma

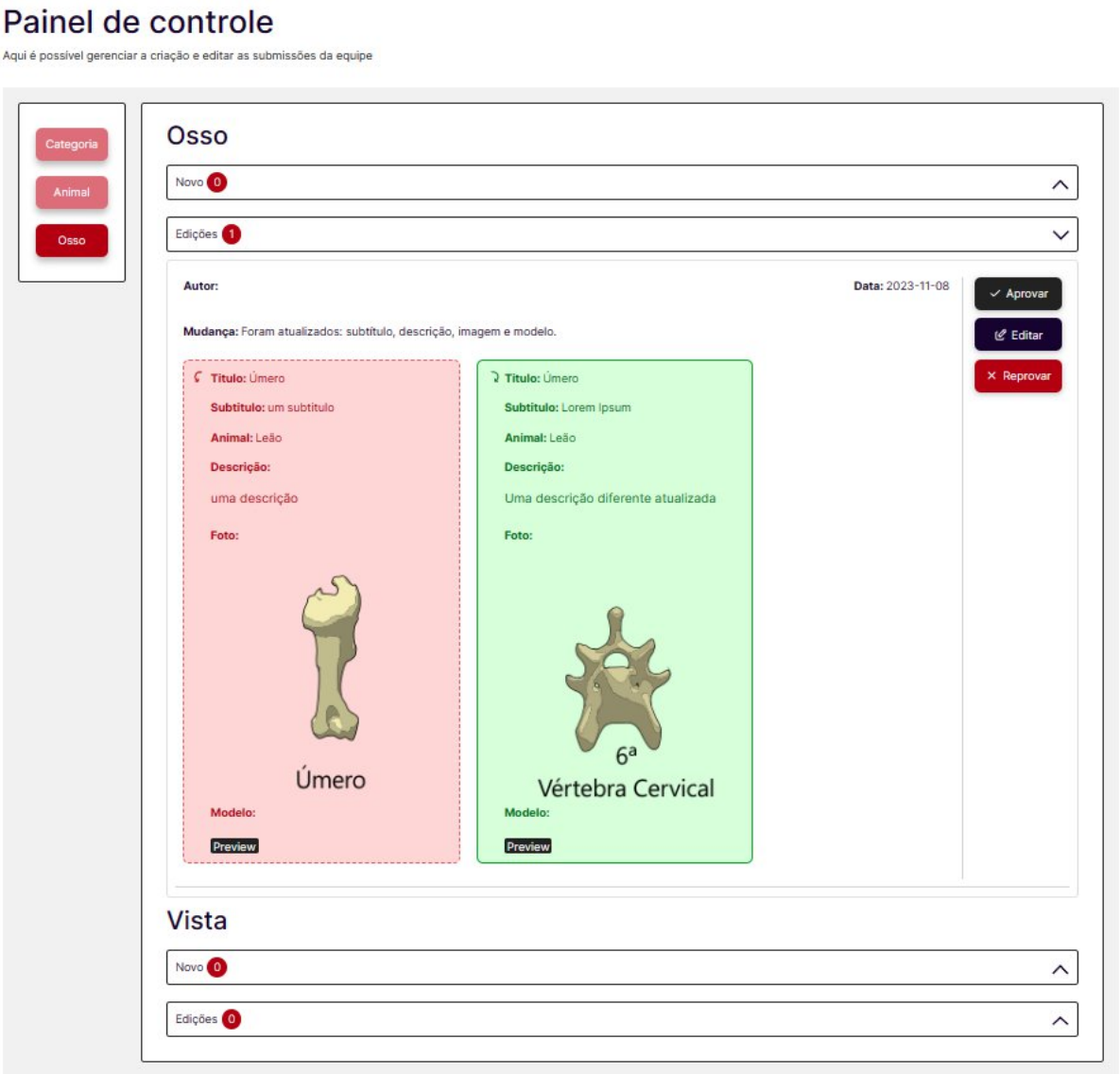
[22] Componente de conteúdo escondido, até seleção do usuário para visualização

tabela temporária para avaliação, e sim, atualizam a submissão imediatamente.

