UNIVERSIDADE FEDERAL DO MATO GROSSO DO SUL

FACOM – FACULDADE DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

SISUNAPI: FERRAMENTA COMPUTACIONAL DE INTEGRAÇÃO DOS DADOS DA UNAPI/UFMS

RENATO LEMES PEIXOTO

ORIENTADORA: PROFª. DRA. ANDREA TERESA RICCIO BARBOSA

CO-ORIENTADORA: PROFª. DRA. SUZI ROSA MIZIARA BARBOSA

Campo Grande - MS
Agosto/2022

UNIVERSIDADE FEDERAL DO MATO GROSSO DO SUL

FACOM – FACULDADE DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

SISUNAPI: FERRAMENTA COMPUTACIONAL DE INTEGRAÇÃO DOS DADOS DA UNAPI/UFMS

RENATO LEMES PEIXOTO

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada da Faculdade de Computação/FACOM - UFMS, como parte dos requisitos para a obtenção do título de Mestre em Computação Aplicada.

Campo Grande - MS
Agosto/2022

TERMO DE APROVAÇÃO

SISUNAPI: FERRAMENTA COMPUTACIONAL DE INTEGRAÇÃO DOS DADOS DA UNAPI/UFMS

Dissertação apresentada ao Programa de Pós-Graduação em Computação Aplicada da Faculdade de Computação/FACOM da Universidade Federal de Mato Grosso do Sul, como parte dos requisitos para obtenção do título de Mestre em Computação Aplicada.

30 de agosto de 2022

Comissão Examinadora

_____
Profa. Dra. Andrea Teresa Riccio Barbosa
FACOM/UFMS

_____
Profa. Dra. Suzi Rosa Miziara Barbosa
INISA/UFMS

_____
Prof. Dr. Ramon Morais Penha
INISA/UFMS

_____
Profa. Dra. Débora Maria Barroso Paiva
FACOM/UFMS

A pequena semente espalhada nesse mundo:
Theodoro Alberto, meu filho.
A força e alegria em todos os dias:
Ligia Freitas, meu amor.
Em memoria a João Lemes Muniz.

Dedico.

AGRADECIMENTO

Agradeço primeiramente ao criador supremo do universo, aquele que é responsável por todas as coisas. Para muitos é Javé, Alá, Tupã, Oxalá... DEUS.

Agradeço todo o apoio e dedicação da professora Andrea Barbosa. Todo o seu carinho, paciência e atenção dedicado durante todo o tempo de estudo. Todo o meu respeito e admiração a sua pessoa professora. Muito obrigado.

Agradeço a todos que estiveram comigo durante esse período. A minha companheira Ligia, que me incentivou e nunca me fez baixar a cabeça por nada. A minha mãe Fátima por sempre me motivar. Ao meu filho Theodoro que, por mais que não tenha noção, me ajuda sempre. Ao meu pai Fernandes, por sempre cuidar do Theo, para que eu possa desenvolver minhas atividades profissionais e acadêmicas.

Agradeço aos meus amigos Vinicus Okamura, João Victor Battistelli e Lucas Silva, companheiros de Geneplus, que o destino fez percorrer por caminhos distintos, deixo aqui o meu obrigado.

Agradeço a equipe do Geneplus e da Embrapa Gado de Corte, ao qual fiz grande parte do caminho.

Agradeço aos irmãos *Dorks*, Francisco, Marcelo, Euzébio e Éder. Crescemos juntos e juntos somos mais fortes... *Hy Dork*!

Agradeço a todos os mestres que me tornaram o que eu sou hoje, em especial aos professores Waldemar, Elô, Alice, Miguel, Aislan e Miska. São mestres da formação secundária, mas os quais estão presentes até os dias atuais. Se aqui cheguei foi porque me apoiei nos ombros de gigantes! Obrigado professores.

Èsù gbe eni se ebo lore o! Ògún olónà olà! Òsun òyéyéni mò!

E, a todos que um dia duvidaram... o meu mais sincero obrigado! É na dúvida que as verdades surgem.

RESUMO

O Instituto Integrado de Saúde (INISA) é uma unidade setorial da UFMS com diretrizes para a integração ensino-serviço-comunidade e através de diversas atividades diferenciadas oferecem atenção à saúde nos diferentes níveis de complexidade. Verifica-se que no Instituto, apesar de diversos projetos de pesquisa e extensão estarem em andamento, não há um sistema centralizado que armazenasse os dados e que possibilite o acompanhamento de informações. Um desses projetos de extensão intitulado Avaliação Multidisciplinar de Saúde dos Idosos e frequentadores da UNAPI/UFMS (Universidade Aberta à Pessoa Idosa/UFMS) tem por objetivo a análise global e multiprofissional da saúde dos idosos acima de 60 anos. Para o projeto em questão, algumas dificuldades foram observadas tais como: a descentralização dos dados referentes aos projetos de extensão de cultura e esporte no qual participam; a falta de centralização das informações fisiológicas e psicossociais do idoso, entre outras. Salienta-se que muitos dados estão distribuídos por diversos setores do Instituto e com pesquisadores diferentes, tanto em formato de planilhas eletrônicas, como em formulários de papel. Esse trabalho apresenta um estudo para a estruturação e levantamento de requisitos de um sistema (SISUNAPI) capaz de atender a demanda da UNAPI, apresentando os principais recursos e funções necessárias. Como método foi utilizado um *framework* e criado um Banco de Dados que atendeu aos requisitos mínimos para obtenção de informações básicas dos participantes dos projetos de pesquisa, ensino e extensão. Foi desenvolvido um protótipo capaz de realizar o armazenamento dos dados de pesquisa e sua ligação entre os pesquisadores e pesquisados.

Palavras chaves: pessoa idosa; sistema acadêmico; banco de dados; *framework;* inclusão digital

ABSTRACT

Integrated Health Institute (INISA) is sectorial unit of UFMS with guidelines for the teaching-service-community integration and, through several different activities, they offer health care different levels complexity. It appears that Institute, although several research and extension projects underway, there no centralized system that stores data allows for obtaining and monitoring information. One these extension projects entitled Multidisciplinary Health Assessment of the Elderly People UNAPI/UFMS (Open University for the Elderly People/UFMS) aims at global and multidisciplinary analysis of health of elderly over 60 years old. For project in question, some difficulties were observed, such as: decentralization of data referring to cultural and sport extension projects in which they participate; the lack of centralization of physiological and psychosocial information of elderly people, among others. It should be noted that many data are distributed across different sectors of Institute and with different researchers, both in the form of electronic spreadsheets and paper forms. This work presents a study for structuring and requirements gathering of a system (SISUNAPI) capable of meeting UNAPI demand, presenting main resources and necessary functions. As a method, a framework was used and a Database was created that met the minimum requirements for obtaining basic information from participants in research, teaching and extension projects. A prototype capable of storing research data and linking them between researchers and research subjects was developed.

Keywords: elderly people; academic system; database; framework; inclusion

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AVA – Ambiente Virtual de Aprendizagem

APP – *Application*

BSB – *Berkeley Software Distribution*

CEI – Clínica Escola Integrada

Cetic – Centro Regional de Estudos para o Desenvolvimento da Sociedade da informação

CEP – Código de Endereçamento Postal

CINTED – Centro Interdisciplinar de Novas Tecnologias na Educação

CMS – *Content Management Systems*

CNPGC – Centro Nacional de Pesquisa de Gado de Corte

CNPJ – Cadastro Nacional de Pessoa Jurídica

COVID-19 – *Corona Virus Disease*, 19

CPF – Cadastro de Pessoa Física

CSRF – *Cross-Site Request Forgery*

CSS – *Cascading Style Sheet* – Folha de Estilos em Cascata

DCT – Departamento de Computação e Estatística

DRY - *Don't Repeat Yourself*

e-gov – *electronic Government*

ETD – Educação Temática Digital

HTML – *Hyper Text Markup Language*

HTTP – *Hyper Text Transfer Protocol*

HTTPS – *Hyper Text Transfer Protocol Secure*

IBCT – Instituto Brasileiro de Informação em Ciência e Tecnologia

IBGE – Instituto Brasileiro de Geografia e Estatística

IFCE – Instituto Federal do Ceará

INISA – Instituto Integrado de Saúde

IP – *Internet Protocol*

LA – *Learning Analytics*

LEDES – Laboratório de Engenharia de *Software*

LGPD – Lei Geral de Proteção de Dados

LV – *Learning Vectors*

MDE – Mineração de Dados Educacionais

MVC – *Model-View-Controller*

RG – Registro Geral

OMS – Organização Mundial de Saúde

ORM – *Object-Relational Mapping*

P2P – *peer-to-peer*

PHP – *Hypertext Preprocessor*

PSF – *Python Software Foundation*

SBGG – Sociedade Brasileira de Geriatria e Gerontologia

SGBD – Sistema de Gerenciamento de Banco de Dados

SISCAD – Sistema Acadêmico

SSL – *Secure Socket Layer*

TI – Tecnologia da Informação

UFMS – Universidade Federal de Mato Grosso do Sul

UFPE – Universidade Federal de Pernambuco

UFRGS – Universidade Federal do Rio Grande do Sul

UML – *Unified Modeling Language*

UnAPI – Universidade Aberta à Pessoa Idosa

UNFPA – Fundo de População das Nações Unidas

UTF-8 – *8-bit Unicode Transformation Format*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

A expectativa de vida do brasileiro vem crescendo com a melhoria da qualidade de vida. Segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), a expectativa de vida de homens e de mulheres é de 73,1 e 80,1 anos, respectivamente (IBGE, 2019). Durante a década de 1940, a média era de 45,5 anos (IBGE, 2019) e com os avanços da medicina, do saneamento básico e da qualidade de vida, a expectativa subiu para os atuais níveis registrados.

Com o aumento da expectativa de vida a ocupação dessa parcela da população passou a ser analisada e incluída nas políticas públicas de diversas secretarias de governo. O conceito de qualidade de vida está ligado a um conjunto de fatores que abrange diversos aspectos como o socioeconômico, o estado emocional, o cuidado pessoal, o suporte familiar e a inclusão social, entre outros (Vecchia, R.D. *et al.,* 2005).

Com o intuito de obter a inclusão e interação social, os idosos buscam por oficinas, cursos profissionalizantes, palestras e atividades para complementar a formação e aumentar as suas habilidades cognitivas. Em contraparte, as entidades de apoio oferecem as ferramentas para a integração dos idosos. Com a implantação da Universidade Aberta Pessoa Idosa (UnAPI) em 2011, no Instituto Integrado de Saúde (INISA), a Universidade Federal de Mato Grosso do Sul (UFMS) passou a oferecer toda a estrutura acadêmica para as pessoas idosas.

O INISA é uma unidade setorial da UFMS destinada a fornecer subsídios para os estudantes de Bacharelado em Enfermagem, Bacharelado em Fisioterapia, Mestrado em Enfermagem e Mestrado Profissional em Saúde da Família, além de Residência em Enfermagem Obstétrica e Residência Multiprofissional em Saúde – Atenção ao Paciência Crítico. O instituto oferece, através da Clínica Escola Integrada (CEI), atendimentos a toda a sociedade, com projetos de ensino, pesquisa e extensão.

O objetivo do INISA é atender, através de projetos de pesquisa e com a participação de estudantes dessas diversas áreas, toda a sociedade, tanto no aspecto social quanto no de saúde. Dessa forma, poderá oportunizar o desenvolvimento da autonomia e independência também dos idosos.

Salienta-se que, atualmente, o instituto possui diversos projetos voltados a pessoa idosa, sendo alguns voltados ao bem-estar geral. Para o projeto Avaliação Multidisciplinar de Saúde dos Idosos Frequentadores da UnAPI/UFMS não há um histórico, em um sistema computacional integrado, sobre a assiduidade de participação dos idosos.

Muitos dados estão distribuídos por diversos setores do INISA e sobre supervisão de diferentes pesquisadores, estando tanto em formato de planilhas eletrônicas como em formulários de papel. Alguns dos participantes do projeto estão ligados às oficinas, em atividades extracurriculares, matriculados em disciplinas de cursos de graduação e também recebem atendimentos ambulatoriais na Unidade de Fisioterapia e Enfermagem.

Observou-se que com o aumento dos atendimentos pelo Instituto, a falta de um sistema que concentre o histórico do atendimento dificulta os estudos posteriores, tanto por parte dos alunos quanto dos pesquisadores. Também não é possível de se obter um *feedback* de fácil acesso pelos participantes dos projetos. Além disso, não há centralização de informação, fato esse que pode gerar a desmotivação por parte dos participantes e até mesmo a evasão dos projetos.

Em contrapartida, com a Tecnologia da Informação (TI), todos os dados processados podem ser tratados e administrados para gerar informação futura. Com a utilização de ferramentas de modelagem de dados é possível realizar a definição dos principais dados que são necessários em uma aplicação. Com a utilização da *UML* (*Unifield Modeling Language*), métodos ágeis em Engenharia de *Software* e a escolha de um Sistema de Gerenciamento de Banco de Dados (SGBD) é possível apresentar um protótipo de sistema manipulável, onde as principais características sejam apresentadas.

Diante do exposto, este trabalho multidisciplinar tem por objetivo apresentar uma estrutura de sistema computacional que realize a integração da participação dos idosos em projetos da UnAPI/UFMS. Em particular o cadastro dos usuários, suas fichas e relatórios de desempenho, seus acompanhamentos de saúde e os históricos acadêmicos em disciplinas de cursos de graduação, entre outros. A proposta é fazer uma versão que poderá ser base a outros sistemas que possam ser desenvolvidos posteriormente e integralizados.

1.1 Objetivo

Desenvolver uma ferramenta computacional de integração de dados (SISUNAPI) e uma estrutura de um Banco de Dados para armazenar as informações dos participantes dos projetos do INISA.

1.2 Objetivos Específicos

- definir os requisitos do sistema;
- definir um Sistema de Gerenciamento de Banco de Dados (SGBD);
- verificar possíveis *frameworks* existentes para suprir com os paramentos preestabelecidos;
- definir modelos de formulários para coleta de informação dos atores do projeto;
- desenvolver um protótipo para testes;
- validar o protótipo para liberação de uma versão de produção;
- estabelecer o Sistema como válido e liberação para a sociedade.

1.3 Justificativa

O desenvolvimento do trabalho baseia-se em uma demanda do INISIA, pois há dificuldade por parte de todos os envolvidos nas atividades, sendo esses pesquisadores ou usuários, de terem acesso às informações básicas sobre suas atividades.

As disciplinas acadêmicas ofertadas aos idosos não estão integradas ao Sistema Acadêmico da UFMS, bem como as atividades de extensão não ficam registradas junto com as demais informações de dados pessoais e de saúde. Ou seja, não há centralização e facilidade de uso da agenda de atividades, dos resultados de performance, das atividades diárias, das notas, entre outras informações relevantes.

Também se percebe que há necessidade do arquivamento das atividades dos docentes que atendem essa parcela da sociedade. E, manter um histórico geral dos participantes é vital para estudos futuros.

Com a criação do Banco de Dados centralizado, não apenas as informações dos usuários estarão disponíveis, como será possível, através da modularização de sistemas, a unificação com os prontuários médicos dos participantes. Dessa forma, será possível gerar um repositório de dados sobre a vida social dos participantes dos projetos da UnAPI, em especial aos participantes da INISA. Para outras informações necessárias nas diversas atividades no INISA poderão ser desenvolvidos sistemas que se integrarão a esse banco de dados único.

Dessa forma, esse trabalho apresenta contribuição social, pois a proposta é desenvolver um sistema que facilite o acesso às informações por parte dos idosos; contribuição acadêmica, pois inúmeras pesquisas poderão ser realizadas com os dados armazenados; contribuição na área de saúde pela possibilidade de acesso ao histórico do paciente e pela possibilidade de aquisição de dados; contribuição tecnológica porque será feito o registro de *software* do que será desenvolvido, além do crescimento profissional e acadêmico por parte do autor.

1.4 Organização do trabalho

A estrutura do trabalho está organizada da seguinte forma: o Capítulo 2, apresenta-se alguns Trabalhos Relacionados, obtidos pela revisão sistemática realizada na literatura sobre a utilização de *Frameworks*, Banco de Dados, Segurança da Informação, Sistemas Acadêmicos, Atenção à Pessoa Idosa, Informática para a Saúde, Envelhecimento entre outros temas correlacionados. No Capítulo 3, apresenta-se a metodologia considerando as etapas e os métodos realizados. No Capítulo 4, apresenta-se a Codificação Utilizada na construção do protótipo e a estruturação do Banco de Dados. No Capítulo 5 são apresentados os resultados de toda a construção do protótipo. O último tópico, o Capítulo 6, apresentam-se os trabalhos futuros e as conclusões finais do trabalho.

Capítulo 2

TRABALHOS RELACIONADOS

Esse capítulo apresenta trabalhos que podem contribuir para o desenvolvimento dessa dissertação, qual seja, o papel do idoso na sociedade atual, educação voltada para terceira idade, sistema de coleta de informação sobre saúde, sistemas de gerenciamento acadêmico, banco de dados e *frameworks*.

2.1 Revisão Bibliográfica

As consultas em fontes relacionadas foram realizadas durante o período de setembro a novembro de 2020. Foi utilizado para a pesquisa o *software Hazing's Publish or Perish*, que utiliza *sites* como *Elsevier* (*Journal of Cleaner Production*) e *Emerald* (*International Journal of Sustainability in Higher Education*). Foi utilizado o motor de busca do *Google Scholar* com suas pesquisas centralizadas em páginas em Português e para a busca de artigos e teses a plataforma virtual da Biblioteca Brasileira de Teses e Dissertações do Instituto Brasileiro de Informação em Ciência e Tecnologia – IBICT, entre outros.

Foram utilizadas as palavras chaves durante as pesquisas:

- Sistema Acadêmico para Idosos;
- Informática + idosos + saúde + sistema de cadastro + usuários;
- Banco de dados + sistema de cadastro + idosos + informática em saúde;
- Sistema Acadêmico + idosos + cadastro;
- Envelhecimento + educação + sistema acadêmico;
- Banco de dados + envelhecimento + saúde;
- *Academic system* + *Elderly People* + *Education;*
- UNAPI + UFMS + Idosos;
- Universidade + Terceira Idade + educação;
- Banco de Dados + *framework*;

- *Academic System Elderly People;*
- *Elderly People + Education + Computer + Social Integration;*
- *Database + System Academic.*

Durante a pesquisa na plataforma IBICT foram apresentados cerca de vinte trabalhos sobre o tema. Após análise preliminar, foram selecionados dois trabalhos, sendo uma tese e uma dissertação. Como a pesquisa é multidisciplinar, os trabalhos apresentados durante a busca são abrangentes sobre os temas e uma análise detalhada restringiu o número de trabalhos relacionados.

Com o *software Hazing's Publish or Perish* foram realizadas cinco buscas distintas. Em ambas as buscas foram apresentados cerca de duzentos trabalhos. Para a seleção de trabalhos que melhor se relacionam com o tema, foi realizado um filtro com os de maior relevância, data de publicação e posteriormente uma análise sobre o título e resumos.

Como a pesquisa é sobre um tema multidisciplinar, foi necessário a busca pelas áreas de saúde, educação, sociologia e tecnologia.

Foi utilizado o repositório da biblioteca da UFMS para a busca de trabalhos que possuíam relação com os temas da pesquisa e assim foram selecionados quatro trabalhos.

Também foi utilizado o motor de busca do *Google* e *Google Scholar* sobre os temas da pesquisa e foram apresentados vinte e quatro trabalhos dos quais três foram retirados para uma leitura mais detalhada.

Com base nas palavras chaves de busca utilizadas nas pesquisas foram selecionados seis trabalhos sobre o tema Banco de Dados; dez trabalhos sobre o tema Educação; quatro trabalhos sobre o tema *Frameworks*; dois trabalhos sobre o tema Qualidade de Vida; cinco trabalhos sobre o tema Sistemas Acadêmicos e seis trabalhos sobre o tema Universidade Aberta, totalizando trinta e três trabalhos.

Foram selecionados quinze trabalhos (doze artigos científicos e três dissertações de mestrado) para uma leitura mais detalhada que serviram de base para a presente pesquisa.

2.2 Detalhamento dos trabalhos

Machado *et. al*. (2016) realizaram um mapeamento de competências digitais em idosos que participavam de cursos de inclusão digital. Para realizar o trabalho, solicitaram, com formulários próprios, autorização para coleta das informações com todos os participantes da

pesquisa. Com a possibilidade de ampliar ou mesmo aprimorar as competências, foram selecionados trinta e um idosos de oficinas da Universidade Federal do Rio Grande do Sul (UFRGS) para a participação.

Segundo Cetic (2014, *apud* Machado *et. al*., 2016), 14% dos idosos da faixa etária acima de 60 anos utilizam computadores diariamente. E os autores definiram na pesquisa que a inclusão digital aprimora a saúde mental e desenvolve competências ao manusear novas ferramentas, muitas vezes ligada a comunicação. A Figura 1 apresenta as competências específicas resultantes de inclusão digital de idosos.

Figura 1 - Competências específicas resultantes de inclusão digital de idosos.

Fonte: Machado *et. al.* 2016

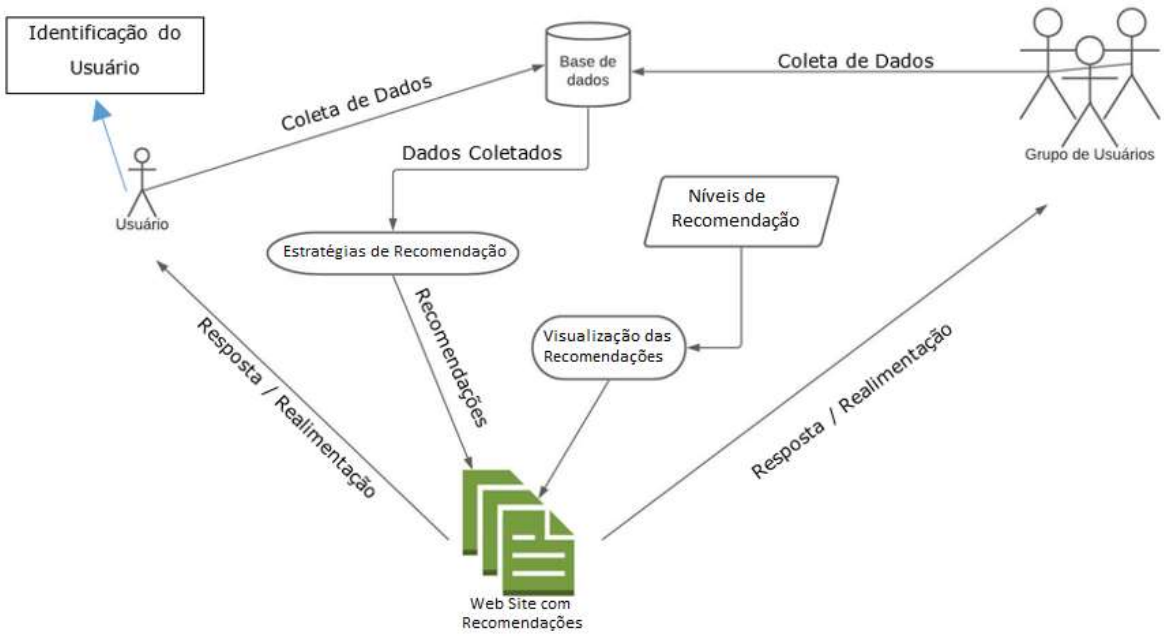
Machado *et. al.* (2016) trabalharam com conjuntos de recursos cognitivos essenciais e suas junções que proporcionaram três características: conhecimento, habilidades e atitude. Na velhice as interações sociais são essenciais e grande parte ocorrem virtualmente, com a utilização de ferramentas.