No caso de submissões de osso, o modelo só pode ser avistado após selecionar a opção *preview*[23] dos cartões, assim como também nas submissões de vistas.

Isso é feito por razões de performance, onde uma grande quantidade de submissões pode ser excessivo para máquinas mais simples (20 submissões de edição ossos seriam 40 aplicações 3D ao mesmo tempo, por exemplo). A página está representada na Figura 23, com um exemplo de edição de osso listada:

Figura 23 - Painel de controle admin



Fonte: Do Autor (2023)

[23] Palavra em inglês, significado de “pré-visualização”

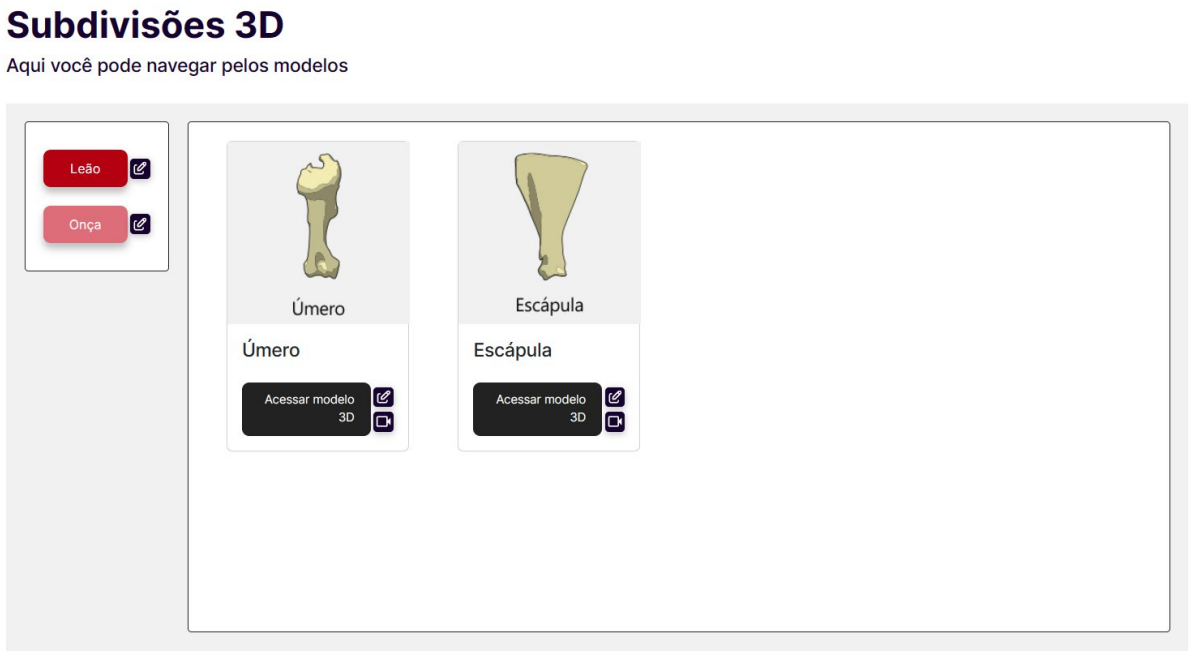
Com todos cadastros, edições e análises concluídos, a aplicação estará pronta para uso do usuário final, que pode acessar os ossos na seção “ADAV 3D” na barra de navegação. Na página direcionada por “ADAV 3D”, pode ser visto uma descrição breve do que se trata o projeto, e uma lista de categorias aprovadas para serem selecionadas. Essa página pode ser vista na Figura 24:

Figura 24 - Listagem de Categorias

Fonte: Do Autor (2023)

Ao selecionar uma categoria, o usuário é direcionado para a listagem de ossos, filtrados pelo animal selecionado. O usuário pode trocar de animal e atualizar a lista, e selecionar um dos ossos presentes para acessar seu modelo 3D. Esta página pode ser vista na Figura 25:

Figura 25 - Listagem dos modelos



Fonte: Do Autor (2023)

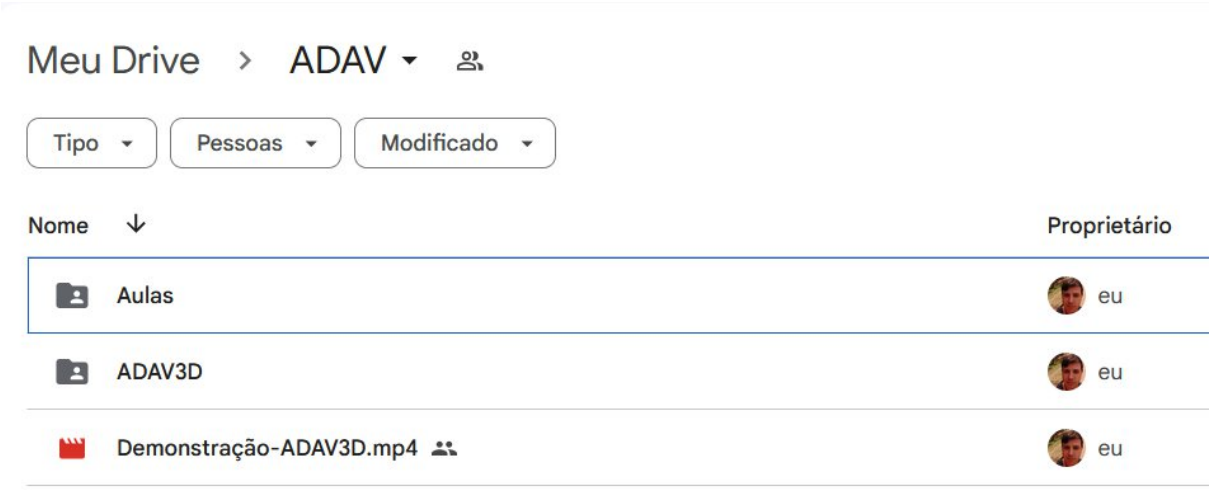
4.8. Gravação de guias de uso do sistema

Para uso do sistema, é necessária a documentação de seu funcionamento, especialmente por parte da equipe responsável pelo cadastro de ossos, já que apesar de poucas mudanças, os modelos devem se adequar aos padrões do sistema para correto funcionamento.

É relevante também para futuras expansões do sistema, pois não é do conhecimento do autor deste trabalho o processo de modelagem de ossos, portanto é de suma importância documentar como os modelos devem ser adequados para futuras expansões do atlas.

Por isso, foi feita uma pasta em Google Drive, compartilhada com os orientadores do projeto, para uso do treinamento da equipe ADAV. A pasta contém todos os arquivos relevantes do projeto (fotos, modelos e imagens), aulas de como adequar modelos, cadastros e edições de ossos, assim como aulas voltadas aos próprios orientadores, sobre o funcionamento da interface de administrador do projeto. Tudo foi organizado em pastas para maior organização da documentação, como pode ser visto na Figura 26:

Figura 26 - Drive de documentação



Fonte: Do Autor (2023)

5. CONSIDERAÇÕES FINAIS

Com seu desenvolvimento inicial de 2013-2015, na era da digitalização do ensino, ADAV foi uma ferramenta inovadora, como ferramenta auxiliar para a UFLA e outras universidades a fora, com solicitações frequentes direcionadas aos coordenadores do projeto para uso da ferramenta em cursos de Medicina Veterinária, graças a seu acervo grande de ossos digitalizados, descrições educativas e iteratividade.

O projeto ADAV 3D foi descontinuado por limitações técnicas e fatores externos. Com sua reconstrução, pôde retornar, com mais funcionalidades do que nunca, e se reintegrar no ensino da UFLA, e potencialmente até mesmo em outras universidades como previamente.

Existe extensa documentação de como se utilizar o projeto, para que novas equipes possam estender seu banco de dados, e criar uma ferramenta de ensino robusta.

O projeto foi desenvolvido ao longo de seis meses, dentre estes, três em capacitação dos membros do projeto em todas as ferramentas, tecnologias e metodologias utilizadas em seu desenvolvimento, e os últimos três em desenvolvimento em equipe, onde suas atividades foram separadas em 3 Sprints de 1 mês de desenvolvimento, respectivamente.