Os autores relatam sobre a falta de literatura associada ao tema, o que provoca a reflexão sobre quais seriam as habilidades que devam ser desenvolvidas. As habilidades propostas foram: alfabetização digital, letramento digital e fluência digital. Assim, foram desenvolvidos formulários, apropriados para as pessoas da terceira idade, sobre cada tema para identificar o nível de conhecimento sobre cada proposta. Houve a preocupação com o histórico social de cada participante da pesquisa. Com as respostas do grupo, foi desenvolvido um mapeamento a ser seguido para que a curva de aprendizado do idoso seja adequada.

Segundo Schafer (2000, *apud* Barcellos *et. al.,* 2007) a estrutura dos sistemas de recomendação é dividida em 4 processos distintos: identificação do usuário, coleta de

informações, estratégia de recomendação e visualização das recomendações, sendo o processo de identificação do usuário o principal processo. Há três tipos de recomendação: "não recomendado", "recomendação efêmera" e "recomendação persistente". O tipo "não recomendado" é indicação de resposta padrão para todos os usuários; "recomendação efêmera" são respostas baseadas apenas na navegação atual do usuário sem cadastro ou *login* prévio; "recomendação persistente" é a recomendação baseada no perfil existente do usuário, sendo considerado todas as pesquisas prévias realizadas anteriormente, para determinar as novas recomendações. A Figura 2 apresenta a estrutura visual do sistema de recomendação para o usuário.

Figura 2 - Estrutura do Sistema de Recomendação para o usuário.



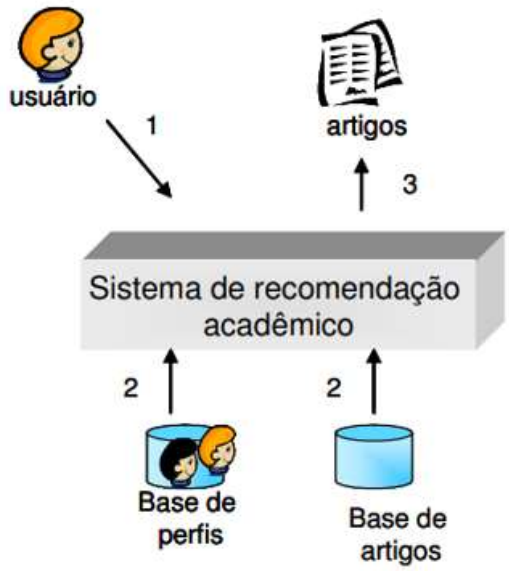
Fonte: Adaptação (Schafer, 2002 *apud* Barcellos *et. al*. 2007).

Barcellos *et. al.* (2007) realizaram a proposta de oferecer um sistema de recomendação que gere automaticamente o perfil do usuário. Essa geração é realizada através da coleta das informações da própria rede, através do currículo, rede social, entre outros. Obtiveram sucesso na geração de respostas relevantes para determinadas pesquisas e na redução do tempo de pesquisa, pois auxiliou nos resultados para os alunos e pesquisadores.

Sales *et. al.* (2014) propuseram a concepção, modelagem e desenvolvimento de um *software* com utilização da linguagem iconográfica em Ambientes de Aprendizagem Virtual (AVA), em especial no ambiente *Moodle*. Em conjunto com o sistema de avaliação proposto, o trabalho conseguiu informações sobre a frequência dos cursistas e suas atividades. Na Figura 3 é apresentado o sistema de recomendação acadêmico projetado. Os autores sugerem como

trabalhos futuros a implementação das funcionalidades para outros ambientes AVA de modo a atingir maior número de alunos e professores.

Figura 3 - Sistema de recomendação acadêmico.



Fonte: Barcellos *et. al*., 2007.

Segundo Sales (2010 *apud* Sales *et. al.,* 2014) os *Learning Vectors* (LV), apresentado na Figura 4, são representações vetoriais concebidas para automatizar o processo de avaliação qualitativos e quantitativo no AVA, trazendo referência do nível de desempenho dos alunos nos cursos (Sales, Barroso e Soras, 2012). A proposta é a utilização do modelo para a classificação da avaliação por parte dos utilizados dos sistemas. Uma representação gráfica apresenta uma visão empírica de um valor numérico. Quando apresentado para pessoas com pouca familiaridade, uma imagem é melhor interpretada do que um valor numérico. Os autores sugerem como trabalhos futuros, a implementação das funcionalidades para outros ambientes AVA, atingindo maior número de alunos e professores.

Figura 4 - Menções qualitativas e LV ícones.

Menções Qualitativas	Muito Bom	Bom	Regular	Fraco	Não Satisfatório	Neutro
LV Ícones						

Fonte: Sales *et. al*. 2014.

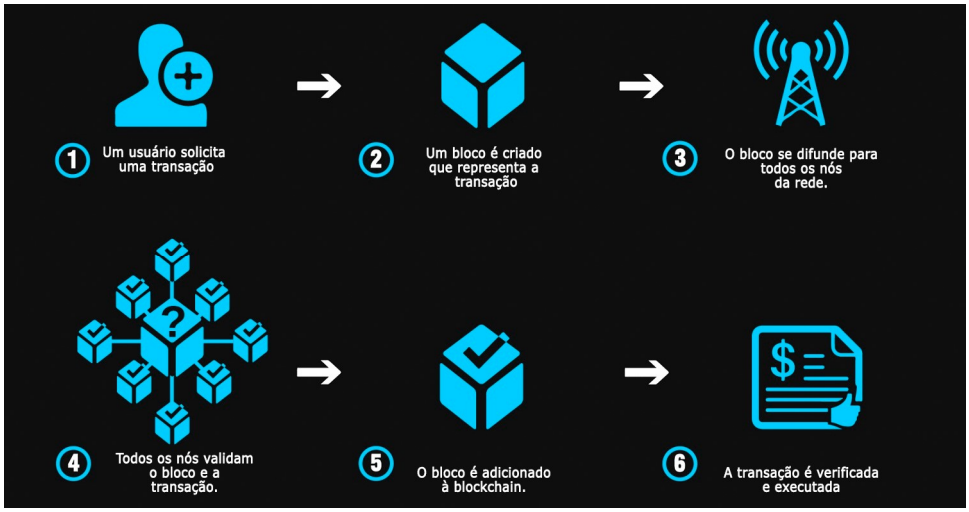
Agostinho *et. al*. (2018) propuseram a unificação de dados, prontuários e resultados de saúde com a utilização do *blockchain* para garantir além da validação, a confiabilidade dos dados. Salienta-se que a *blockchain* é uma tecnologia que lida com informações em aplicações *peer-to-peer* (Nakamoto, 2018 *apud* Agostinho *et. al*. 2018). Ou seja, para realizar uma validação de uma informação, essa deve ser validada por diversos *nodos* que fazem parte da rede. Dessa forma, apenas após essa validação completa a informação pode ser inserida ou excluída da rede.

Os autores citam que devido à alta sensibilidade das informações de saúde e a ausência de tecnologia capaz de suprir o sigilo, entraves foram criados nos sistemas de informação que realizam a unificação desse tipo de informação.

Segundo ainda Tasatanattakool e Techapanupreeda (2018 *apud* Agostinho *et. al*. 2018) *blockchain* é uma forma de armazenamento de dados não centralizado, confiável e difícil de utilizar para fins fraudulentos (Figura 5). Os dados são armazenados de forma distribuída e, todos os *nodos* participantes da rede, possuem uma cópia da informação sem a necessidade da centralização de um banco de dados (Tema *et. al*, 2018 *apud* Agostinho *et. al*. 2018).

Os autores citam diversos trabalhos que manipulam os dados de saúde. Muitos dos trabalhos citados tratam de acesso a informações para a manipulação dos dados sensíveis, porém nenhum trata especificamente da segurança em caso de acesso indevido nessas informações. Segundo Sucurovic (2007 *apud* Agostinho *et. al.* 2018) o foco do sistema estava voltado a diversas instâncias de unidades de saúde visando a política de acesso aos dados e ao sistema.

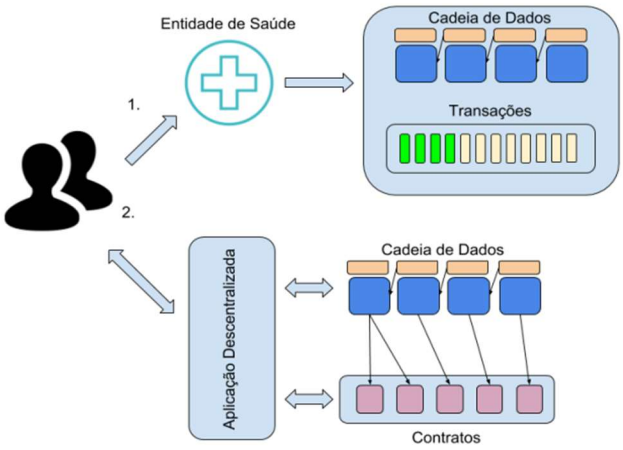
Figura 5 - Funcionamento do *blockchain*.



Fonte: Lamounier, 2018.

No trabalho de *Yuan and Wang* (2018) foi proposto o desenvolvimento de *blockchains*, dividindo o seu funcionamento em camadas. Baseando-se nesse trabalho Agostinho *et. al.* 2018 propuseram a criação de 6 camadas diferentes, porém apenas 4 camadas foram utilizadas (Figura 6). Observou-se que um detalhe em especial para a criação da rede é com relação a camada de armazenamento, pois como cada *nodo* deve ter uma cópia exata da informação, os *nodos* não podem ter restrição de armazenamento de dados. Nota-se que no experimento foi proposto como chave de acesso o CPF do usuário. Dessa forma, os resultados do trabalho demonstraram que a rede possui a viabilidade necessária para garantir que os dados se mantenham confiáveis.

Figura 6 - Arquitetura de armazenamento dos dados em saúde conforme proposta de Agostinho *et. al*. (2018)



Fonte: Agostinho *et. al.*, 2018

Os *Frameworks* podem ser definidos como um esqueleto da aplicação que será instanciada por um desenvolvedor (Pree, Sikora, 1997; Bhatti*et. al*., 2005 *apud* Weschter, 2008). Segundo o autor, *frameworks* cumprem com a função principal das técnicas de reutilização "como fazer com que desenvolvedores não precisem começar do nada para especificar e construir uma nova aplicação". O *framework* é responsável por instanciar um gerador de conteúdo que é responsável pelo Sistema de Gerenciamento de Banco de Dados (SGBD) e os arquivos do sistema.

Weschter (2008) trabalhou com a continuidade dos trabalhos realizados por Carromeu (2007) sobre *WebApps* para auxiliar a gestão das propostas eletrônicas nas agências de fomento. O trabalho centralizou-se na reusabilidade de *software*, princípio básico dos *frameworks*, onde a forma mais elementar de reutilização de artefatos de *softwares* é o reuso de linhas de código.

Weschter (2008) propôs um gerador de aplicação que integre o Sistema de Gestão do Fênix e o *framework Titan*. Segundo Carromeu (2016), o *Titan* é um *framework* para instanciação de Sistemas de Gerenciamento de Conteúdo (do inglês *Content Management Systems* – CMS), em aplicações *web* utilizadas para criar, editar, gerenciar e publicar conteúdo de forma consistentemente organizada, permitindo que o mesmo seja modificado, removido e adicionado com facilidade.

A ferramenta Fênix foi objeto de mestrado de Carromeu (2007) e seu objetivo era instanciar e gerar *WebApps* no domínio e-gov para auxiliar a gestão de propostas eletrônicas e projetos a serem avaliados por agências de fomento no Brasil. O objetivo da ferramenta é promover a transparência nas informações e desburocratização de ações no gerenciamento de projetos submetidos as agências, automatizando o processo de gerência desde a fase de submissão até as fases de administração e de manutenção (Weschter, 2008).

Weschter (2008) finaliza mencionando a importância na automatização no processo de desenvolvimento de *softwares* e que em seu trabalho o gerador automatizou a criação dos *WebApps*.

Azraoui *et. al.* (2018) propuseram uma estrutura de criptografia para banco de dados *PostgreSQL* e *MySQL*. O objetivo do trabalho era garantir a criptografia dos dados que eram enviados para nuvem[1]terceirizada sem nenhuma preocupação com a criptografia dos dados. Para o desenvolvimento do estudo, Azraoui *et. al.* (2018) trabalharam para resolver os desafios de Popa *et. al.* (2012) onde era utilizado o *CryptoDB*, *framework* com suporte a consultas de dados criptografadas. Sempre quando a criptografia era aplicada a um conjunto de dados, uma camada era removida, deixando os dados expostos.

Azraoui *et. al.* (2018) desenvolveram uma aplicação que é executada em *background* junto ao *PostgreSQL* para criptografar os dados sensíveis antes de sua inserção. Observou-se durante os testes no protótipo que o armazenamento no servidor é maior do que no caso de nenhuma criptografia. A fase de recuperação de dados é ágil, com menos de 1 segundo para realizar a descriptografia.

Os autores concluem que houve êxito no trabalho, pois conseguiram tratar dados sensíveis em servidor terceirizados na nuvem e indicam como trabalhos futuros o teste de desempenho em servidores com milhões de registros para garantir os resultados em cenários do mundo real.

[1]Nuvem é o termo utilizado para descrever uma rede global de servidores que estão conectados e operando como um único ecossistema.

Sokhranyaeva (2015) analisa a gerontologia como uma esfera de práticas educacionais em desenvolvimento ativo de grande importância para a sociedade moderna. O aspecto do envelhecimento da população russa é semelhante ao do brasileiro, onde grande parte da população vem passando por um envelhecimento ativo. Segundo a autora, em 2010 existiam 23 "países de idosos" e segundo projeções, em 2040 serão 89%, sendo que os 80% desses viverão em países em desenvolvimento (OMS, 2002; UNFPA, 2012 *apud* Sokhranyaeva, 2015).

Gerontologia é o estudo do envelhecimento nos aspectos biológicos, psicológicos, sociais entre outros. Os profissionais que atuam nessa área possuem formação diversificada, interagem entre si e com os geriatras (SBGG, 2020).

Os estudos europeus e norte-americanos foram centralizados durante a década de 1960 e 1970, em particular O. F. Von Bolnov que propôs, em 1962 o termo gerontologia como nova área de estudo e pesquisa voltada a educação de pessoas na velhice (Bollnow, 2015 *apud* Sokhranyaeva, 2015). A autora menciona a importância do idoso como guardião da cultura, seu ordenador e organizador.

Salova (2017) trabalhou com o desenvolvimento dos idosos inserindo-os na civilização da informação, por ela assim definido. Utilizando modernas tecnologias de comunicação e informação, ela projetou a redução na desigualdade social gerada durante o envelhecimento populacional russo.

Essa mesma autora discute sobre os principais valores durante a era pré-industrial, onde a autossuficiência e a independência dos idosos estava associada ao controle da propriedade familiar, que garantia a velhice digna. Já durante a era industrial houve o excesso da propaganda do culto à juventude, com depreciação da velhice o que resultou nos atuais índices de expectativa de vida e visão negativa sobre a velhice.

Salova (2017) conclui que a utilização pelos idosos de meios de comunicação e informação de massa garante sua inclusão, gerando estímulo e mudança na qualidade de vida. O ambiente virtual dá origem a interesses específicos que induzem a atividade mental e social, pois durante a troca de informações em redes sociais, passam a fazer parte da sociedade moderna. Trocam experiências sem a necessidade, muitas vezes, de identificar-se gerando um conforto social, porque passam a se comunicar com os jovens de forma igualitária, unindo-se em torno de ideias e *hobbies* comuns.

Silva *et. al.* (2015) mencionam como desafio e obrigação mundial o acolhimento da população em processo de envelhecimento, proporcionando o aprendizado permanente ao

longo da vida, desenvolvendo novas aprendizagens principalmente com relação as tecnologias de informação e comunicação (Goulart, 2007; Santos 2013 *apud* Silva *et. al.*, 2015).

Os autores realizam oficinas de informática com 40 idosos, divididos em turmas de 20 alunos cada. Nessas turmas são aplicados questionários para criar um perfil de cada participante. Ao final do curso aplicam-se novos questionários com os idosos e com um representante do grupo familiar de cada idoso. Os autores ressaltam que em média 26% dos idosos que participam das oficinas inicialmente colocam a solidão como uma doença e 70% afirmam que melhoraram sua qualidade de vida, indicando o uso do computador como fator principal.

Silva *et. al*. (2018) trabalham com essas oficinas multidisciplinares para desenvolver a autonomia e independência dos idosos participantes da UnAPI/UFMS. Já foram desenvolvidas oficinas na área de saúde, educação, memória, informática e exercícios físicos. O projeto é desenvolvido por acadêmicos da UFMS com idosos a partir de 60 anos, independente do grau de formação escolar, que residam em Campo Grande, MS com duração de cinco horas semanais.

Os resultados do projeto, que está em andamento, serão apresentados com base na perspectiva dos idosos. Silva *et. al.* (2018) indicam que, devido ao aspecto multidisciplinar, os participantes das oficinais estão apresentando maior autonomia e independência funcional com percepção positiva de seu envelhecimento.

Os trabalhos de Silva *et. al*. (2018), Weschter (2008), Carromeu (2016), Sales *et. al*. (2014), Barcellos *et. al*. (2007) e Vaz *et. al.* (2018) são base para essa pesquisa, pois apresentam, cada um em sua área, as informações necessárias para a criação de um perfil de usuário, dados para armazenamento seguro e variáveis necessárias para o *feedback* aos usuários.

Os trabalhos de Machado *et. al* (2016), Agostinho *et. al.* (2018), Azraoui *et. al.* (2018), Sohranyaeva (2015) e Salova (2017) contribuíram, cada qual em sua área de atuação, com detalhes na formação do perfil de utilização do sistema, a competência e importância do usuário idoso incluso na sociedade moderna, a importância na guarda segura das informações e a importância no pensar humano nas relações entre indivíduos.

O trabalho de Silva *et. al* (2018) é a principal fonte de informações para a realização dessa pesquisa.

A Tabela 1 apresenta, de forma resumida, os trabalhos pesquisados que foram utilizados como fonte de informação para esta pesquisa.

Tabela 1 - Trabalhos que contribuíram com o desenvolvimento do projeto.

Nome do Trabalho	*Ano*	*Citação*	*Autores*	*Local de publicação*	*Área e Característica do trabalho*	*O que pode ser utilizado na atual pesquisa*
Universidade Aberta à Pessoa Idoso (UnAPI/UFMS) e as Estratégias propostas para o Envelhecimento Ativo	2018	Silva *et. al.* (2018)	Janaina Ávalos da Silva Adriana Ferreira Modesto Fabiana R. Pessoa de Britto Juscilaine Souza da Costa Guarinti Nayla Cristiane Ferreira de Farias Suzi Rosa Miziara Barbosa	INISA/UFMS	**Área**: Saúde. Demonstram as oficinas oferecidas aos Idosos participantes da UnAPI/UFMS	É o projeto que idealizou a pesquisa, pois não há um repositório digital para o armazenamento das informações geradas pelas diversas pesquisas realizadas.
Arquitetura do Gerador de Aplicação *Web* Baseado no *FrameworkTitan*	2008	Weschter (2008)	Eberson Omar Weschter	DCT/UFMS	**Área**: Engenharia de *Software*. Demonstra a estrutura do gerador de aplicação do *Framework* Titan	Demonstra um possível *framework* para servir de *back-end* para a pesquisa.
Titan *Framework Cookbook*	2016	Carromeu (2016)	Camilo Carromeu	*Please Lab*/CNPGC LEDES/UFMS	**Área**: Desenvolvimento Ágil. Demonstra as funcionalidades do *Framework* Titan	Demonstra um possível *framework* para servir de *back-end* para a pesquisa.

Nome do Trabalho	*Ano*	*Citação*	*Autores*	*Local de publicação*	*Área e Característica do trabalho*	*O que pode ser utilizado na atual pesquisa*
SIG@LV: Sistema de Gerenciamento Acadêmico *Learning Vector*	2014	Sales *et. al.* (2014)	Gilvandenys Leite Sales Ricky Perviso Paz Allyson Bonetti França	IFCE	**Área**: Desenvolvimento de *Software*. Demonstra um sistema de avaliação em Ambiente Virtual de Aprendizado (AVA) utilizando linguagem iconográfica.	Demonstra um método de fácil compreensão para a avaliação dos usuários
Sistema de Recomendação Acadêmico para Apoio a Aprendizagem	2007	Barcellos *et. al.* (2007)	Carla Duarte Barcellos Daniela Leal Musa André Luiz Brandão MariusaWarpechowski	CINTED/UFRGS	**Área**: Sistema de Recomendação. Demonstra uma ferramenta que gera perfis de usuários baseados nas buscas realizadas na *web*.	Demonstra um método de criação de perfis de recomendação para os usuários participantes dos projetos.
Mapeamento de Competências Digitais: A inclusão Social dos Idosos	2016	Machado *et. al.* (2016)	Leticia Rocha Machado Tassia Priscila Fagundes Grande Patrícia Alejandra Behar Fabiana de Miranda Rocha Luna	ETD – Educação Temática Digital – Campinas, SP	**Área**: Inclusão Digital. Expõe as competências que devem ser desenvolvidas em idosos na área de informática	Demonstra os requisitos que devem ser desenvolvidos para os usuários do sistema.