5.2. Trabalhos Futuros

Apesar de ser uma base sólida, existem vários trabalhos que podem ser desenvolvidos em cima deste projeto:

- Uma interface dedicada aos discentes da equipe do projeto, onde possam acompanhar status de suas submissões, possivelmente com integração de e-mails.
- Expansão de funcionalidades de cadastro, com ainda mais informações, e adição de transformações no cadastro do modelo, diminuindo ainda mais o trabalho manual feito em Blender.
- Um Atlas 3D pode ser expandido para novos departamentos e equipes, ADAV é o primeiro atlas digital 3D desenvolvido pela UFLA, mas muitos outros ainda podem vir, a partir deste mesmo sistema, com a aplicação WebGL já desenvolvida.
- Suporte a realidade aumentada, cada vez mais acessível graças à acessibilidade e culturalização de celulares.
- Geração de relatório de atividades automatizado, sistema faz ligação de autor com cadastros, possibilitando esta expansão.
- Reintegração do ADAV 2D ao sistema, através de um novo módulo de expansão.

REFERÊNCIAS

BECK, Kent; BEEDLE, Mike; BENNEKUM, Arie van; *et al.* **Manifesto para Desenvolvimento Ágil de Software.** 2001. Disponível em: https://agilemanifesto.org/iso/ptbr/manifesto.html Acesso em: 06/12/2023.

BLENDER, Blender, 2023. Disponível em: https://www.blender.org/ Acesso em: 05/11/2023.

BLENDER. **Blender 3.6 Manual: glTF 2.0.** Blender Fountation, 2023. Disponível em: https://docs.blender.org/manual/en/latest/addons/import_export/scene_gltf2.html#gltf-2-0. Acesso em: 05/11/2023.

CAETANO, Adrielle de P.; CASTRO, Elias M.; GUIMARÃES, Gregório C.; *et al*. **Atlas Digital de Anatomia Veterinária 3D (ADAV 3D): módulo osteologia dos ruminantes**. CIUFLA, 2015.

CETIC. **TIC DOMICÍLIOS 2017**. Cetic.br, 2017. Disponível em: https://cetic.br/media/analises/tic_domicilios_2017_coletiva_de_imprensa.pdf. Acesso em: 03/12/2023.

CONCI, Aura. **O QUE É COMPUTAÇÃO GRÁFICA? C.G.** Universidade Feral Fluminense, 2019. Disponível em: http://profs.ic.uff.br/~aconci/Aula1-2019-2.pdf. Acesso em: 04/12/2023.

CUBA, Gabrielle A.; GUIMARÃES, Gregório C.; MELCHIORI, Ana P. **Melhorias no ADAV: Atlas de Anatomia Veterinária**. CIUFLA, 2020. Disponível em: https://prp.ufla.br/ciuflasig/generateResumoPDF.php?id=14517. Acesso em: 05/11/2023.

DJANGO, Django Rest Framework, 2023. Disponível em: https://www.django-rest-framework.org/ Acesso em: 07/12/2023.

DREI, React Three Drei, 2023. Disponível em: https://github.com/pmndrs/drei Acesso em: 07/11/2023.

GOOGLE DRIVE, Google. Disponível em: https://www.google.com/intl/pt-br/drive/about.html?offset=480 Acesso em: 05/11/2023.

POSTMAN, Postman, 2023. Disponível em: https://www.postman.com/ Acesso em: 08/11/2023.

REACT, React, 2023. Disponível em: https://pt-br.react.dev/ Acesso em: 08/11/2023.

REDUX TOOLKIT, Redux Toolkit, 2023. Disponível em: https://redux-toolkit.js.org/ Acesso em: 08/11/2023.

REACT THREE FIBER, **React Three Fiber Docs**, React Three Fiber, 2023. Disponível em: https://docs.pmnd.rs/react-three-fiber/getting-started/introduction Acesso em: 02/11/2023

SCHWABER, Ken; SUTHERLAND, Jeff. Guia do Scrum: **The Scrum Guide: The Rules of the Game.** Scrum Guides, 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf#zoom=100 Acesso em: 06/12/2023.

SHORTCUT, Shortcut, 2023. Disponível em: https://www.shortcut.com/ Acesso em: 07/12/2023

SQLITE, SQLite, 2023. Disponível em: https://www.sqlite.org/index.html Acesso em: 06/11/2023.

THREE.JS. **Three.js manual.** Three.js Manual, 2023. Disponível em: https://threejs.org/docs/index.html#manual/en/ Acesso em: 08/11/2023.

THREE.JS., Three.js, 2023. Disponível em: https://threejs.org/ Acesso em: 08/11/2023

UFRGS. **Vistas Ortográficas.** Universidade Federal do Rio Grande do Sul, 2016. Disponível em: https://www.ufrgs.br/destec/wpcontent/uploads/2016/07/vistas_ortograficas.pdf Acesso em: 10/11/2023.

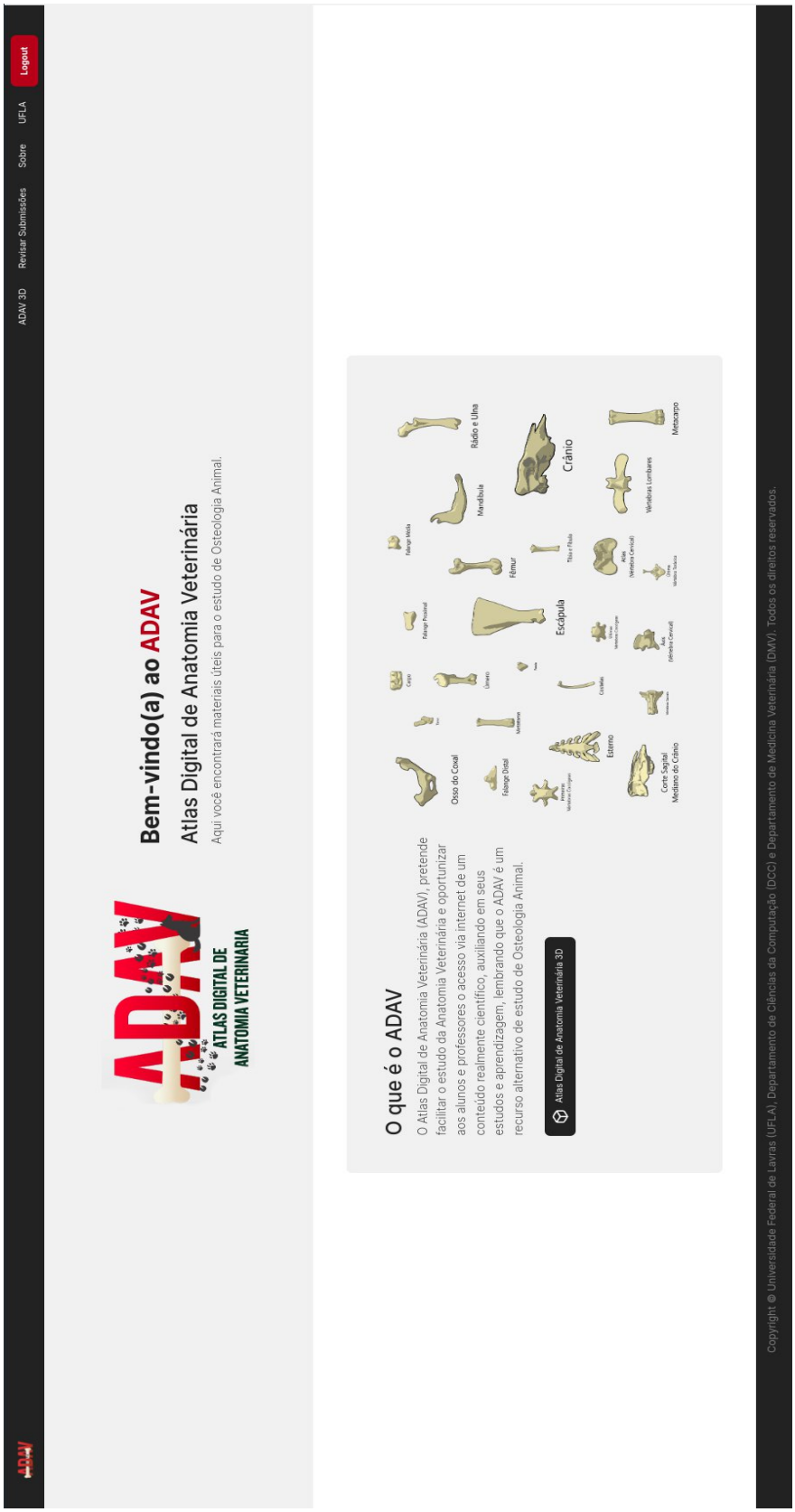
UNITY, Unity Technologies, 2023. Disponível em: https://unity.com/pt Acesso em: 05/11/2023