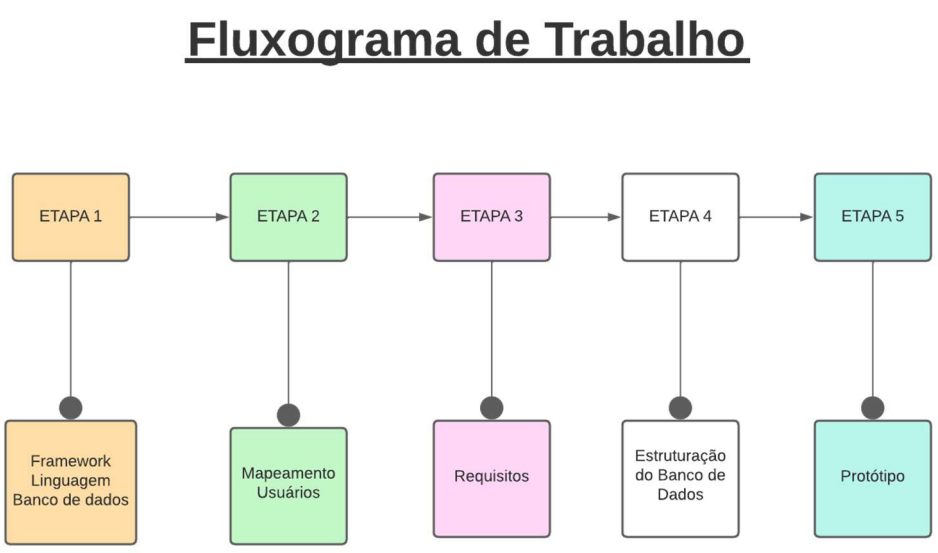
Nome do Trabalho	*Ano*	*Citação*	*Autores*	*Local de publicação*	*Área e Característica do trabalho*	*O que pode ser utilizado na atual pesquisa*
Elderly People Education: Social Effectiveness and Personal Significance	2015	Sokranyaeva (2015)	Tatiana VilevnaSokhranyaeva	*Lomonosv Moscow State University, Russian Federation*	**Área**: Educação e sociologia. Demonstra o papel do idoso na atual sociedade russa.	Em comparação ao cenário brasileiro, demonstra soluções para problemas semelhantes.
Information Society and the Mechanisms of Social Adaptation of the Elderl	2017	Salova (2017)	Tamara L. Salova	*Sochi State University, Russian Federation*	**Área**: Informação e educação. Demonstra a importância da inclusão de pessoas idosas na sociedade da informação.	Demonstra a importância da inclusão digital em pessoas idosas.

Capítulo 3

METODOLOGIA

Com o objetivo de desenvolver um sistema para os idosos da UnAPI, o método consistiu em dividir a implementação em cinco etapas, representadas em fluxograma conforme Figura 7, que serão descritas em detalhes a seguir. Dessa forma, nelas foram definidos o perfil dos usuários, as principais áreas envolvidas, os parâmetros e dados para o sistema, entre outros.

Figura 7 - Fluxograma de Trabalho



Fonte: Autor, 2022

Etapa 1: Realizar a escolha de um *framework*, linguagem de programação e banco de dados

Observada todas as características do sistema foi realizado a escolha pelo *framework* e banco de dados que atenda a demanda do INISA. Os *frameworks* são estruturas compostas por um conjunto de códigos genéricos que permite o desenvolvimento de sistemas e aplicações. Salienta-se que a utilização de *frameworks* acelera o desenvolvimento de aplicações, com personalização genérica do processo, deixando o foco principal do desenvolver na solução do problema da aplicação. Entender qual *framework* é o ideal para o projeto é uma etapa importante e que requer atenção, já que implementar o errado pode envolver muito tempo e gerar problemas.

Etapa 2: Mapeamento do SISUNAPI

Foram realizadas reuniões, através de videoconferências devido a isolamento imposto pela pandemia de COVID 19 (infecção respiratória aguda causada pelo coronavírus SARS-CoV-2, potencialmente grave, de elevada transmissibilidade e de distribuição global. 2021, Ministério da Saúde), onde foram realizados o mapeamento das funções necessárias para o funcionamento do sistema.

Uma das principais etapas no desenvolvimento de *software* é a definição do usuário que o utilizará. Segundo Spínola (2008), 12,4% dos fatores críticos relacionados ao desenvolvimento de *software* estão ligados na criação incorreta dos usuários. Outros fatores que influenciam o sucesso na conclusão dos projetos de *software* estão ligados a correta obtenção dos requisitos, falta de recursos, expectativas irreais, mudanças de requisitos e especificações, entre outros.

> Frustação e ansiedade são parte da vida diária de usuários de sistemas de informação computadorizados. Eles lutam para aprender a linguagem de comando ou sistemas de seleção de menus que se destinam a ajudá-los a fazer seu serviço. Algumas pessoas deparam com casos tão sérios de susto com computador, terror de terminal ou neurose de rede que evitam usar sistemas computadorizados (PRESSMAN, 2002, p. 393).

Segundo Pressman (2002) o foco no usuário deve ser pensado desde a concepção do projeto do *software*. Dessa forma, deve-se seguir os seguintes passos para garantir sucesso nos projetos: colocar o usuário no controle, reduzir a carga de memória do usuário e projetar uma interface consistente. Esses três passos são definidos por Theo Mandel como “regra de ouro”.

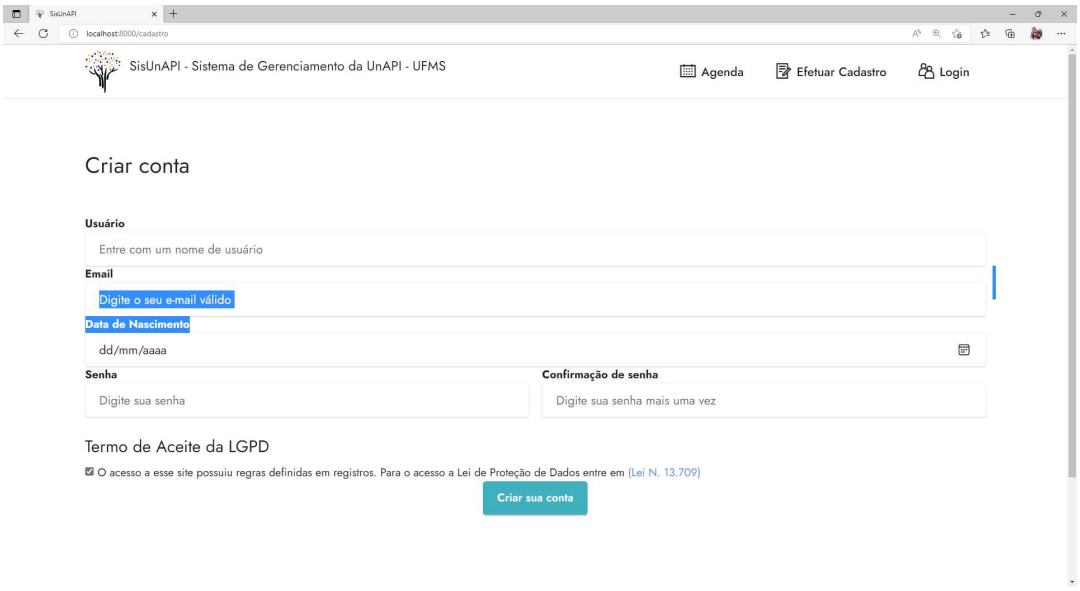
Segundo Barbosa *et. al.* (2008) não é possível medir de forma absoluta a usabilidade de um sistema e nem desenvolver um artefato de sistema completamente "usável" sem qualificar com maior precisão para qual perfil de usuário é mais adequado a tarefa. Ainda segundo o autor, um sistema que não seja criado de modo centralizado no usuário, precisamente no ponto de vista de utilização e lógica de uso, torna-se frequentemente obstáculo na realização da tarefa gerando frustação e resistência ao uso.

Ainda segundo Barbosa *et. al.* (2008) pode-se entender como inclusão digital do idoso a necessidade para que o sujeito se mantenha como protagonista de suas atividades cotidianas, sem depender do auxílio de outras pessoas.

Levando como princípio a solicitação de dados mínimos, o cadastro de usuário solicitará dados como nome de usuário, e-mail, data de nascimento e senha. Após o cadastro do usuário no sistema, é necessário o complemento de dados, fornecendo Nome Completo, CPF, RG e número de Celular. O cadastro do grupo de usuário é delimitado por um modelo de classificação interno, ajustável conforme a necessidade da UnAPI. Por padrão, todo o usuário cadastrado é considerado um usuário comum, sendo necessário posterior ajuste na área administrativa.

A proposta de solicitar o mínimo de dados para o cadastro é vital para a proposta do trabalho, tendo em vista que o principal usuário do sistema será: pessoa idosa, com baixa experiência em utilização de sistemas informatizados e com certa deficiência visual, causada pela idade apresentada. Com a definição do tipo de usuário, novos campos serão apresentados para complemento dos dados de cadastro. A Figura 8 apresenta a tela de cadastro de usuários no protótipo criado para testes.

Figura 8 Tela de Cadastro de Usuário



Fonte: Autor, 2022

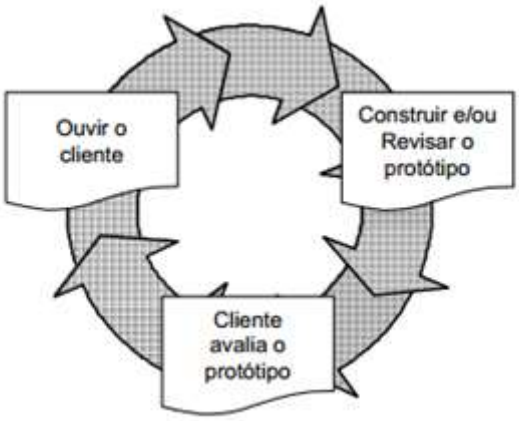
Etapa 3: Definir os principais requisitos do sistema

Segundo Spínola (2008), as definições encontradas na literatura técnicas para requisito:

- é uma característica do sistema capaz de atingir um objetivo;
- é definido por funções e restrições;
- é uma propriedade de um problema que deve ser resolvido.

Ainda segundo Spínola (2008), a definição dos requisitos é importante para delinear o que o cliente espera e o que o *software* e oferecerá. Também é através dos requisitos que é possível realizar a validação do produto final. O paradigma de prototipagem é apresentado na Figura 9, onde deve-se primeiramente ouvir o cliente, para construção e/ou revisão do protótipo que será avaliado posteriormente pelo cliente.

Figura 9 O paradigma de prototipagem



Fonte: Pressman, 2002.

Conforme a descrição do problema apresentado pela equipe da UnAPI, em especial as oficinas oferecidas aos idosos, não há um banco de dados que armazene as atividades realizadas pelos participantes das oficinas. Esses participantes ficam isolados, cada qual em uma oficina ou disciplina.

Os docentes não possuem um sistema que realize a unificação dos dados e, posteriormente, emita *feedback* sobre o aproveitamento de cada participante, bem como as atividades realizadas.

Os pesquisadores não possuem um repositório, onde possam realizar pesquisas para a extração de dados para análises objetivas. Todo esse arcabouço de informações se encontra

disperso em diversos arquivos dentro da UnAPI. Isso porque há documentos armazenados de forma analógica, em pastas físicas com formulários impressos.

A função da criação do banco de dados é a digitalização e centralização das informações em um único local. Dessa forma, os idosos que participam das oficinas tem acesso a suas atividades. Os docentes têm um local facilitado tanto para disponibilizar o conteúdo ministrado e um local para o *feedback*. Os pesquisadores têm um local para realizar a coleta de informações e, com estudos específicos, encontram maneiras de realizar melhorias em processos.

a. Requisito 1 – Perfil do usuário

Definiu-se três tipos de acessos distintos: aluno, pesquisador e técnico;

- O perfil aluno, é o perfil público do sistema, todo o usuário que realizar um cadastro será automaticamente cadastrado nesse perfil, podendo ser remanejado para outro através da área administrativa do sistema, realizado pelo administrador do sistema. O perfil não possui acesso a área administrativa do site;

- O perfil Pesquisador, tem acesso a listagem de clinicas, esportes e cursos. O perfil tem acesso a área administrativa do site, porém não tem permissão de exclusão de dados;

- O perfil técnico, tem acesso as funções do sistema sem possibilidade de exclusão de dados.

Além dos usuários listados, haverá um perfil administrativo, que é responsável pela manutenção geral do sistema, e terá acesso a todas as telas disponíveis e a todos os níveis existentes.

b. Requisito 2 – Armazenamento dos dados

A Principal função da criação do sistema é o armazenamento de dados. Todos os perfis de usuários têm acesso a um repositório para armazenamento de dados. Esse armazenamento é utilizado para o compartilhamento, seja de documentos, apostilas, vídeos, como também de troca de informações de atividades complementares que são oferecidos para os participantes dos cursos e oficinas da UnAPI.

Etapa 4: Estruturação do Banco de Dados

Após a definição do projeto e o que deveria ser realizado, foi implementado a criação de uma base de dados com as características necessárias ao *framework* escolhido. Foi realizado a configuração deste para a criação das tabelas administrativas, relativas aos acessos dos usuários, criação de *logs* de acesso e monitoramento, bem como toda a área administrativa do *framework*. Com a criação da base de dados, houve também a criação dos ajustes de segurança, nativos do próprio *framework* e listados com um dos principais requisitos para o sistema.

Durante a configuração inicial do *framework*, suas tabelas gerenciais são automaticamente criadas. Uma das principais vantagens na utilização de *frameworks* é a otimização durante a codificação. A criação do protótipo, utilizando o *framework*, acelerou o processo de criação das tabelas no banco de dados, uma vez que a própria codificação criou as estruturas necessárias no banco de dados.

Etapa 5: Protótipo

Após a criação da estrutura do banco de dados e ajuste inicial do *framework*, foi projetado uma interface de usuário e sua versão de *frontend*.

Segundo Pressman (2002) apesar de problemas que possam ocorrer durante os testes do protótipo, essa passa a ser um efetivo paradigma de engenharia de *software*. Com o objetivo de acelerar os passos para a entrega do produto, através do protótipo é possível avaliar os itens que já possuem as funcionalidades ativas e definir, através de indicações, o que deve ser melhorado e desenvolvido para o produto de *software* final.

A criação do protótipo foi validada com dados fictícios.

Capítulo 4

CODIFICAÇÃO E BANCO DE DADOS

Nesse capítulo são apresentados e descritos como o banco de dados foi desenhado e estruturado. É apresentada a linguagem escolhida para o desenvolvimento do sistema, o *framework* e a codificação realizada.

4.1 – *Definição do Framework*

Foram realizadas pesquisas sobre os melhores *frameworks* existentes.

O *framework Titan* desenvolvido por Carromeu (2016) apresenta as funcionalidades necessárias para o desenvolvimento da ferramenta SISUNAPI, todavia sua implementação foi realizada em PHP (*Hipertext Preprocessor*) a qual não dificulta a adição de funcionalidades para novos projetos de pesquisa. Sua comunidade virtual é diminuta, concentrando-se inicialmente as pessoas ligadas ao desenvolvimento do *framework* e, posteriormente a Empresa de Pesquisa Agropecuária (EMBRAPA) no Centro de Desenvolvimento de Gado de Corte em Campo Grande.

O *Joget* é um *framework* desenvolvido em Java de fácil implementação que fornece além do desenvolvimento de uma aplicação para o Banco de Dados também oferece suporte pela comunidade virtual. Disponibilizado para implantação nos principais servidores *cloud* é uma grande ferramenta de desenvolvimento. Disponível em versões para os desenvolvedores, sua versão grátis oferece suporte apenas pela comunidade virtual. Um dos grandes empecilhos para a escolha do *Joget* é relativo aos sistemas já existentes. Em uma busca por sistemas implantados utilizando a ferramenta, foram encontrados poucos itens, em sua maioria disponibilizada fora do território nacional (Joget, 2022).

Para evitar a falta de adaptabilidade e manutenção da ferramenta desenvolvida, foram realizadas buscas por um novo *framework* que atenda a necessidade e que possua uma comunidade virtual ativa. O *framework Django* foi desenvolvido em linguagem *Python* para a *web* na geração de aplicativos seguros, de forma rápida e dinâmica. Conta com uma grande comunidade virtual distribuída, gerando troca de informações a respeito deste *framework*. A utilização de *frameworks* acelera o desenvolvimento de aplicações, com personalização genérica do processo, deixando o foco principal do desenvolver a solução do problema da aplicação.

O *framework Django* possui diversas ferramentas de ligação com Banco de Dados. Nativamente o *framework* utiliza *SQLite*. O *SQLite* é um banco de dados desenvolvido em linguagem C onde sua principal característica é a simplicidade de administração, implementação e manutenção. Sua vantagem é a não exigência de um Sistema de Gerenciamento de Banco de Dados (SGBD) para a sua execução. O *framework Django* utiliza o padrão de projeto MTV (M – *Model*, T – *Template*, V – *View*). O *Model* é responsável pelo gerenciamento dos dados, o *Template* é responsável pelo gerenciamento de entrada e saída e *View* que é responsável pela saída gráfica e textual. O *Django* permite a criação de páginas de maneira rápida. É guiado pela premissa DRY (do inglês *Don't Repeat Yourself*) e permite o reaproveitamento de códigos com pequenas alterações (ByLearn, 2020)

Como o *Django* permite a escolha pelo melhor gerenciamento de Banco de Dados, foi realizado a opção pelo *PostgreSQL*. O *PostgreSQL* é um gerenciador altamente utilizado pela comunidade virtual, com grande utilização não apenas por projetos públicos como também por grupos corporativos. A escolha pelo *PostgreSQL* também está ligada a grande quantidade de dados a serem inseridos no sistema e pelas características existentes no gerenciador para tratamento de dados. É possível realizar a inserção de funções específica e criação de gatilhos e visões para a manipulação dos dados presentes.

O *framework Django* possui a versatilidade de alternância entre os bancos de dados existentes. Em um ambiente de desenvolvimento é possível utilizar um banco de dados simplório, com menor carga de dados, como o *SQLite*. Já em ambiente de produção, é possível a configuração de um banco mais robusto como o *PostgreSQL* para preservar a segurança dos dados inseridos (Lucas, 2021). Como o foco principal do projeto foi a modelagem de um banco de dados para o armazenamento das informações, a escolha por um *framework* que agilize o desenvolvimento da aplicação foi vital para o sucesso do conteúdo entregue.

A escolha pelo *Framework Django* foi motivada pela linguagem de programação utilizada em usa implementação. O *Python* é uma linguagem de uso geral muito utilizada na

área de ciências de dados. Uma das principais características da linguagem é a sua legibilidade, o que a diferencia das outras linguagens existentes. Sua simplicidade destaca-se o que facilita o aprendizado e a manutenção futura. Sua portabilidade é outra característica marcante, pois uma codificação realizada em um Sistema Operacional é facilmente portada para outro Sistema sem muito desgastes por conta de configurações. A especificação da linguagem *Python* é mantida pela *Python Software Foundation* (PSF) (Franco, 2016).

4.2 – Definição da Linguagens de Programação

Foram realizadas diversas pesquisas pela linguagem de programação de melhor manutenção disponível atualmente no mercado, todavia, para a escolha de uma determinada linguagem levou-se em evidência a comunidade virtual disponível, a amplitude de ferramentas desenvolvidas com a linguagem e requisitos básicos de segurança.

A escolha pela linguagem *Python* deve-se a sua facilidade em manutenção e portabilidade. Sua comunidade virtual é ativa, o que possibilita que a maioria das aplicações desenvolvidas terá fácil adaptabilidade.

A escolha também foi motiva pelas diversas ferramentas de manipulação de dados, nas áreas de Inteligência Artificial e Engenharia de Dados. Tais ferramentas, em união com as informações que devem ser armazenadas, garantem um grande repositório para novos projetos de pesquisa ligados à área de saúde e bem estar voltados para a pessoa idosa.

4.3 – Definição do Banco de dados

Assim como todo o sistema de gerenciamento de informações, garantir que os dados estejam interligados e conectados de forma correta é a maneira de garantir que dados pertencentes a um usuário não sejam manipulados por usuários sem autorização. Considerando que, para a interação no sistema seja necessário que o usuário esteja devidamente cadastrado e logado e que, a cada login realizado, uma sessão for criada única e exclusivamente para o

usuário, garantimos que nenhuma informação será repassada de forma não autorizada para um terceiro.

Para o desenvolvimento do SISUNAPI foi utilizado o *PostgreSQL* em sua versão 10.0. O *PostgreSQL* é um sistema gerenciador de banco de dados objeto-relacional, desenvolvido na Universidade da California no Departamento de Ciências da Computação em *Berkeley*, o qual foi pioneiro em diversos conceitos atualmente difundidos e utilizados por diversos outros bancos de dados. É um projeto *open-source* e possui uma comunidade virtual ativa que trabalha em melhorias constantes.

Durante os estudos para escolha da linguagem de programação para o desenvolvimento do SISUNAPI foram observados diversos SGBDs que poderiam atender as expectativas de robustez e segurança nos dados. Foi realizado um estudo com o *MySQL* e com *SQLite*. Ambos os SGBDs atenderiam os requisitos. A escolha pelo *PostgreSQL* deu-se pelos diversos projetos já existentes na comunidade universitária. A escolha garante que novos projetos possam ser realizados e inclusos no SISUNAPI.

4.4 – *Mapeamento do SISUNAPI*

Atualmente, a UnAPI disponibiliza para a comunidade externa à UFMS diversas oficinas, cursos e atendimento ambulatorial para as pessoas idosas participantes de seus projetos.

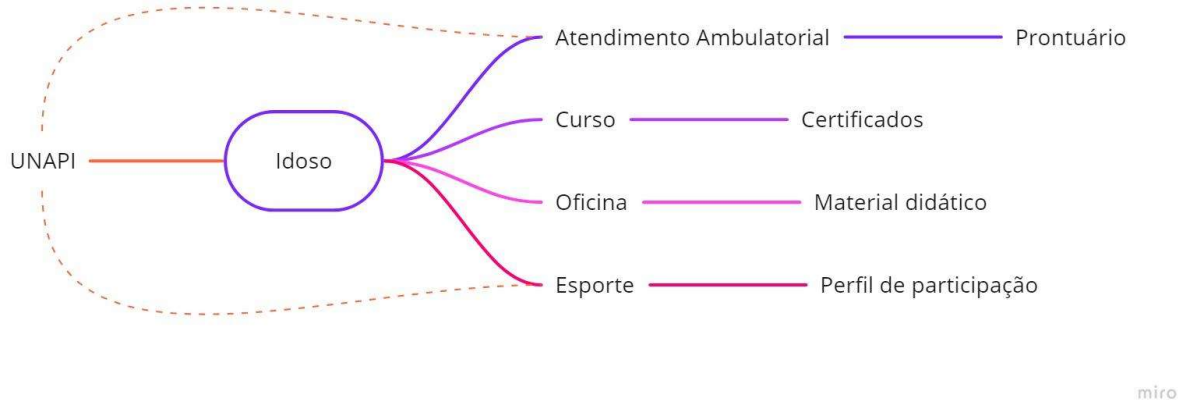
O problema principal é que toda a informação gerada pelos atendimentos não é centralizada, e muitas informações geradas são armazenadas em formulários de papel ou de forma não centralizada, pelos pesquisadores, professores e alunos, causando grande possibilidade de perda de dados alteração, devido a não gerencia dos mesmos.