UNITY. **Unity Manual 2022.3.** Unity Technologies, 2022. Disponível em: https://docs.unity3d.com/Manual/webgl-browsercompatibility.html. Acesso em: 10/11/2023.

VITE, Vite. Disponível em: https://pt.vitejs.dev/guide/ Acesso em: 05/11/2023.

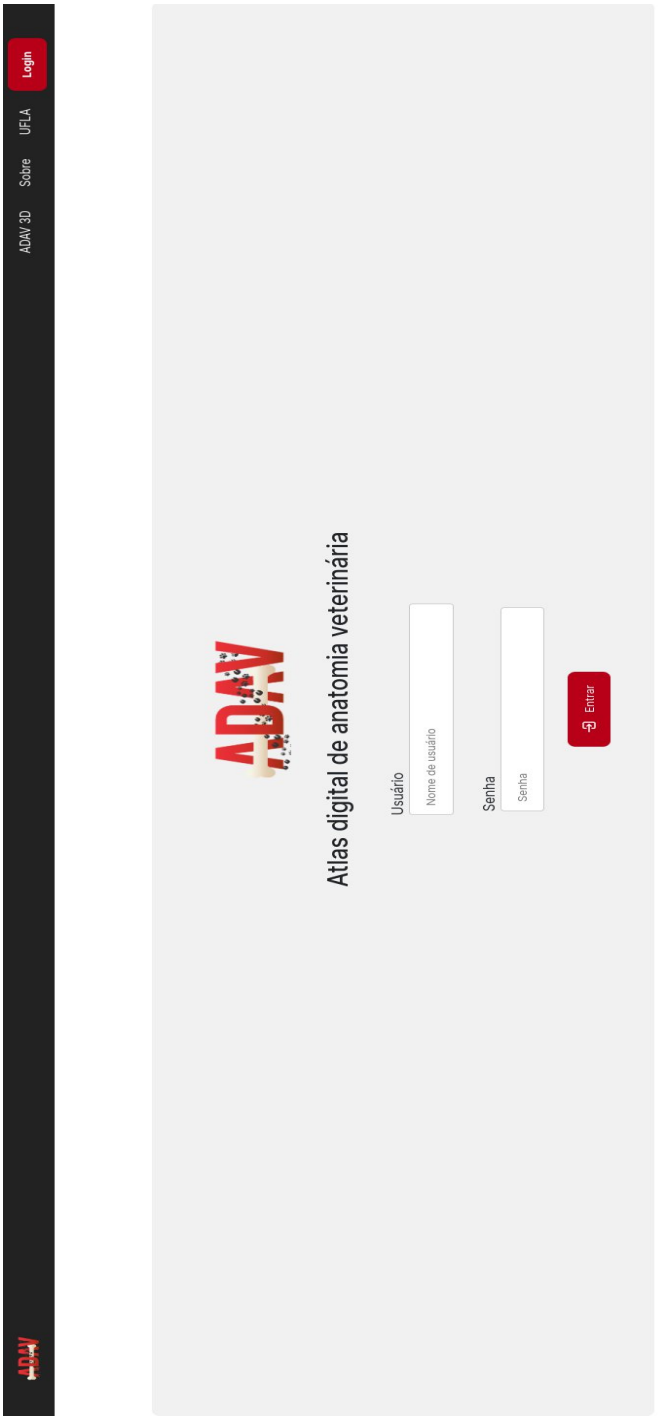
APÊNDICE A

Figura 27 - Página inicial



APÊNDICE B

Figura 28 - Página de Login



APÊNDICE C

Figura 29 - Página Sobre