Depois de compreendido o problema apresentado pela UnAPI, foi proposto o desenvolvimento de um banco de dados central, o qual será responsável pelo armazenamento e gerenciamento das informações dos participantes dos diversos projetos desenvolvidos pela UnAPI.

Como o foco principal do problema é o usuário final, sendo ele pessoa idosa, participante dos projetos desenvolvidos pela unidade setorial, o sistema apresenta as seguintes características: ter acesso as agendas de atendimento ambulatorial e prontuários; acessos aos cursos e certificados disponibilizados; acesso as oficinas e materiais didáticos disponibilizados

e agendas de esportes, com definição do perfil de participação de cada usuário (Figura 10). Por outro lado, o armazenamento dos arquivos por parte dos pesquisadores e professores garantindo assim um repositório de informações para futuras novas pesquisas.

Figura 10 - Visão Geral do Acesso do Usuário ao Sistema



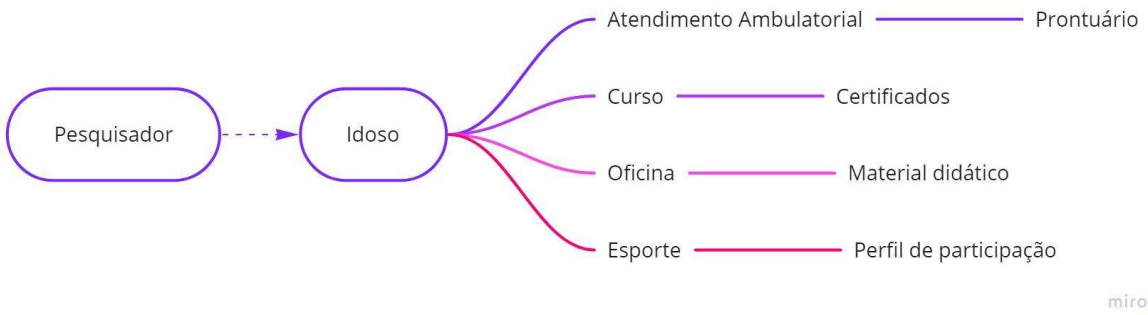
Fonte: Autor, 2022

A proposta principal do projeto é garantir que todo material obtido seja armazenado em um local único e seguro, que os participantes tenham acesso aos seus dados mediante cadastro prévio e que seja possível ter acesso às novas oficinas ou atendimentos ambulatoriais.

Ao mesmo tempo em que os idosos necessitam de um local para acesso de suas informações, os pesquisadores e alunos necessitam das informações de seus assistidos. Todo esse universo de interações deve ser protegido e salvaguardado por criptografia, tendo em vista a presença dos dados de saúde, presentes nos mais diversos registros ambulatoriais. Dessa forma, conforme a Figura 11, o pesquisador também terá acesso aos dados dos idosos, conforme permissões.

Os participantes da UnAPI podem ou não estar vinculados a uma atividade. Para serem considerados participantes da UnAPI é necessário realizar um registro e ter um usuário ativo no sistema.

Figura 11 Visão do Pesquisador ao Sistema



Fonte: Autor, 2022

Para realizar o registro inicial no sistema (Figura 12), o usuário deve fornecer dados básicos necessários para o primeiro acesso (*login*). Após realizado o primeiro acesso é direcionado para o registro de dados complementares ao sistema, necessários para a sua correta identificação.

Figura 12 Tela Inicial do SISUNAPI, versão Desktop

Fonte: Autor, 2022

Toda a movimentação realizada dentro do SISUNAPI é registrada dentro do banco de dados do sistema. Para um usuário conseguir realizar interação nas informações do sistema é necessário que o mesmo esteja conectado (logado), garantindo a segurança das informações.

4.5 – *Estruturação do Banco de Dados*

A escolha pelo *Framework Django* para a criação do SISUNAPI foi devido ao seu poderoso motor de ligação a banco de dados. O *Object-Relational Mapping* (ORM), em português mapeamento objeto-relacional, nativo do *Django* permite a criação, gerenciamento e migrações de banco de dados, juntamente com a criação e visualização de formulários de maneira rápida (Bylean, 2020).

O *Django*, em sua estrutura inicial, fornece um poderoso painel de controle principal, que fornece diversas ferramentas de gerenciamento de usuários, incluindo ferramentas criptografia de senhas, essenciais para projetos que envolvam dados e que estejam ligados à internet. A Figura 13 apresenta a estrutura da tabela de usuários do SISUNAPI.

4.5.1 A tabela *auth_user*

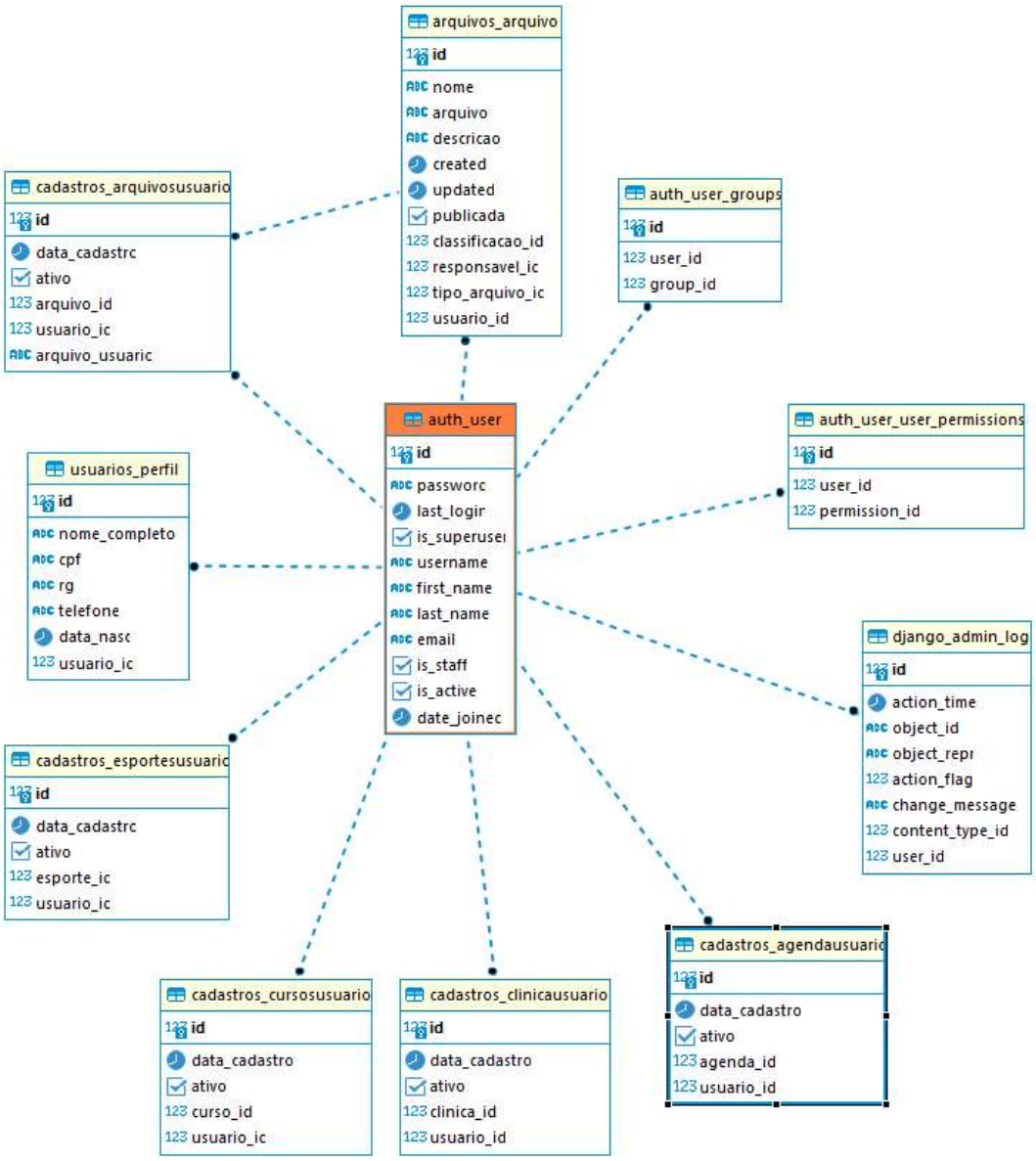
Observando a Figura 13 é identificado a forte ligação da tabela *auth_user* com diversas outras do sistema. Essa tabela é considerada principal, pois é nela realizado o registro dos usuários, sendo esses administradores de sistema, professores, pesquisadores, técnicos ou usuários, o qual definimos como idosos. Sua ligação é realizada principalmente pelo ID (código identificador), sendo utilizado como chave estrangeira nas suas ligações com outras tabelas do sistema. Uma chave estrangeira é a ligação de um campo presente em uma tabela do banco de dados que possui relacionamento com outro campo indexado (Wikipédia, 2022). Chaves estrangeiras são referências de índices de outras tabelas que possuem ligação com os dados ao qual está inserida. Observando a classe *arquivos_arquivos*, ela faz referência a uma chave estrangeira na tabela *auth_user*, isso significa que, para cada registro referenciado nessa tabela, deverá haver um *id* de usuário vinculado. Esse vínculo é realizado através da chave estrangeira.

A tabela *auth_user* é uma tabela de sistema do *Framework Django*. Considerando essa informação, é aceito que a segurança sobre os dados ali presentes seja segura. Sua estrutura é composta por *password*, *last_login*, *is_superuser*, *username*, *first_name*, *last_name*, *email*, *is_staff*, *is_active* e *date_joined*. Os principais dados utilizados são *password* (senha), *username* (nome de usuário), *first_name* (primeiro nome), *last_name* (último nome), *email* (e-mail de cadastro) e *is_superuser* (é superusuário).

Considerando que as informações estão presentes em nossas vidas e fazem com que direcionamentos sejam tomados (Ferreira, 1999), foi acrescentado uma tabela ao sistema para

que os usuários do SISUNAPI fazem a inserção dos dados complementares ao sistema, porém vitais para futuras pesquisas.

Figura 13 Estrutura da Tabela de Usuários do SISUNAPI



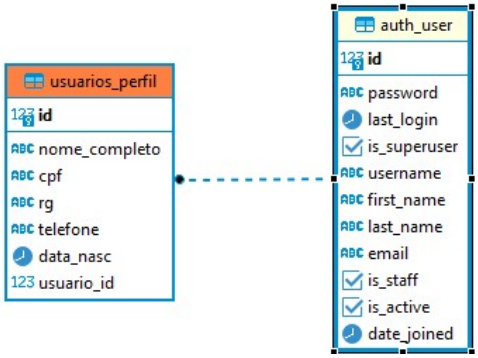
Fonte: Autor, 2022

4.5.2 A tabela usuarios_perfil

A Figura 14 apresenta a estruturação da tabela *usuarios_perfil* e sua ligação a tabela *auth_user*. Em linhas gerais, todo o usuário do sistema SISUNAPI deverá possuir um perfil de usuário devidamente cadastrado. A tabela *usuarios_perfil* é composta por *nome_completo*, *cpf*,

rg, *telefone*, *data_nasc* e *usuario_id*. O *usuário_id* é chave estrangeira de ligação com a tabela *auth_user*. Em definição, cada usuário terá um perfil e cada perfil terá um usuário. Sua relação de cardinalidade é 1 para 1.

Figura 14 Tabela para Cadastro de Perfil do Usuário



Fonte: Autor, 2022

Durante o levantamento de informações para o desenvolvimento do SISUNAPI houve a identificação da necessidade de criação de uma estrutura de dados que seria responsável pelo registro das pessoas responsáveis envolvidas nos projetos. Essas pessoas não seriam necessariamente usuários do sistema, mas devem possuir seus dados devidamente registrados para serem localizados e vinculados as atividades cadastradas no sistema.

4.5.3 A tabela pessoa_pessoa

Na Figura 15 identifica-se a tabela *pessoa_pessoa*, responsável pelo armazenamento dos dados das pessoas responsáveis pelos projetos do sistema. Essa tabela é composta por *nome*, *email*, *data_nasc*, *cpf*, *rg*, *cep*, *num_cep*, *foto*, *data_cadastro*, *tipo_pessoa_id*.

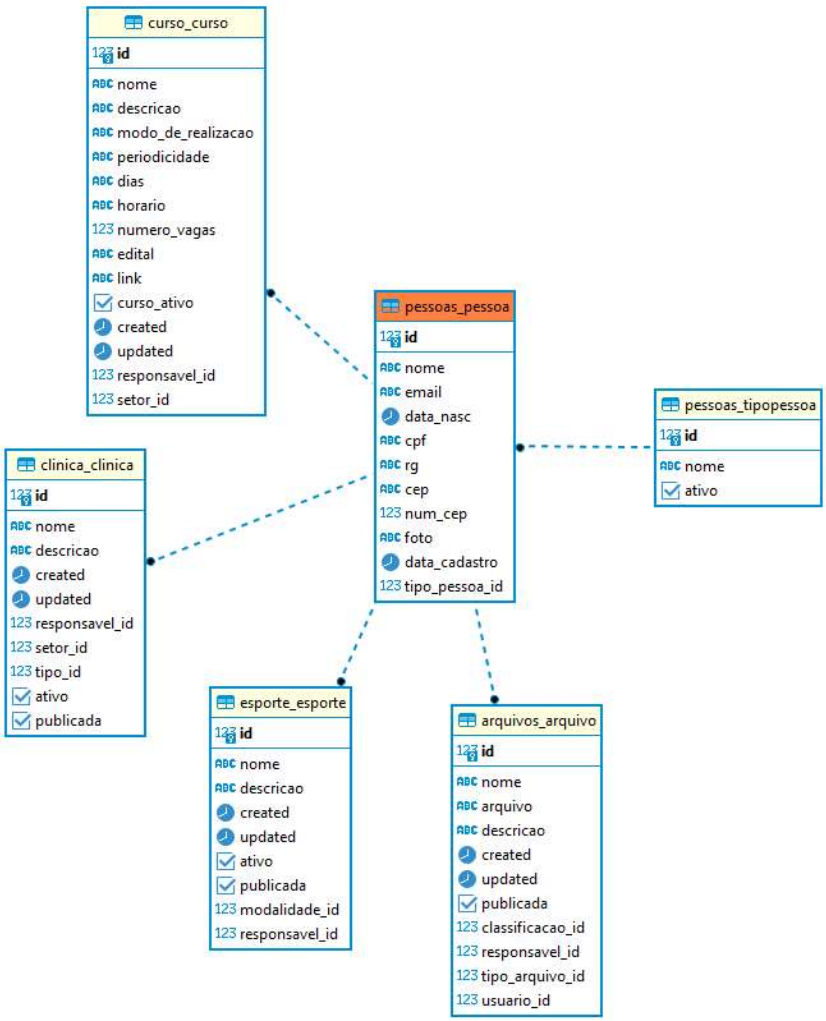
Conforme a descrição de cada item da composição da tabela podemos notar que há o registro do *CEP* do endereço da pessoa e o *NUM_CEP* o qual é o número do endereço correspondente ao *CEP* fornecido. Há também um campo *FOTO*, onde é possível aos usuários a inserção de uma imagem, que será adicionada aos arquivos de mídia do sistema e o caminho registrado na tabela no campo *FOTO*.

É possível verificar a forte ligação da tabela *pessoa_pessoa* com outras principais existentes (Figura 15). As tabelas *curso_curso*, *clinica_clinica*, *esporte_esporte* e *arquivos_arquivo* são utilizadas pelo sistema para registrado de cada curso, registro clinico, atividade esportiva, pesquisa ou arquivo relacionado a um usuário o qual a pessoa registrada é responsável.

Há ainda a tabela *pessoas_tipopessoa* o qual defini, dentro uma relação pré-existente, a qualificação da pessoa dentro do sistema. Essa qualificação, realizada no levantamento de informações pode ser definida como: professor, pesquisador, auxiliar, técnico, aluno ou assistente. Essa qualificação é totalmente ajustável e configurável pelo administrador do sistema, no painel administrativo do sistema.

Durante o levantamento realizado, através de videoconferências com os envolvidos no projeto, e avaliação visual do site da UnAPI, foi identificado uma necessidade da UnAPI: deixar registradas as pesquisas realizadas pelos diversos pesquisadores, professores e alunos que são armazenadas por diversos meios. A necessidade primaria é a criação de um banco de dados para armazenamento dos dados já existentes de pesquisas realizadas durante anos anteriores, garantindo o acesso, mediante registro e a autorização da inserção das informações, com agilidade e segurança.

Figura 15 Tabela de Cadastro de Pessoas



Fonte: Autor, 2022

4.5.4 A tabela arquivos_arquivo

A tabela *arquivos_arquivo* é a responsável pelo armazenamento dos arquivos no repositório de mídias do sistema. A tabela é composta por *nome*, *arquivo*, *descricao*, *created*, *updated*, *pulicada*, *classificacao_id*, *responsável_id*, *tipo_arquivo_id*, *usuário_id*.

Para inserção de um arquivo no repositório de mídias do sistema, é necessário realizar a inserção de um nome para o arquivo, sua descrição, relacionar uma classificação para o arquivo, um responsável e um tipo de arquivo e se o arquivo é ativo ou não no sistema.

O tipo de arquivo é escolhido através de uma listagem pré-definida, estabelecida no desenvolvimento do sistema, sendo possível acréscimo, adição ou remoção, tendo como opções artigo cientifico, relatório de participação, laudo médico, exame médico, certificado e atestado de participação.

Todo arquivo cadastro necessita obrigatoriamente estar vinculado a uma pessoa responsável, e o sistema automaticamente definirá o arquivo como de propriedade do usuário que está realizando a sua inserção no sistema.

Observando a Figura 16, é possível verificar os diversos vínculos existentes durante a criação de um registro de arquivo no sistema. Para o registro é necessário que o mesmo esteja vinculado a um usuário do sistema, que haja uma classificação e um tipo já cadastrado. Toda essa estrutura foi desenvolvida para garantir a integridade dos dados e futuras indexações.

4.5.5 A tabela agenda_agenda

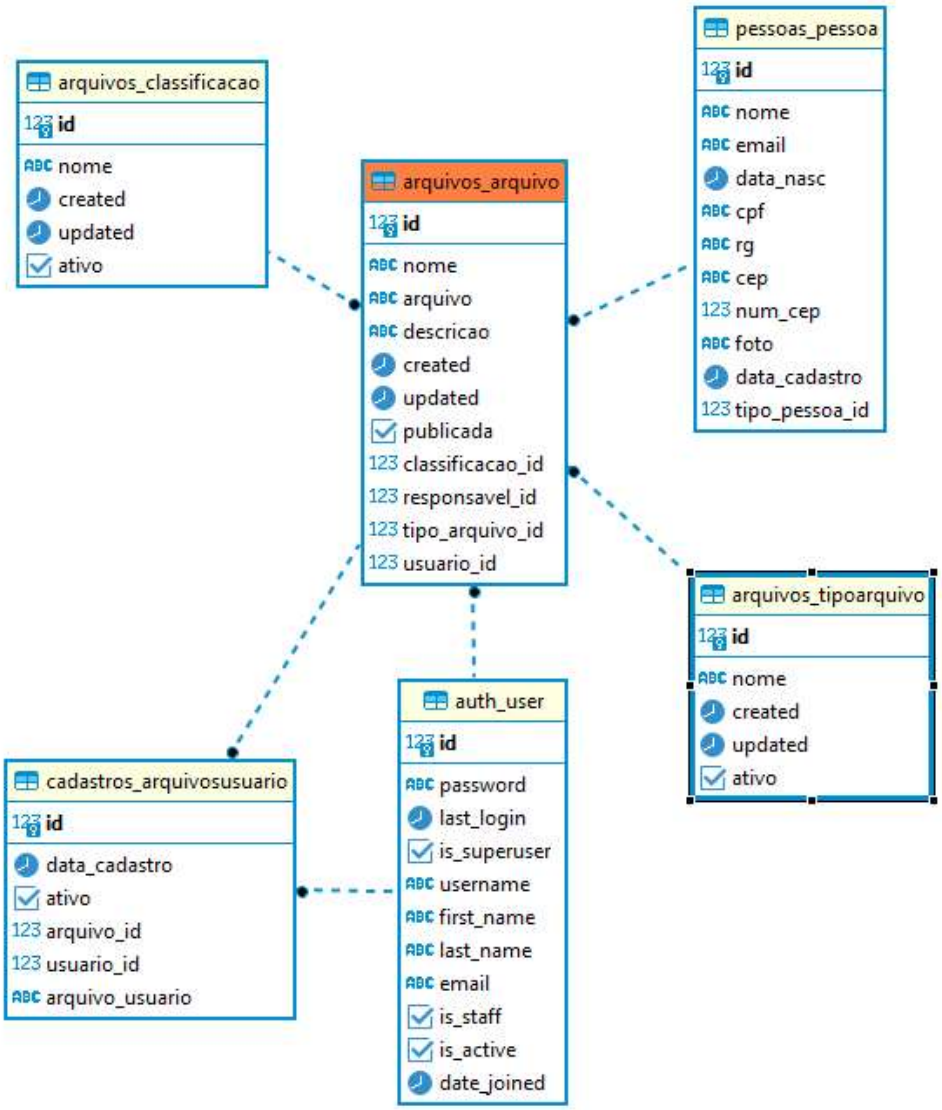
Durante o levantamento de informações para o desenvolvimento do sistema, foi identificada a necessidade da apresentação dos recursos que a UnAPI oferece aos seus participantes.

Levando em consideração as informações que devem ser apresentadas foi criada a tabela *agenda_agenda* que é responsável pelo cadastro de todas as atividades disponíveis aos usuários participantes da UnAPI.

A tabela *agenda_agenda* é composta por *nome_atividade*, *descrição_atividade*, *responsavel*, *data_inicio*, *data_fim*, *numero_vagas*, *edital*, *link*, *data_publicacao* e *publicada*. Toda a agenda cadastrada possui um responsável previamente cadastro no sistema.

A atividade cadastrada na agenda pode possuir número de vagas, edital com maiores informações ou um *link* de um *site* onde há maiores informações, todas essas informações podem ser inseridas nos campos específicos.

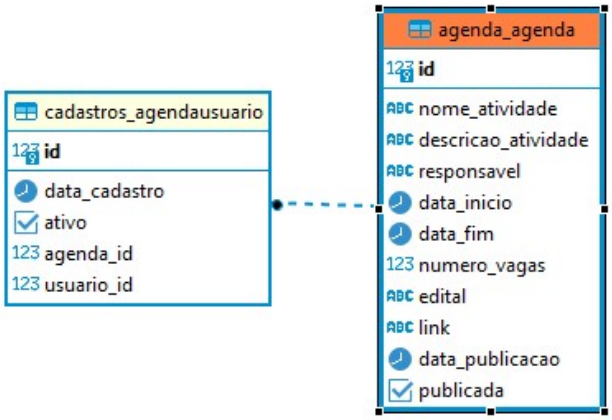
Figura 16 Tabela de Registros de Arquivos



Fonte: Autor, 2022

Há também uma opção onde é possível disponibilizar a exibição da atividade. As atividades cadastradas na tabela *agenda_agenda* são disponibilizadas de forma pública, na página principal do sistema, desde que estejam marcadas para publicação, conforme Figura 12.

Figura 17 Tabela de Cadastro de Atividades



Fonte: Autor, 2022

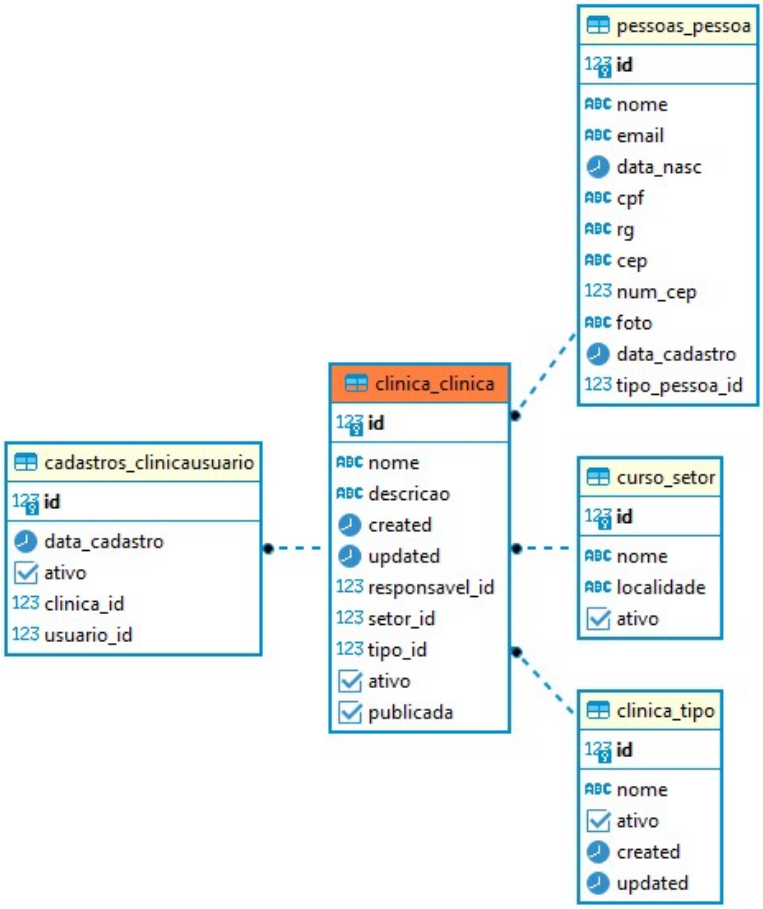
4.5.6 A tabela clinica_clinica

Durante o levantamento de informações para a construção do sistema foi verificado que muitos usuários da UnAPI realizam atendimento nas clínicas disponíveis. Esses atendimentos não eram registrados, ficando as informações geradas pelos atendimentos perdidas.

Como atualmente não há registro dos atendimentos, não se sabe com precisão qual o número de usuários que utilizam dos recursos disponíveis e se há necessidade de aumento de oferta desses recursos.

Para o cadastro dos atendimentos criou-se a tabela *clinica_clinica* à qual será responsável pelo registro dos atendimentos clínicos dos usuários da UnAPI. Ela é composta por *nome*, *descricao*, *created*, *updated*, *responsavel_id*, *setor_id*, *tipo_id*, *ativo* e *publicada*, conforme Figura 18. Entre os itens de sua composição há o nome do atendimento, sua descrição, qual a data de criação (*created*) e data de possível atualização (*updated*), a pessoa responsável por esse atendimento, o setor responsável pelo atendimento, se o atendimento se encontra ativo, se está publicado no sistema e o tipo de atendimento pré-selecionado a partir do levantamento realizado, tendo como cadastro básico os atendimentos: radiologia, psicologia, fisioterapia, geriatria, clínica geral e odontologia. Assim como os outros registros realizados durante o levantamento, o cadastro dessas categorias poderá ser atualizado para melhor atender a UnAPI e seus usuários.

Figura 18 Tabela de Cadastro de Atendimentos Clínicos



Fonte: Autor, 2022

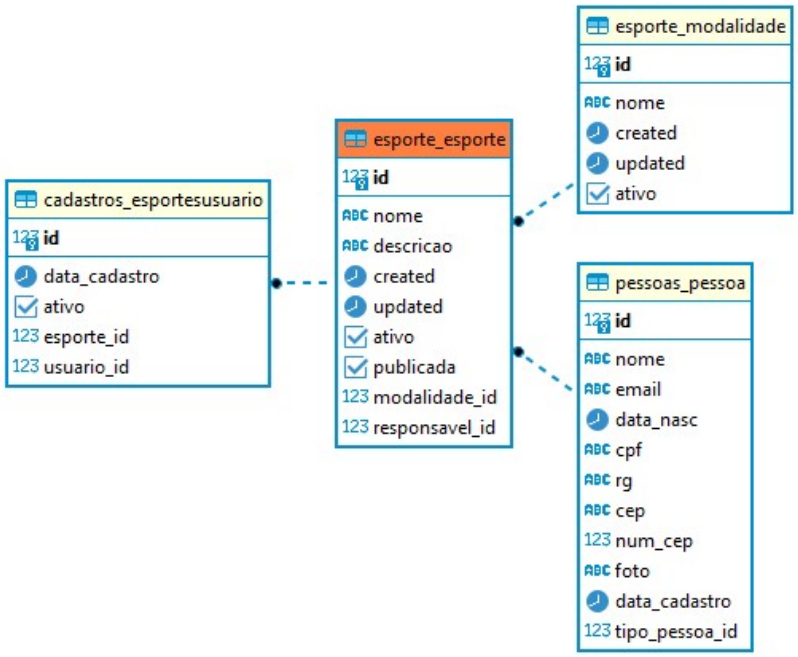
4.5.7 A tabela esporte_esporte

Ainda durante os levantamentos, foi identificado que muitos usuários praticam atividades esportivas vinculadas aos projetos da UnAPI. Muitas destas atividades geram informações que devem ser armazenadas e podem ser utilizadas em projetos de pesquisas futuras.

Para o armazenamento das informações relativas as atividades esportivas (Figura 19), foi criado a tabela *esporte_esporte*. Ela é composta por *nome*, *descricao*, *created*, *updated*, *ativo*, *publicada*, *modalidade_id*, *responsavel_id*. Para o cadastro é necessário o fornecimento do nome da atividade, uma descrição, o cadastro da criação (*created*) junto ao banco e se houve alguma atualização (*updated*), também há a informação se a atividade se encontra ativa no sistema e se está publicada.

Cada atividade esportiva possui uma modalidade previamente cadastrada. No levantamento inicial do sistema foram definidas algumas modalidades previamente cadastradas, sendo necessário a atualização da listagem para posterior vínculo dos participantes. Cada uma das atividades disponíveis deve ter uma pessoa responsável, para coordenação das atividades previamente cadastrada no sistema.

Figura 19 Tabela de Cadastro de Atividades Esportivas



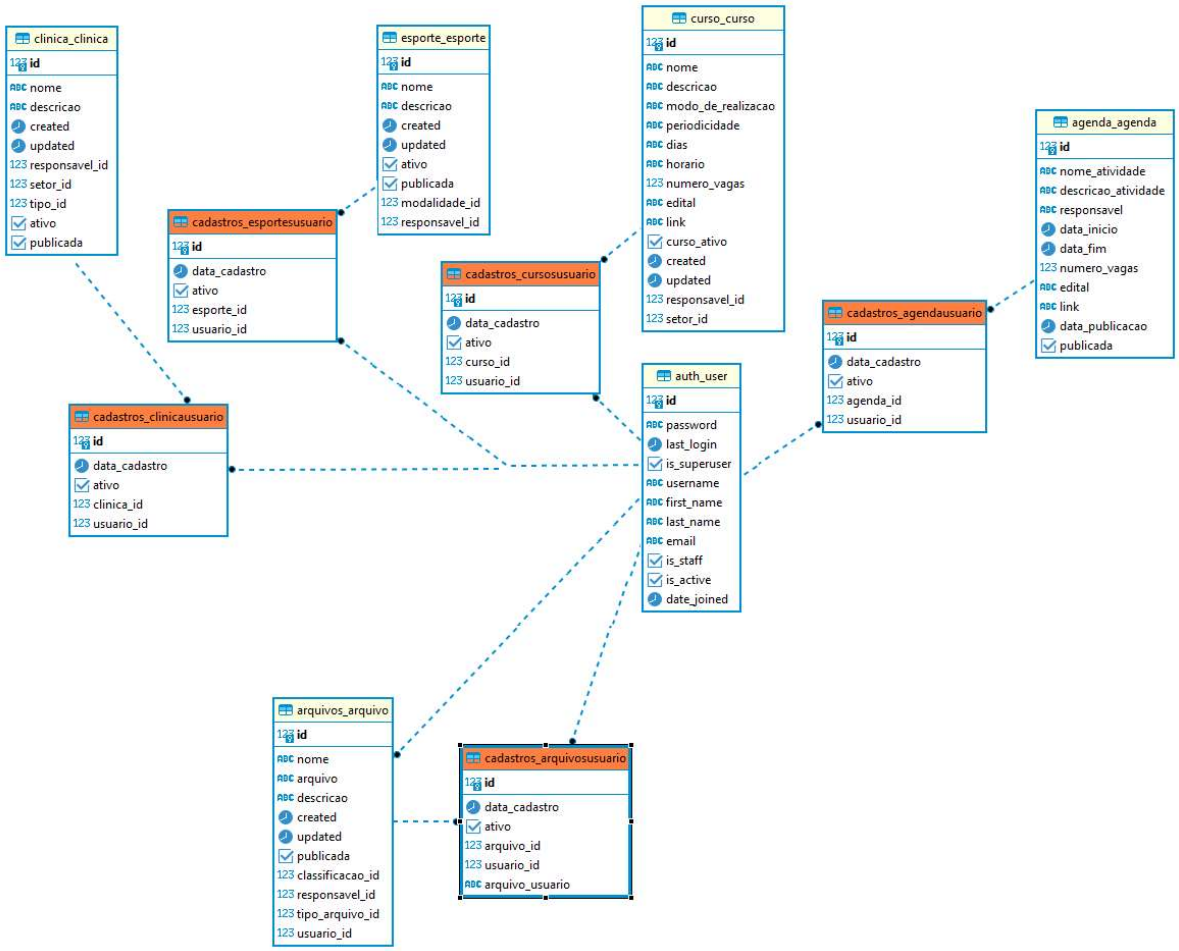
Fonte: Autor, 2022

Ainda para vínculo das atividades realizadas pelos usuários do sistema foram definidas tabelas de cadastros para cada módulo disponível. Cadastro de Atendimentos Clínicos, Cadastro de Atividades Esportivas, Cadastro de Atividades Gerais – Agendas e Cadastro de Arquivos, conforme pode ser observado na Figura 20.

4.5.8 As tabelas de vínculos dos cadastros

Para definição das interações dos usuários no sistema, foram criadas tabelas de cadastro dos itens disponíveis. Um item é um recurso disponível dentro do sistema o qual o usuário pode realizar interação. As tabelas criadas possuem estruturas semelhantes, pois são tabelas de auto-relacionamento, onde indicam que o usuário possui vínculo com determinado arquivo, curso, clinica, esporte ou agenda.

Figura 20 Visão Geral das Tabelas dos Cadastros do Usuário



Fonte: Autor, 2022

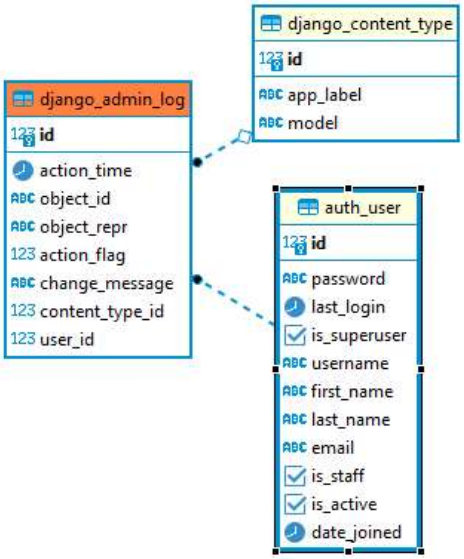
Cada tabela de cadastro é composta por *data_cadastro*, *ativo*, *item_id*, *usuario_id*. Para cada tabela de cadastro, o *item_id* está diretamente relacionado ao item que o usuário está cadastrando, seja ele agenda, curso, clinica, esporte ou arquivo. Durante o *login* em sua sessão de usuário, apenas os itens cadastrados para o usuário estarão disponíveis em sua sessão, por isso a importância do cadastro de cada item nas tabelas de auto relacionamento.

4.5.9 A tabela django_admin_log

O *framework Django* conta com uma robusta implementação de segurança e registros de atividades. Todas as transações realizadas são registradas em uma tabela especifica no banco de dados. A tabela *django_admin_log* é construída assim que a instância do administrador é configurada e todos os registros de acesso e modificações são ali armazenadas. A tabela é construída por *action_time*, *object_id*, *object_repr*, *action_flag*, *change_message*,

contente_type_id, *user_id*, conforme Figura 21. A tabela possui relação direta cm a *auth_user* pois, como já mencionado anteriormente, é nessa tabela onde são registrados os usuários do sistema.

Figura 21 Tabela de Registro de Eventos dos Usuários

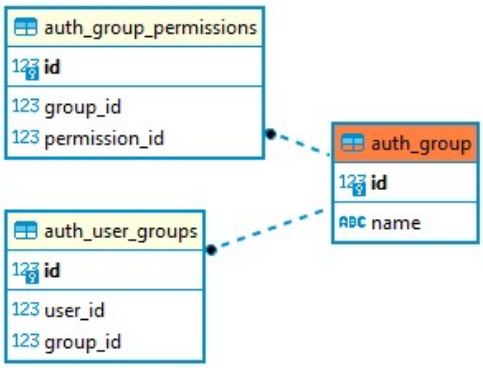


Fonte: Autor, 2022

4.5.10 Tabela auth_group

Ainda para manter os níveis de autorização dentro do sistema, o *framework Django* fornece, quando realizada a sua instância inicial, um nível de autorização ajustável e configurável para garantir os acessos aos usuários. Para tal, é criado um conjunto de tabelas que mantem um auto relacionamento que garante o registro das autorizações para cada usuário, conforme Figura 22. Dessa forma, são definidos grupos distintos e cada grupo possui suas autorizações.

Figura 22 Tabela com Configurações do Grupos de usuários



Fonte: Autor, 2022

4.6 – *Codificação*

Para a construção inicial do SISUNAPI foi realizado a verificação das dependências existentes no sistema operacional para o início do projeto através do *framework Django*.

Para o desenvolvimento foi utilizado a versão do *Python* 3.9.13. Pelas boas práticas de desenvolvimento de aplicações na linguagem *Python*, é indicado a criação de ambientes virtuais. Um ambiente virtual é um módulo que fornece suporte para a criação de diretórios próprios, operacionalmente isolados dos diretórios do sistema operacional que está hospedando o sistema. Os ambientes virtuais são chamados de *venv*, que nada mais é que um módulo operacional do *Python*. Cada ambiente virtual possuiu seu próprio binário *Python* e seu próprio conjunto de pacotes necessários para o correto funcionamento da aplicação.

A linguagem *Python* é baseada em pacotes que são interpretados pelo aplicativo executável *Python*. Ao sair ou entrar de uma instância, as referências passadas anteriormente são perdidas, sendo necessário a execução para o correto carregamento das configurações.

Uma grande vantagem da linguagem *Python* é o fato dela ser utilizada através de pacotes. Apenas os pacotes solicitados são alocados e executados, garantindo preservação de memória em execução. Tal procedimento garante sistemas equilibrados e leves nos servidores de hospedagem.

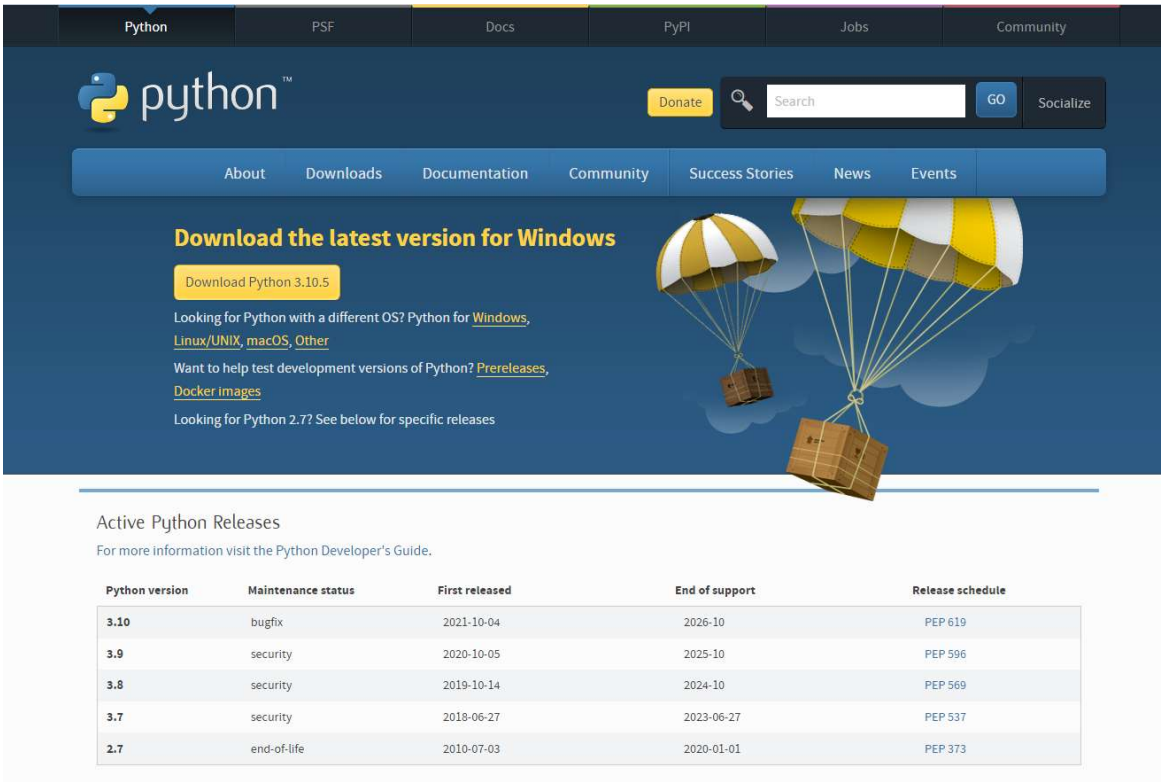
Para o desenvolvimento do SISUNAPI, foi utilizado um *Notebook Pentium* B950 2.10Ghz com 4Gb de memória RAM instalado com o Sistema Operacional *Microsoft Windows* 10 Pro, versão 21H2. Foi utilizado o *Microsoft Visual Studio Code* versão 1.69.1. Para o desenvolvimento do Banco de Dados foi utilizado *PostgreSQL* versão 10. Para o desenvolvimento do *Front-End* do sistema foi utilizado software *Mobirise* 5.6.8 além de editores de texto *Microsoft Word*, versão 2016 e os navegadores de internet *Google Chrome*, *Microsoft Edge* e *Mozilla Firefox* em suas últimas versões disponíveis durante o desenvolvimento e escrita do documento.

4.6.1 – Ajustes Iniciais do Ambiente

Para realizar as configurações iniciais do ambiente de trabalho, foi analisado o ambiente de programação disponível e dependências existentes. No *framework Django* sua única dependência é que a linguagem *Python* instalada e configurada no ambiente de desenvolvimento.

Para realizar a instalação da linguagem *Python* foi realizado acesso ao site oficial da ferramenta (Figura 23) (Python, 2022).

Figura 23 Site Oficial da Linguagem Python



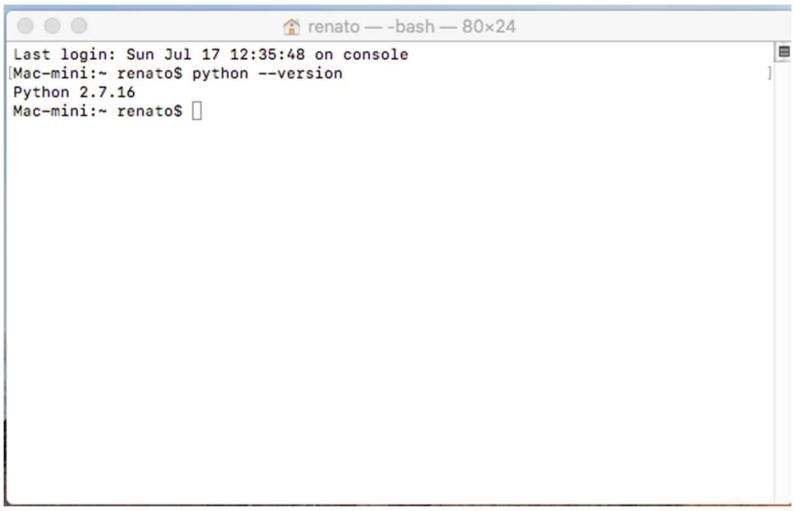
Fonte: Autor, 2022

Nos sistemas operacional com base *Unix* (*Linux*, BSB e *MacOS*) em sua grande maioria já possuem a linguagem *Python* instalada por padrão, sendo necessário apenas a verificação da versão disponível e, caso necessário, sua atualização através de comandos básicos de terminal (Figura 24).

Após o *download* e instalação da linguagem *Python* é possível realizar a configuração dos módulos do *framework Django* e sua configuração inicial. Dessa forma, para configuração do ambiente de trabalho para o projeto do SISUNAPI foi criado um diretório e configurado o ambiente virtual *venv* do *framework* e posterior desenvolvimento de todo o projeto.

Para a criação do ambiente virtual foi utilizado o seguinte comando *python3 -m venv venv*. O comando *python* é aplicativo executável responsável pela execução da linguagem; o parâmetro "-m" é responsável pela execução de módulos e bibliotecas; o parâmetro *venv* é responsável pela criação de ambientes virtuais *python*; e o segundo parâmetro *venv* é o caminho especifico onde será criado o ambiente.

Figura 24 Terminal do MacOS exibindo versão do Python disponível



Fonte: Autor, 2022

Para o projeto SISUNAPI o ambiente virtual foi criado na raiz do projeto. Após a execução do comando *python3 -m venv venv.* foi criado um diretório com a configuração padrão da linguagem *python* para a criação e execução de projetos.

Toda a alteração ou inclusão de módulos, quando o ambiente está ativo, é realizado apenas no ambiente virtual, deixando o *host*[2] isolado de qualquer alteração necessária para o projeto.

O procedimento de criação de ambientes virtual agiliza o desenvolvimento e a portabilidade do projeto para outros ambientes, tendo em vista que todos os módulos necessários a execução estarão presentes no ambiente virtual destinado para a aplicação.

Após a configuração, para a execução e ativação do ambiente virtual é necessário realizar a execução do *script* de ativação, conforme Tabela 2, onde é apresentado os comandos para os sistemas operacionais existentes.

Tabela 2 - Comandos de ativação para o Ambiente Virtual

Plataforma	Shell	Comando para ativar o ambiente virtual
POSIX	bash/zsh	$ source <venv>/bin/activate
	fish	$ source <venv>/bin/activate.fish
	csh/tcsh	$ source <venv>/bin/activate.csh
	PowerShell Core	$ <venv>/bin/Activate.ps1
Windows	cmd.exe	C:\> <venv>\Scripts\activate.bat

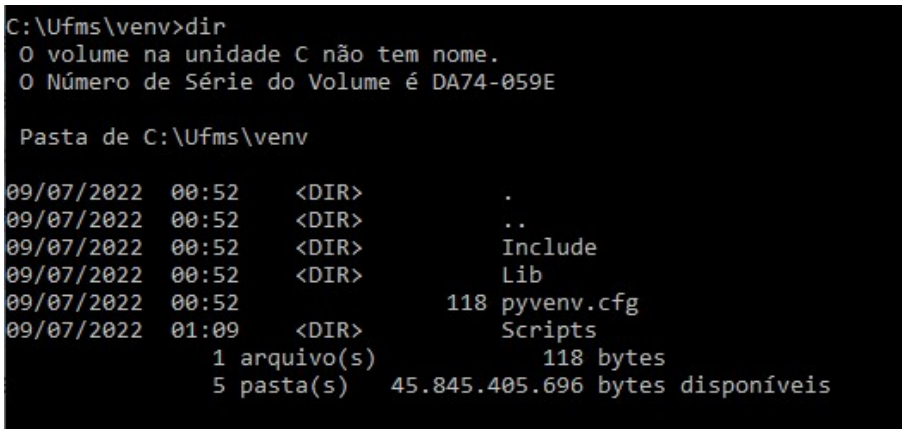
[2] *Host* é qualquer computador ou máquina conectado a uma rede, que conta com número IP e nome definidos (Viana, 2012)

Plataforma	Shell	Comando para ativar o ambiente virtual
Windows	PowerShell	PS C:\> <venv>\Scripts\Activate.ps1

Com o ambiente virtual criado e ativado é possível realizar a instalação do *framework Django*. Para realizar a instalação, basta inserir o comando *python -m pip install Django*. Para a execução do *python* é necessário que o aplicativo executável sempre seja chamado juntamente com o argumento "*-m*" para indica a execução de um módulo ou biblioteca.

O argumento *pip* indica a instalação de um pacote. O *pip* é um gerenciador de pacotes para projetos *Python*. É sempre com esse gerenciador que realizamos a instalação, atualização ou remoção de pacotes. O argumento *install* é indicado quando é necessário a instalação de um novo pacote e, o argumento *Django* indica o pacote o qual desejamos fazer a instalação. A Figura 25 apresenta uma tela com a estrutura de diretórios do ambiente virtual.

Figura 25 Estrutura dos Diretórios Criados



Fonte: Autor, 2022

Para a execução do SISUNAPI foi necessário a instalação de módulos auxiliares que foram utilizados durante o desenvolvimento do sistema (Figura 26).

Para o gerenciamento do banco de dados foi necessário a instalação do módulo *psycopg2*. O módulo é responsável pelo gerenciamento e ligação do *Python* com o servidor *PostgreSQL*. A instalação foi realizada executando o comando *python3 -m pip install psycopg2.*

Para o gerenciamento e manipulação de dados na tela de exibição e controles da interface gráfica foi realizado a instalação do módulo *pyautogui*. A instalação foi realizada executando o comando *python3 -m pip install pyautogui*.

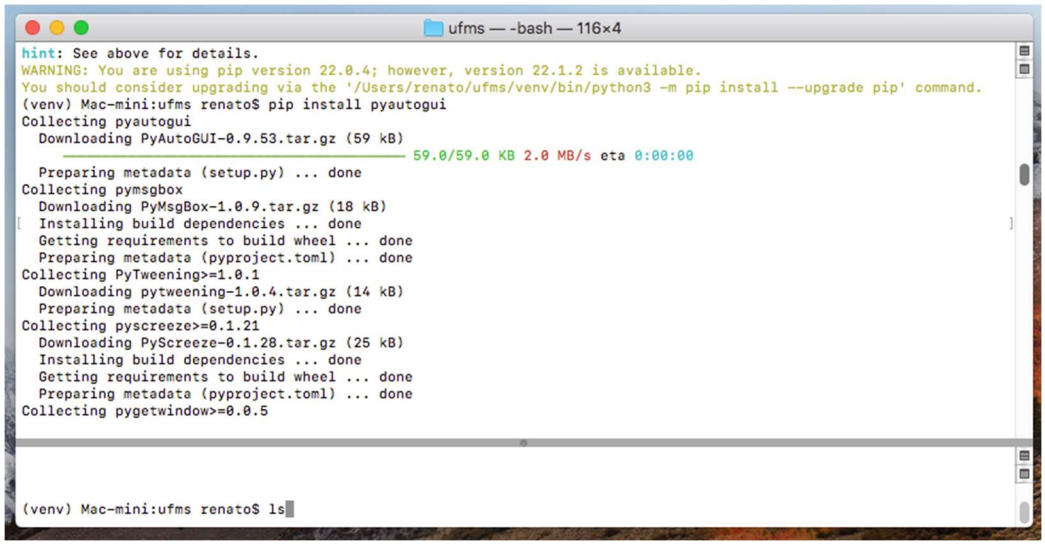
Para o gerenciamento e manipulação de imagens pelo Django é necessário a instalação do módulo *Pillow*. Para a instalação do módulo foi executado o comando *python3 -m pip install pillow*.

Para o gerenciamento de informações em arquivos de texto foi necessário a instalação do módulo *pandas*. O módulo *pandas* é essencial para a exportação de dados para arquivos de texto. Para instalação foi realizada executando o comando *python3 -m pip install pandas*.

Para o gerenciamento de dados como CPF e CNPJ foi instalado o módulo *django-cpf* e *django-cpf-cnpj*. Os módulos foram criados pela comunidade virtual para validação dos CPF e CNPJ existentes no Brasil. Para a instalação foram executados os comandos *python3 -m pip install django-cpf-cnpj django-cpf.*

Para ajuste do ambiente virtual, muitas vezes é necessário a atualização do gerenciador de pacotes *pip,* executando o comando *python3 -m pip install –upgrade pip,* que realiza a atualização do *pip*.

Figura 26 Instalação dos módulos necessários ao SISUNAPI



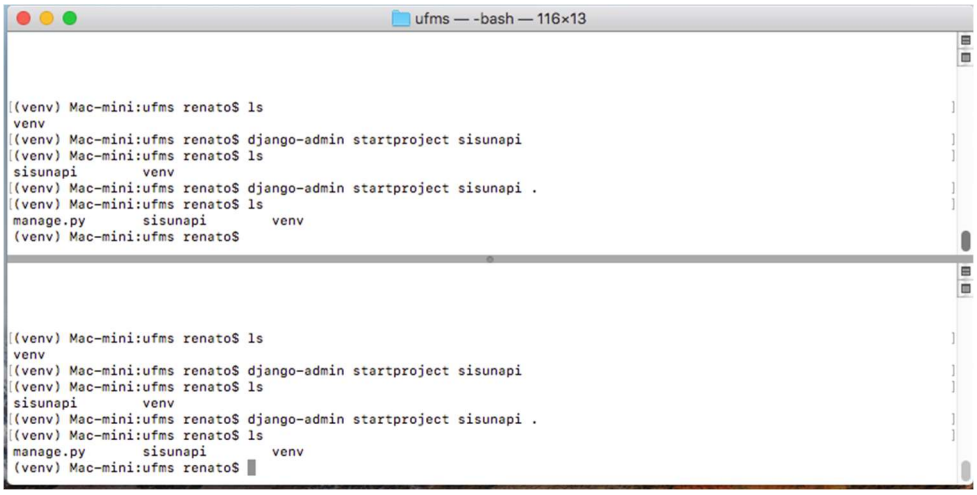
Fonte: Autor, 2022

4.6.2 – Iniciando a aplicação SISUNAPI

Após todos os ajustes finalizados é possível realizar o início do sistema SISUNAPI. Sua configuração inicial é semelhante a qualquer outro aplicativo utilizando o *framework Django* e pode ser iniciada a partir do terminal de comandos.

O comando executado para dar início ao projeto foi *django-admin startproject sisunapi .,* sendo que o argumento *django-admin* é responsável pela administração do *framework* instalado no ambiente virtual; o *startproject* é indicado para início de novos projetos; o *sisunapi* é o nome dado ao novo projeto; e o argumento "." foi indicado para que o projeto fosse criado na raiz do diretório, no qual os dados são fornecidos, conforme Figura 27.

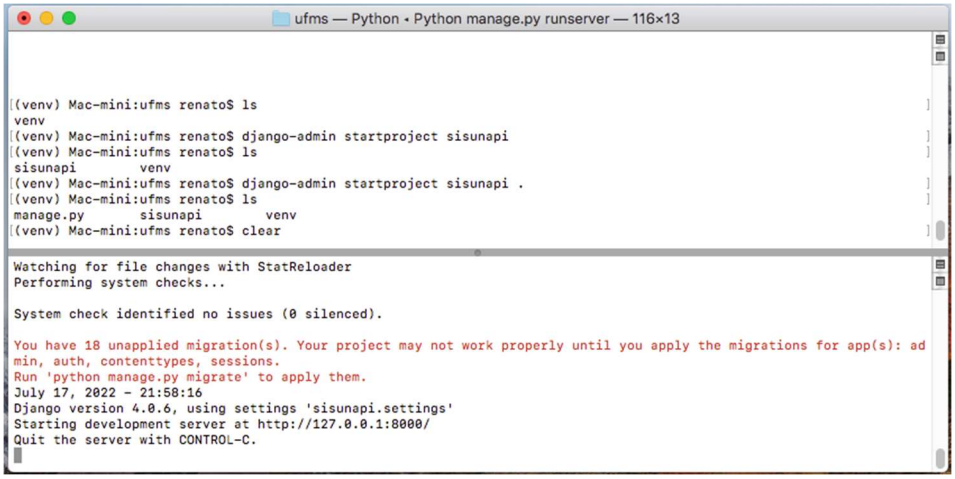
Figura 27 Inicialização do projeto SISUNAPI



Fonte: Autor, 2022

Após a execução do comando de inicialização do projeto, no *framework Django* é realizada a instalação de toda a estrutura necessária para a sua execução (Figura 28), incluindo uma versão de acesso via navegador.

Figura 28 Iniciando o servidor da Aplicação SISUNAPI



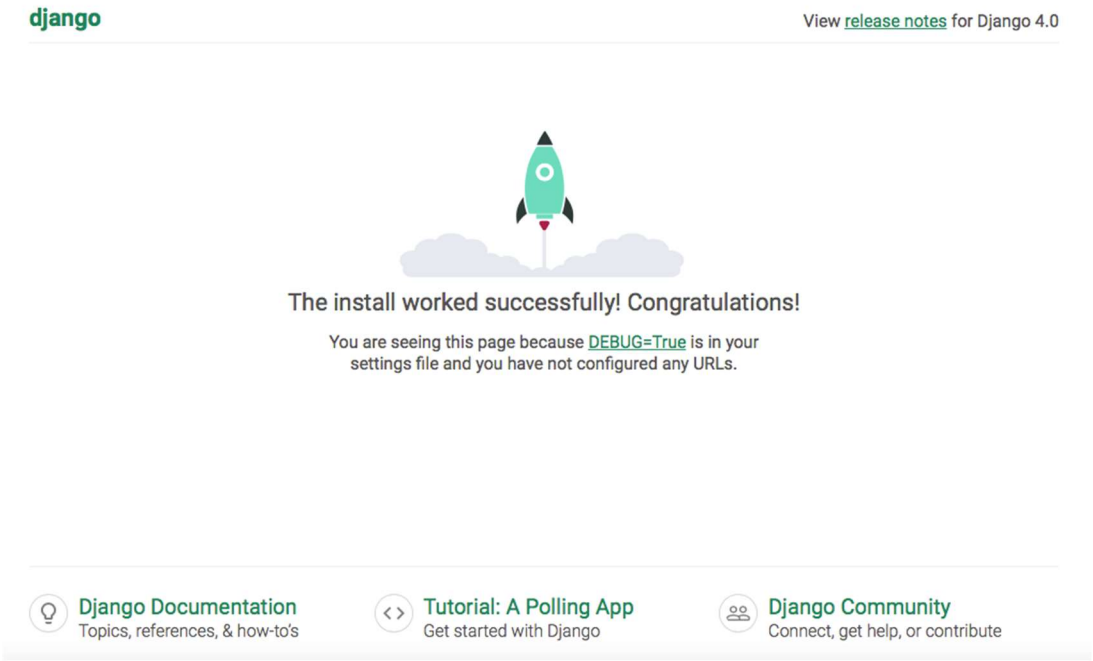
Fonte: Autor, 2022

Com a execução do comando *python3 manager.py runserver* é possível dar início ao servidor virtual do *framework* e exibir a estrutura básica do sistema em operação (Figura 29**Erro! Fonte de referência não encontrada.**).

Após as configurações iniciais e inicialização do servidor da aplicação, alguns ajustes são necessários. Com a criação do servidor, uma estrutura de diretórios e arquivos foi estabelecida de forma automática, sendo que o arquivo *settings.py* possui as configurações gerais do *framework*. As configurações de localidade e linguagem, que são ajustáveis, estão

presentes neste arquivo. Localizando a opção *LANGUAGE_CODE* e alterando para *PT-BR* e *TIME_ZONE* para *America/Campo_Grande* modificou-se a linguagem do *framework* para o português brasileiro e o fuso horário para (-4) Campo Grande.

Figura 29 Servidor da Aplicação SISUNAPI



Fonte: Autor, 2022

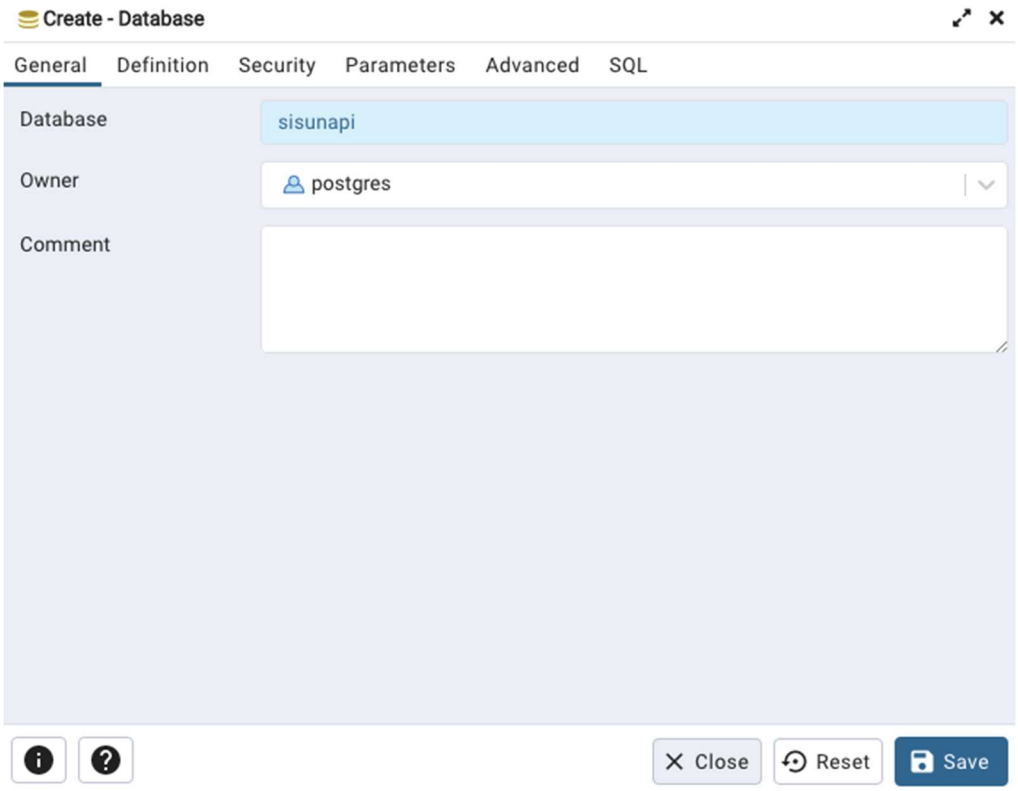
Para a melhor compreensão do usuário, durante o levantamento, identificou-se que as configurações acima seriam ideias.

4.6.3 – Configurando o Banco de Dados

Após as configurações iniciais do servidor da aplicação é necessário a configuração do servidor do banco de dados. Para essa configuração é necessário a criação de uma base de dados que será responsável pelo armazenamento de todas as informações relativas ao sistema.

Para o desenvolvimento do SISUNAPI considerou-se os bancos de dados mais utilizados pela comunidade virtual e comunidade interna da UFMS, sendo escolhido o *PostgreSQL*. A criação da base de dados com nome SISUNAPI é a base da configuração dos dados do sistema. Para a criação da base de dados pode-se utilizar o software *PgAdmin4* com a codificação UTF-8, conforme pode ser observado na Figura 30.

Figura 30 Tela de Criação da Base de Dados



Fonte: Autor, 2022

Após a criação da base de dados é necessário o ajuste no *framework Django*, mais especificamente no arquivo *settings.py*. Localizado as configurações *DATABASE* foi necessário realizar a alteração para a configurações para o *PostgreSQL* (Figura 31).

Figura 31 Alterações necessárias para acesso ao Banco de Dados

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'sisunapi',
        'USER': 'postgres',
        'PASSWORD': 'postadmin01',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

Fonte Autor, 2022

4.6.4 – Visão geral da configuração do framework

No *site* oficial do *framework Django*, ele é descrito como uma estrutura *web Python* de alto nível que incentiva o desenvolvimento rápido e o *design* limpo e pragmático (Lucas, 2022). Sua estrutura é controlada através de uma arquitetura sustentada por *Model View Template* (MVC) presente em linguagens de programação como *Java*, *C#*, *Delphi,* entre outras.

Para o desenvolvimento são escritos *Models*, caracterizados por classes *Python*, interligadas com um banco de dados relacional, controladas por um sistema de modelos na *web* (*view*) e um controle central de *urls* com expressões regulares (*controller*).

O *Model* é a representação da fonte de dados, que contém os campos e os comportamentos dos dados em armazenamento. A *View* é uma função *Python* que realiza a conexão da *web* com o *framework*. Sua resposta pode ser qualquer documento existente. Em caso de erro, a função da *View* é o retorno da mensagem. Um *Template* é caracterizado pelas informações fixas do sistema. Geralmente é representado pela saída HTML apresentada no navegador (Lucas, 2022).

4.6.5 - *Apps* criados para o SISUNAPI

Todo o sistema criado foi baseado em *Apps*. Os *Apps* são módulos do *framework* que correspondem a uma pequena parte de um todo. São nos *apps* que são definidas as unidades básicas que compõe o SISUNAPI.

Durante a modelagem do sistema foram observadas, no primeiro momento, as necessidades primárias essenciais a serem mapeadas. Para o projeto inicial do SISUNAPI foram criados os seguintes *Apps*: Agenda, Arquivos, Cadastros, Clínica, Curso, Esporte e Pessoa. Cada um dos *Apps* é responsável por uma sequência de recursos existentes dentro da ferramenta, que estão interligados entre si e são codependentes.

a) App Agenda

O *App Agenda* é responsável pelo cadastro geral de todas as atividades existentes pela UnAPI e pela exibição pública das informações existentes na página principal de todo o sistema.

b) App Arquivo

O *App Arquivo* é responsável pelo arquivamento dos arquivos dos usuários. É o principal *app* do sistema, pois é responsável pelo gerenciamento dos arquivos de pesquisa, dos registros de participação dos usuários, entre outros.

c) App Clinica

O *App Clinica* é responsável pelo registro dos principais atendimentos clínicos oferecidos aos usuários da UnAPI. Os atendimentos devem estar registrados para se manter um histórico de todas as atividades realizadas pelos usuários na UnAPI.

d) App Curso

O A*pp Curso* é responsável pelos registros dos cursos oferecidos pela UnAPI aos seus participantes.

e) App Esporte

O *App Esporte* é responsável pelos registros das atividades esportivas pela UnAPI aos seus usuários participantes.

f) App Pessoa

O *App Pessoa* é responsável pelo cadastro das pessoas que oferecem atividades através da UnAPI. Uma pessoa cadastrada pode ser um professor, um dentista, um psicólogo, um fisioterapeuta, entre outros.

g) App Cadastro

O *App Cadastro* é responsável pela ligação dos usuários participantes da UNAPI registrados no sistema, com as suas atividades participantes.

O foco inicial do SISUNAPI é o registro das atividades realizadas pelos usuários para garantir que tudo fique registrado, permitindo estudos futuros com esses dados.

4.6.6 – *Models* de Cada *App* Criado

Um *Model* é a camada de ligação do *framework Django* com o gerenciador de Banco de Dados (Figura 32) e a escolha do SGBD para o sistema foi o *PostgreSQL*.

4.6.6.1 Configuração do Model para o App Agenda

O *Model* criado para o *App* Agenda ocorreu da importância desse *App* para o sistema.

Como a página principal do sistema está sendo direcionada para os eventos registrados nesse *App* é vital que a Tabela no Banco de Dados possua todas as informações necessárias para auxiliar os usuários durante a sua utilização.

Figura 32 Classe Agenda do Model do APP Agenda.

```
class Agenda(models.Model):
    nome_atividade = models.CharField(max_length=200)
    descricao_atividade = models.TextField()
    responsavel = models.CharField(max_length=150)
    data_inicio = models.DateField()
    data_fim = models.DateField()
    numero_vagas = models.IntegerField()
    edital = models.FileField(upload_to='media/agenda/%Y/%m/%d/', blank=True)
    link = models.URLField(max_length=200, null=True, blank=True)
    data_publicacao = models.DateTimeField(default=datetime.now, blank=True, null = True)
    publicada = models.BooleanField(default=False)
```

Fonte: Autor, 2022

Observando a Figura 32 identifica-se que o *nome_atividade* e *descricao_atividade* são campos que possuem importância, pois são inseridos neles os detalhes da atividade que está sendo exibida na página principal do sistema. Há também um campo publicada que indica se essa inserção no banco está ativa para exibição pelo sistema. Nessa tabela também há a indicação para um *edital* e um *link*, que pode ser externo ao sistema.

4.6.6.2 Configuração do Model para o App Arquivo

O *Model* criado para o *App* Arquivo (Figura 33) possui detalhes para que os arquivos que sejam armazenados dentro da ferramenta, possam ser classificados de maneira ordenada. Para esse *Model* foram criadas 3 classes diferentes, cada uma com sua especialização e todas interligadas.

A classe *TipoArquivo* é responsável pelo cadastro de todos os tipos de arquivos possíveis de serem incluídos no sistema. A classe *Classificacao* é responsável pelo cadastro de todos os possíveis níveis de classificação de arquivos dentro do sistema. Já a classe Arquivo é a classe principal do *Model*, sendo responsável diretamente pelo registro do armazenamento dos arquivos dentro do sistema.

Figura 33 Model do App Arquivo

```
class TipoArquivo(models.Model):
    nome = models.CharField(max_length=150)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=False)

    def __str__(self):
        return self.nome

class Classificacao(models.Model):
    nome = models.CharField(max_length=150)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=False)

    def __str__(self):
        return self.nome

class Arquivo(models.Model):
    nome = models.CharField(max_length=255)
    arquivo = models.FileField(upload_to='media/arquivo/%Y/%m/%d/', blank=False)
    descricao = models.TextField(max_length=250, null=True, blank=True)
    classificacao = models.ForeignKey(Classificacao, on_delete=models.CASCADE, verbose_name="Classificação do Arquivo")
    tipo_arquivo = models.ForeignKey(TipoArquivo, on_delete=models.CASCADE, verbose_name="Tipo de Arquivo")
    usuario = models.ForeignKey(User, on_delete=models.PROTECT, verbose_name = "Usuário")
    responsavel = models.ForeignKey(Pessoa, on_delete=models.CASCADE, verbose_name="Responsável")
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    publicada = models.BooleanField(default=False)
```

Fonte: Autor, 2022

Por definição, realizada durante o levantamento de requisitos, um arquivo pode ser um projeto de pesquisa, um trabalho acadêmico, uma prova, um exame físico, uma ficha odontológica, uma dissertação, um artigo científico, entre outros. Todo o documento que gera informação sobre os participantes do sistema é um arquivo habilitado para ser armazenado dentro do sistema SISUNAPI.

4.6.6.3 Configuração do Model para o App Clinica

O *Model* criado para o *App Clinica* é composto por duas classes, sendo a primeira *Tipo*, onde serão armazenados todos os tipos de atendimento clínicos possíveis de serem realizados pela UnAPI e a segunda a *Clinica*, onde serão armazenados os atendimentos clínicos disponíveis na UnAPI. Em definição, todo o atendimento clínico necessita de um responsável direto, que é uma pessoa registrada no *App Pessoa*. Há um campo existente na classe *Clinica* onde indica se essa está ou não disponível para publicação, sendo exibida no sistema (Figura 34).

4.6.6.4 Configuração do Model para o App Curso

O *Model* criado para o *App Curso* está composto por duas classes (Figura 35). A primeira é a classe *Setor*, onde são realizados os cadastramentos de todos os setores existentes, que disponibilizam cursos para os participantes do SISUNAPI. A segunda classe é a *Curso*, responsável pelo cadastro de cursos disponibilizados pela UnAPI. A classe possui campos como

o responsável, que é uma pessoa registrada no *App* Pessoa e uma indicação se o curso está ativo ou não, tendo a sua exibição permitida no ambiente do sistema.

Figura 34 Model do App Clinica

```
class Tipo(models.Model):
    nome = models.CharField(max_length=150)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=False)

    def __str__(self):
        return self.nome

class Clinica(models.Model):
    nome = models.CharField(max_length=255)
    descricao = models.TextField()
    tipo = models.ForeignKey(Tipo, on_delete=models.CASCADE)
    responsavel = models.ForeignKey(Pessoa, on_delete=models.CASCADE)
    setor = models.ForeignKey(Setor, on_delete=models.CASCADE)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=False)
    publicada = models.BooleanField(default=False)

    def __str__(self):
        return "{} ({})".format(self.nome, self.descricao)
```

Fonte: Autor, 2022

Figura 35 Model do App Cursos

```
class Setor(models.Model):
    nome = models.CharField(max_length=255)
    localidade = models.CharField(max_length=250)
    ativo = models.BooleanField(default=True)

    def __str__(self):
        return "{} ({})".format(self.nome, self.localidade)

class Curso(models.Model):
    nome = models.CharField(max_length=255)
    descricao = models.TextField()
    responsavel = models.ForeignKey(Pessoa, on_delete=models.CASCADE)
    setor = models.ForeignKey(Setor, on_delete=models.CASCADE)
    modo_de_realizacao = models.CharField(max_length=250)
    periodicidade = models.CharField(max_length=250, null = True, blank=True )
    dias = models.CharField(max_length=250, null = True, blank=True )
    horario = models.CharField(max_length=250, null = True, blank=True )
    numero_vagas = models.IntegerField()
    edital = models.FileField(upload_to='media/curso/%Y/%m/%d/', blank=True)
    link = models.URLField(max_length=200, null=True, blank=True)
    curso_ativo = models.BooleanField(default=False)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)

    def __str__(self):
        return "{} ({})".format(self.nome, self.descricao)
```

Fonte: Autor, 2022

O *Model Esportes* é composto por duas classes (Figura 36). A classe Modalidade onde é efetuado o cadastro de todas as modalidades esportivas as quais a UnAPI oferece aos seus participantes. Já a classe Esporte, onde é efetuado o registro das atividades esportivas

disponíveis para serem oferecidas. Todo a atividade esportiva ofertada na UnAPI possui um responsável, previamente cadastrado no *App Pessoa.*

Figura 36 Model do App Esportes

```
class Modalidade(models.Model):
    nome = models.CharField(max_length=150)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=False)

    def __str__(self):
        return self.nome

class Esporte(models.Model):
    nome = models.CharField(max_length=255)
    descricao = models.TextField()
    modalidade = models.ForeignKey(Modalidade, on_delete=models.CASCADE)
    responsavel = models.ForeignKey(Pessoa, on_delete=models.CASCADE)
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
    ativo = models.BooleanField(default=False)
    publicada = models.BooleanField(default=False)

    def __str__(self):
        return "{} ({})".format(self.nome, self.descricao)
```

Fonte: Autor, 2022

4.6.6.5 Configuração do Model para o App Pessoa

O *Model Pessoa* é responsável pelo cadastro de todas as pessoas que possuem funções dentro da UnAPI (Figura 37). As funções são descritas na classe *TipoPessoa*, onde é definido se ela é professora, coordenadora, pesquisadora, auxiliar, entre outros.

A pessoa cadastrada no *App Pessoa* é diferente de um usuário registrado no sistema. O usuário vai possuir acesso, entretanto, uma pessoa cadastrada apenas no *App Pessoa* não terá acesso ao sistema, mas será referenciada em atividades ou arquivos registrados no sistema.

Figura 37 Model do App Pessoa

```
class TipoPessoa(models.Model):
    nome = models.CharField(max_length=200)
    ativo = models.BooleanField(default=True)

    def __str__(self):
        return self.nome

class Pessoa(models.Model):
    nome = models.CharField(max_length=200)
    email = models.CharField(max_length=200)
    data_nasc = models.DateField()
    tipo_pessoa = models.ForeignKey(TipoPessoa, on_delete=models.CASCADE)
    cpf = CPFField(masked=True)
    rg = models.CharField(max_length=50, null=True, blank=True)
    cep = models.CharField(max_length=10, null=True, blank=True)
    num_cep = models.IntegerField(null=True, blank=True)
    foto = models.ImageField(upload_to='pessoas/upload/%Y/%m/%d', null=True, blank=True)
    data_cadastro = models.DateTimeField(default=datetime.now, blank=True, null = True)

    def __str__(self):
        return self.nome
```

Fonte: Autor, 2022

Para o *Model* foi definido duas classes. A *TipoPessoa* onde são registrados todos os tipos de pessoa que possuem atividade dentro da UnAPI, seja ela professor, assistente, médico, fisioterapeuta, ou outro. E a classe *Pessoa*, onde são armazenados todos os dados de cadastro desses profissionais que atuam na UnAPI.

4.6.6.6 Configuração do Model para o App Cadastro

Para o *App Cadastro* (Figura 38) foi necessário criar diversos relacionamentos para a construção de seu *Model*. Como é nesse *App* que os usuários farão os seus registros e vínculos, foi necessário realizar o referenciamento, através de chaves estrangeiras nos outros *Apps* e seu *Models* para as corretas relações, entre cada item.

Figura 38 Model do App Cadastro

```
class ArquivosUsuario(models.Model):
    data_cadastro = models.DateTimeField(auto_now_add=True, verbose_name = 'Data do Cadastro')
    ativo = models.BooleanField(default=True)
    arquivo_usuario = models.FileField(upload_to='media/arquivo/usuario/%Y/%m/%d/', blank=False, verbose_name='Arquivo do Usuário')
    arquivo = models.ForeignKey(Arquivo, on_delete=models.PROTECT)
    usuario = models.ForeignKey(User, on_delete = models.PROTECT)
    class Meta:
        unique_together =['arquivo','usuario']

    def __str__(self):
        return "{}".format(self.arquivo_usuario)

class AgendaUsuario(models.Model):
    data_cadastro = models.DateTimeField(auto_now_add=True)
    ativo = models.BooleanField(default=True)
    agenda = models.ForeignKey(Agenda, on_delete=models.CASCADE, verbose_name='Item da Agenda')
    usuario = models.ForeignKey(User, on_delete=models.PROTECT)
    class Meta:
        unique_together =['agenda','usuario']

    def __str__(self):
        return "{}".format(self.agenda)

class ClinicaUsuario(models.Model):
    data_cadastro = models.DateTimeField(verbose_name = 'Data do Cadastro', blank=True)
    ativo = models.BooleanField(default=True)
    clinica = models.ForeignKey(Clinica, on_delete=models.CASCADE, verbose_name = 'Clinica para Atendimento')
    usuario = models.ForeignKey(User, on_delete=models.PROTECT)
    class Meta:
        unique_together =['clinica','usuario']

    def __str__(self):
        return "{}".format(self.clinica)

class CursosUsuario(models.Model):
    data_cadastro = models.DateTimeField(auto_now_add=True, verbose_name = 'Data do Cadastro')
    ativo = models.BooleanField(default=True)
    curso = models.ForeignKey(Curso, on_delete=models.CASCADE, verbose_name='Nome do Curso')
    usuario = models.ForeignKey(User, on_delete=models.PROTECT)
    class Meta:
        unique_together =['curso','usuario']

    def __str__(self):
        return "{}".format(self.curso)

class EsportesUsuario(models.Model):
    data_cadastro = models.DateTimeField(auto_now_add=True, verbose_name = 'Data do Cadastro')
    ativo = models.BooleanField(default=True)
    esporte = models.ForeignKey(Esporte, on_delete=models.CASCADE, verbose_name='Nome do Esporte')
    usuario = models.ForeignKey(User, on_delete=models.PROTECT)
    class Meta:
        unique_together =['esporte','usuario']

    def __str__(self):
        return "{}".format(self.esporte)
```

Fonte: Autor, 2022

Durante a definição de como o sistema SISUNAPI funcionaria em sua primeira versão, ficou definido que um usuário administrador fará a inclusão dos dados disponíveis para acesso, como uma Atividade, cadastrada no *App Agenda*, uma Oficina de Dança, que será disponibilizada no *App Curso* ou uma atividade esportiva, que será cadastrada no *App Esporte*, por exemplo.

Após o cadastro realizado pelo usuário administrador de todas as possíveis atividades que um usuário devidamente cadastrado no sistema junto a UnAPI possa realizar, terá a opção de vincular o usuário a uma determina atividade. É o *Model Cadastro* que será responsável por todos os registros e vínculos dos usuários e suas atividades dentro do sistema.

Cada *Model* representa uma estrutura dentro do Banco de Dados. Uma das características do *Framework Django* é sua facilidade de administração e manutenção. Quando necessário quaisquer alterações dentro da estrutura do banco de dados, apenas com uma modificação dentro do *Model* escolhido, o *framework* automatiza todas as alterações necessárias a nível de banco de dados, incluindo relacionamentos e cardinalidades necessárias para manter o bom funcionamento do mesmo.

Após os primeiros modelos desenhados no sistema, foi necessário a inclusão de mais uma *Classe* que realiza o cadastro das informações auxiliares dos usuários registrados, para terem acesso ao sistema. Nessa *Classe* foi incluído informações adicionais, tais como nome completo do usuário, CPF, RG, número de telefone para contato e sua data de nascimento.

Figura 39 Model do App Usuários - Classe Perfil

```
class Perfil(models.Model):
    nome_completo = models.CharField(max_length=250, null=True, verbose_name='Nome Completo')
    cpf = CPFField('cpf', null=True)
    rg = models.CharField(max_length=50, null = True, blank=True, verbose_name= 'RG')
    telefone = models.CharField(max_length=16, null=True, verbose_name= 'Telefone')
    data_nasc = models.DateField(null=True, verbose_name='Data de Nascimento')
    usuario = models.OneToOneField(User, on_delete=models.CASCADE)
```

Fonte: Autor, 2022

Essa *Classe* possui vínculo fortíssimo e necessário com a tabela de usuários do sistema do *Framework Django*. Após o usuário realizar o seu cadastro, esse necessariamente ganha um vínculo com essa tabela, sendo necessário o cadastro de seus dados para que seu acesso seja considerado válido e possa utilizar o sistema de forma total. Todo o usuário deve possuir, portanto, um perfil que é vinculado apenas a um usuário.

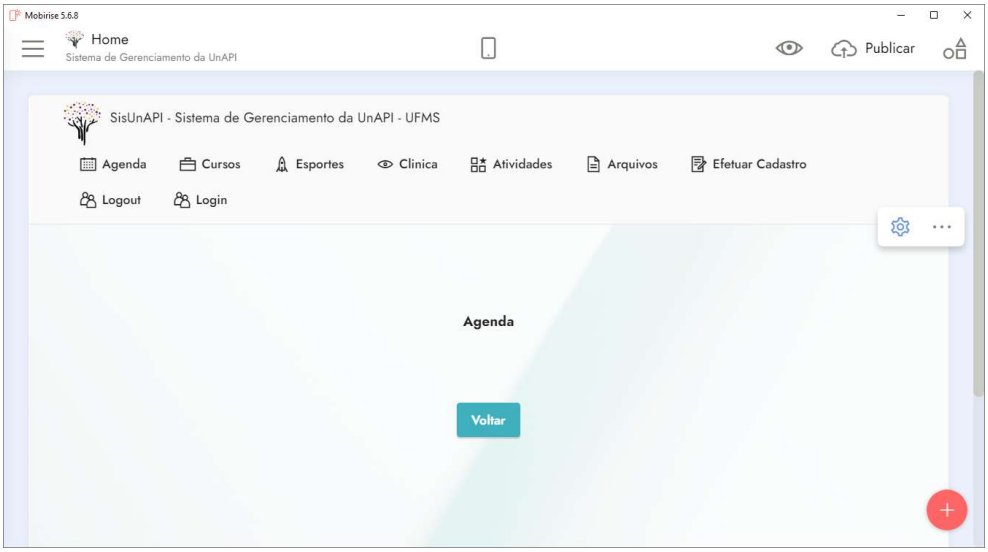
4.6.7 – *Layout* desenvolvido para o SISUNAPI

Para a primeira versão do sistema SISUNAPI foi desenhado um *layout* minimalista, que apresentasse as principais funções necessárias para a compreensão dos usuários (Figura 40). Tendo em vista que os usuários poderão ser pessoas com certa dificuldade em tecnologia, pensou-se na facilidade de uso.

Já para o usuário administrador, que poderá ser um ou mais pessoas, será disponibilizado a área administrativa do próprio *framework Django,* que atenderá de forma clara e objetiva o princípio inicial do sistema.

O *layout* inicial do sistema foi projeto utilizando-se a ferramenta *Mobirise 5.6.8*. Todos os *plugins* necessários para o funcionamento do *layout* também foram contemplados juntamente com a ferramenta.

Figura 40 Tela de Acesso da Ferramenta de criação de Layouts

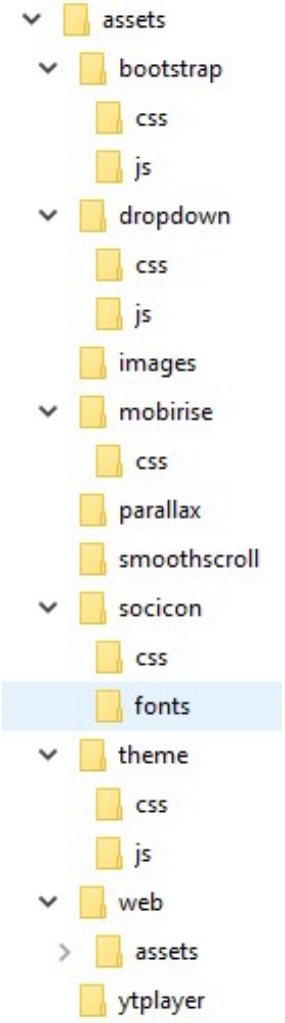


Fonte: Autor, 2022

Todo o conteúdo gerado pela ferramenta foi vinculado ao *framework Django* de forma a atender todo o sistema de forma igualitária. A ferramenta *Mobirise*, após o desenho inicial do *layout*, gerou quatro arquivos *HTML* e uma pasta com arquivos auxiliares e *scripts* relativos ao *layout* desenhado (Figura 41). O *layout* escolhido utiliza o *bootstrap*.

O *bootstrap* é um *framework front-end* que fornece estrutura *CSS* (Folha de Estilo em Cascata) para a criação de *sites* e aplicações responsivas de forma rápida e simples, além de permitir que sites *desktop* e páginas de dispositivos móveis sejam projetados do mesmo *layout* (Lima, 2021).

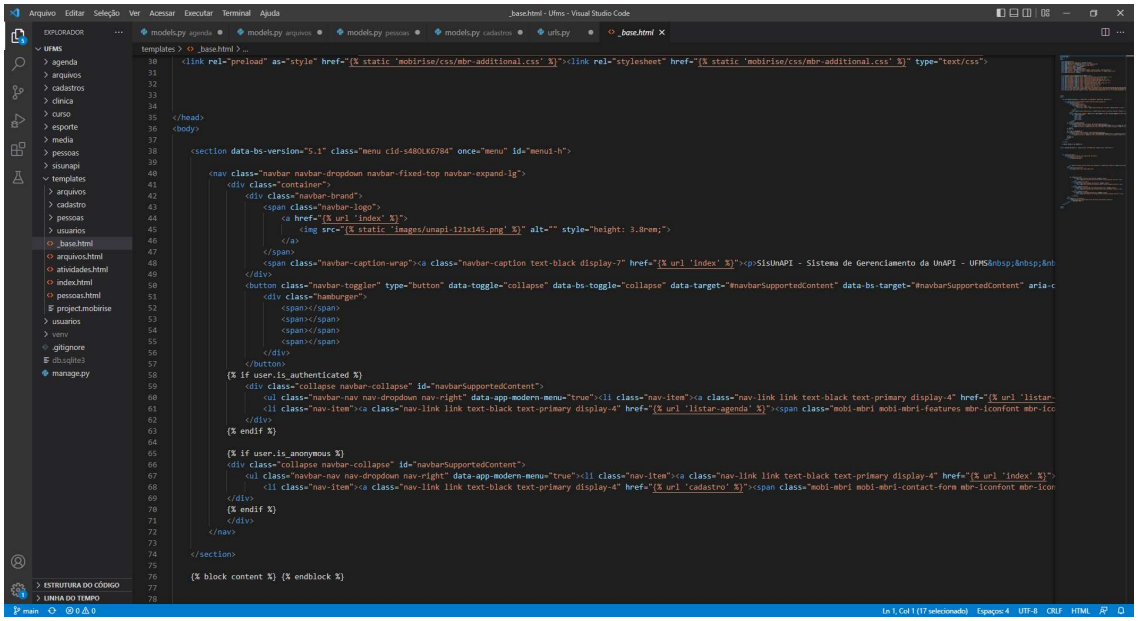
Figura 41 Estrutura de Diretórios Criado pelo *Mobirise*



Fonte: Autor, 2022

O *framework Django* possui um gerenciamento nativo de arquivos estáticos (Figura 42). Após a criação dos arquivos pela ferramenta *Mobirise* foi necessário um ajuste no arquivo *index.html*, transformando-o em arquivo base para todos os outros do sistema SISUNAPI.

Figura 42 Janela com a Codificação do arquivo _base.html derivado do index.html



Fonte: Autor, 2022

Para a utilização da estrutura no ambiente do *framework Django* foram incluídos os arquivos e diretórios na estrutura de *templates* do *framework* e se indica nas configurações a utilização desses arquivos.

Em linhas gerais, para a indicação de um arquivo estático no *framework* é necessária a sinalização através de um comando no início do arquivo, que faz parte da codificação de arquivos estáticos. A sentença {%load static%} indica que ali há um carregamento de arquivos estáticos pertencentes ao *framework* e a aplicação. Também se faz necessário o ajuste de todos os arquivos, que outrora estavam indicando locais fixos, para serem indicados como pertencentes aos arquivos estáticos da aplicação. A sentença {%static%} indica que ali faz-se uso de arquivos estáticos do *framework*.

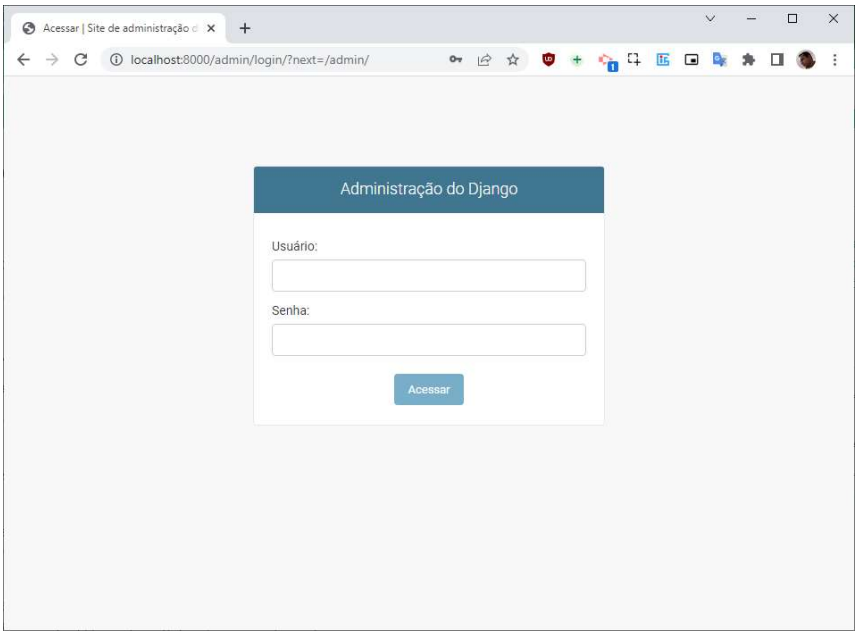
Segundo Magalhães (2018) diferente do que ocorre em outras tecnologias, no *framework Django* a ideia é explicitamente delegar a entrega de conteúdo estático para outras aplicações. A utilização de um segundo *framework*, o *bootstrap*, para o gerenciamento e controle do *frontend* fez-se necessário devido a essas características presente no *framework Django*.

4.6.8 – Interface de Administração

O *framework Django* fornece uma *interface* de administração completa na qual é possível realizar todas as manipulações de informações que serão apresentadas para os

usuários do sistema. Essa *interface* será fornecida aos administradores do sistema (Figura 43). O número de pessoas responsáveis pela inserção dos dados pode ser infinito, porém a administração geral do sistema caberá a um número mínimo de pessoas, sendo necessário a informação destes no momento da sua implantação.

Figura 43 Interface de Gerenciamento do SISUNAPI



Fonte: Autor, 2022

4.6.9 – Segurança da Aplicação

Uma das principais preocupações de aplicações *web* é com relação aos requisitos de segurança. Um dos motivos da escolha do *framework Django* foi relacionado aos requisitos de segurança, pois foi projetado para evitar os ataques mais comuns. O *Cross Site Request Forgery* (CSRF) previne que um usuário execute ações usando credenciais de um terceiro. O *framework Django* gera uma chave específica para cada transação de usuário/navegador. Durante a inserção de dados é realizado a validação dessa chave, sendo considerada inválida, o acesso será negado e não será permitido o acesso ao sistema.

Além da segurança CSRF, há a implementação da proteção contra injeção de SQL (SQL *injection*), que evita que usuários maliciosos executem *scripts* SQL para inserções no banco de dados de forma indevida.

O *framework Django* possui proteção contra *clickjacking* em *X-Frame-Options middleware*, evita que o navegador, onde o usuário esteja "logado", renderize um *frame* de outro *site* ou ambiente.

O *framework Django* também possui habilitação para SSL/HTTPS. O *Secure Sockets Layer* (SSL) é um protocolo de segurança digital que permite a comunicação criptografada entre um domínio e um navegador. O *Hyper Text Transfer Protocol Secure* (HTTPS) é o protocolo de camada de aplicação que realiza a comunicação entre clientes e servidores de forma protegida (Delavy, 2022).

A escolha pelo SGBD *PostgreSQL* está ligada a seus critérios de segurança. Seus critérios de autenticação, sistema de controle de acesso robusto, segurança em nível de coluna e linha e a autenticação multifator com certificados, fazem do *PostgreSQL* uma escolha segura para a guarda dos dados.

O SISUNAPI será contemplado em *Docker*. O *Docker* é uma plataforma *open-source* que facilita a criação e administração de ambientes isolados, empacotando todo o ambiente, tornando portátil para qualquer outro *host* que contenha o *Docker* instalado.

A escolha pelo empacotamento em *Docker* garante mais um requisito de segurança pois, além da segurança existente no *framework Django* e no SGBD *PostgreSQL,* garantirá um ambiente com o mínimo de modificações, independente do sistema ao qual for hospedado futuramente.

Capítulo 5

RESULTADOS

Nesse capítulo são apresentados os resultados das implementações e os testes realizados em cada funcionalidade. Foram realizados testes na ferramenta com o perfil administrador, com a inclusão de dados fictícios, para a validação. Com os dados armazenados, foram realizados testes com o perfil de usuário comum (idoso) onde foram testados os recursos e validados.

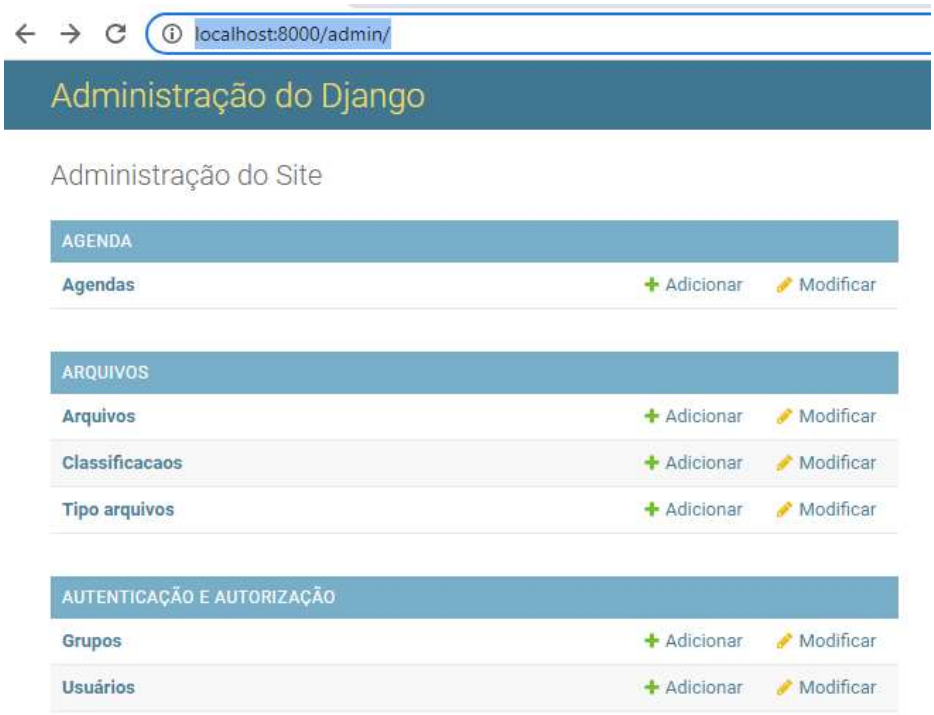
5.1 Utilização da ferramenta como usuário administrador

Durante a formulação da ferramenta, foi pensando que haveria a necessidade de um usuário administrador do sistema. Esse usuário seria responsável pela criação de todas as personalizações possíveis dentro da ferramenta.

Após a integração inicial da ferramenta e a ligação ao servidor, o *framework Django* fornece uma plataforma de administração completa. Tal plataforma é segura e objetiva, tendo como principal finalidade a personalização geral de todo o sistema. Com esse perfil é possível realizar as integrações e ajustes necessários em todo sistema.

Ao realizar o acesso no painel de administração do *framework Django*, por consequência, do painel do SISUNAPI, é possível verificar a classificação de todos os itens *Apps* criados (Figura 44).

Figura 44 Painel de Acesso Administrativo.



Fonte: Autor, 2022

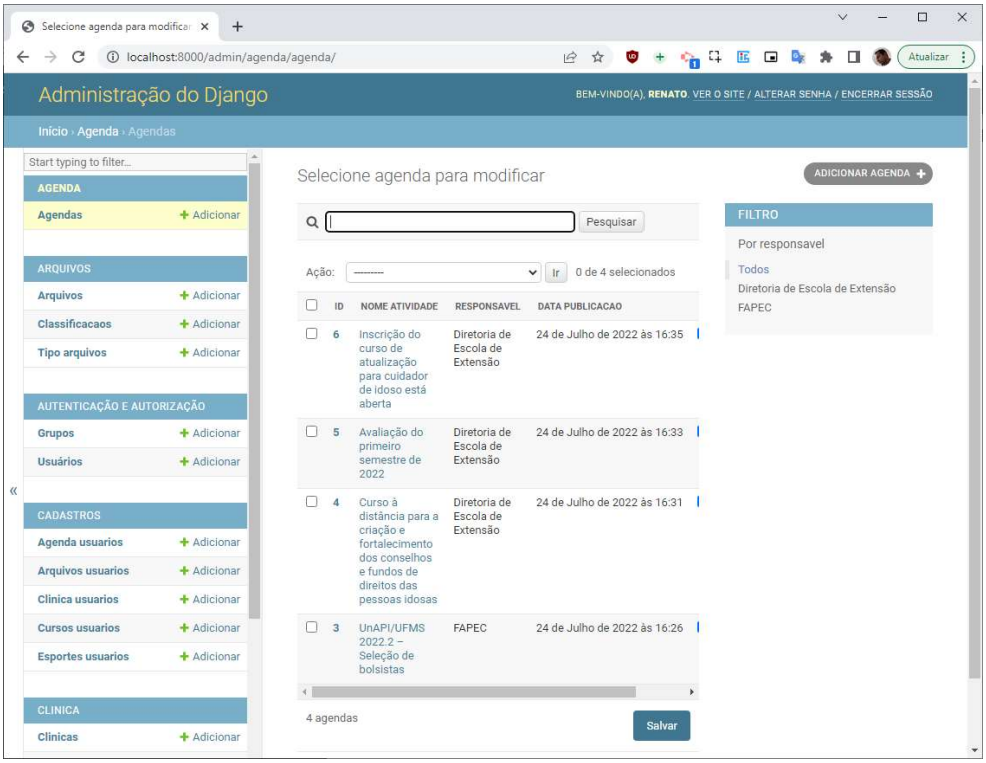
Cada *App* criado possui o seu acesso no painel administrativo. Esse acesso é realizado durante a configuração do *framework Django*.

Para a correta utilização do sistema é necessário realizar o cadastramento de todos os tipos de dados no sistema. Em cada *App* há a sua peculiaridade com relação aos tipos de dados existentes e previstos.

Para as configurações iniciais do Sistema, foi realizado o teste de cadastro das atividades disponíveis dentro da UnAPI. Foi realizado o levantamento no *site* da UNAPI (http://www.unapi.ufms.br) e realizada a coleta de alguns itens em destaque. Durante o levantamento de requisitos, foi idealizado que o *App Agenda* será o responsável pela apresentação desses itens aos usuários do SISUNAPI e esses, por sua vez, devem optar por estar associado a cada item de interesse, para posterior consulta.

Para o cadastro dos itens, foi realizado o acesso ao painel administrativo e selecionado o item *Agenda* / *Agendas*. Nessa tela é apresentado um botão "ADICIONAR AGENDA". E, ao realizar a sua seleção, é apresentado uma nova janela onde é possível fazer a inclusão dos detalhes do Evento disponibilizado pela UnAPI aos seus usuários (Figura 45).

Figura 45 Cadastro de Agenda, perfil Administrativo



Fonte: Autor, 2022

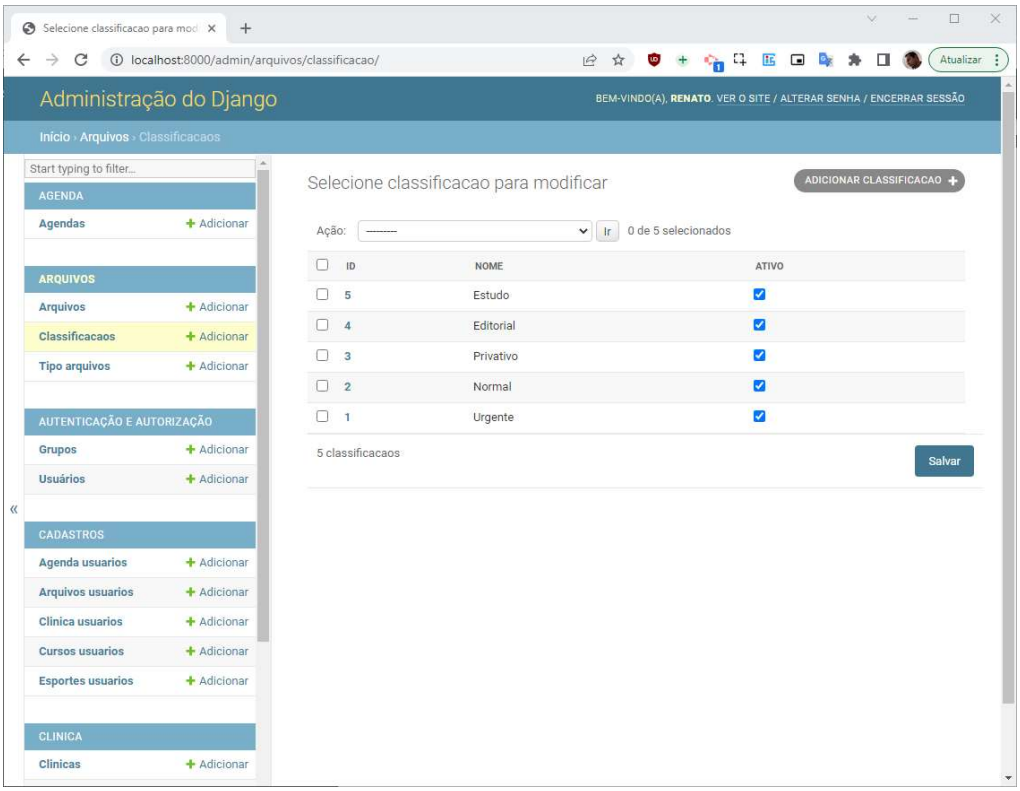
Após a inserção de alguns itens na Agenda é possível verificar, na página principal do sistema, a exibição desses novos itens (Figura 46). A exibição ficou definida para ser pública, não sendo necessário o *login* prévio para a sua exibição.

Figura 46 Tela inicial do SISUNAPI

Fonte: Autor, 2022

Para a correta funcionalidade da inserção de arquivos no *App Arquivos* é necessário realizar o cadastro das classificações que cada arquivo pode possuir. No *App Arquivos* / *Cadastro* é possível realizar a inserção dessas classificações. Para os testes foram cadastrados cinco itens: Urgente, Normal, Privativo, Editorial e Estudo, conforme Figura 47.

Figura 47 Cadastro de Classificação de Arquivos

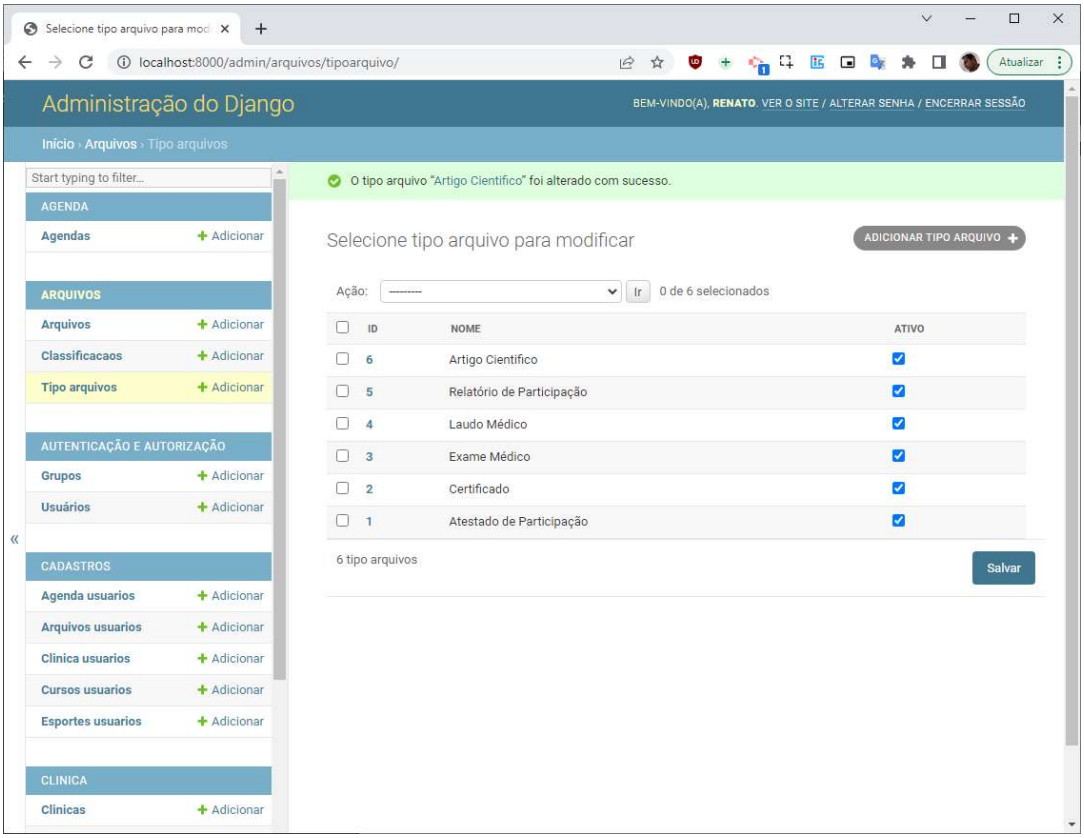


Fonte: Autor, 2022

Ainda no *App Arquivos* é necessário realizar o cadastro dos possíveis tipos de arquivos que podem ser armazenados no sistema. Um tipo de arquivo é qualquer documento digital, que possua relação com os participantes dos projetos vinculados a UnAPI. Um documento pode ser um artigo científico, um relatório de participação, um atestado ou laudo médico, um certificado de participação de evento, uma foto de algum evento de participação coletiva dos usuários, entre outros.

Para a correta inserção e posterior classificação, é necessário realizar o registro do tipo de arquivo antes de qualquer inserção. Para os testes foram cadastrados seis tipos de arquivos: Atestado de participações, Certificado, Exame Médico, Laudo Médico, Relatório de Participação e Artigo Cientifico (Figura 48).

Figura 48 Cadastro de Tipos de Arquivos



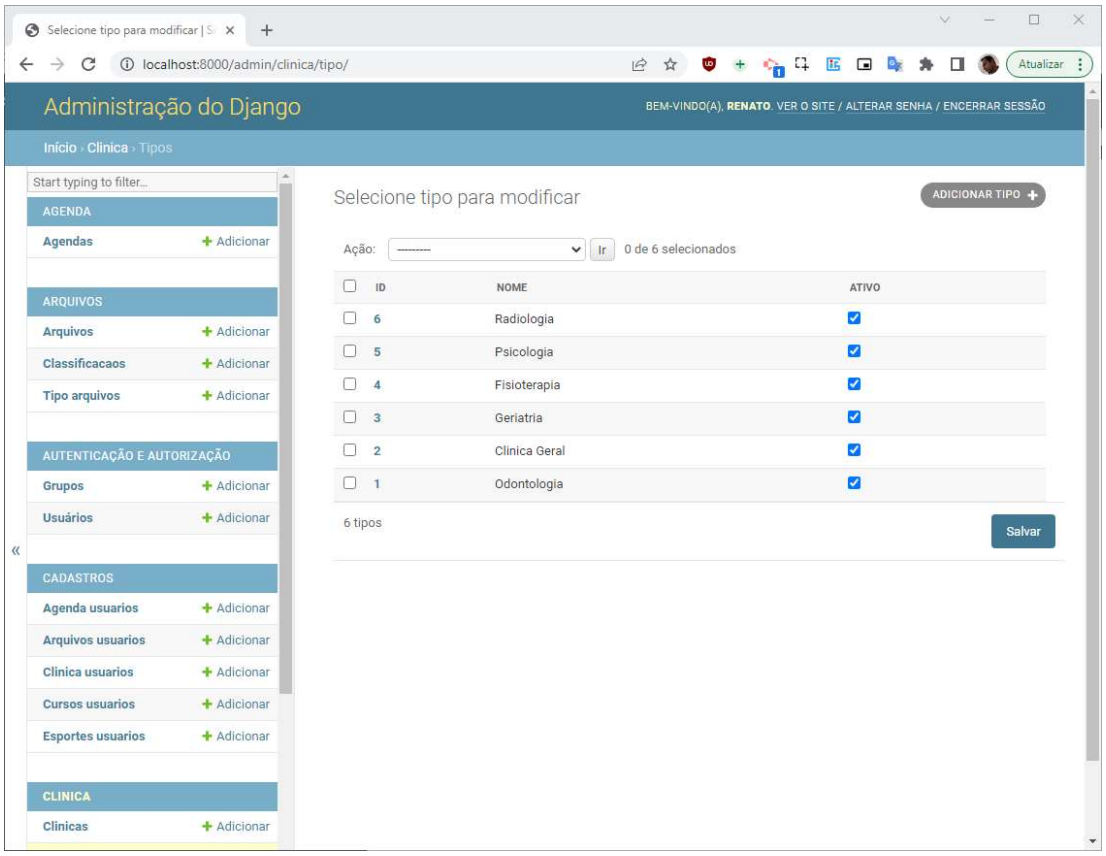
Fonte: Autor, 2022

Para o *App Clinica* é necessário realizar o cadastro dos tipos de atendimento clínico existentes e oferecidos aos usuários da UnAPI. No *App Clinica* / *Tipos* é possível realizar o cadastro dos tipos de atendimento oferecidos aos usuários. Para a correta classificação e aproveitamento futuro dos dados é necessário que haja um cadastro da classificação o mais exato possível, pois assim haverá a garantia do aproveitamento futuro das informações geradas. Para os testes realizados foram efetuados o cadastro dos tipos de atendimento clínico: Odontologia, Clinica Geral, Geriatria, Fisioterapia, Psicologia e Radiologia (Figura 49). Vale ressaltar que, havendo mais atendimentos, é necessário o correto cadastramento destes no *App*.

O *App Curso* é responsável por disponibilizar todas as oficinas e cursos oferecidos aos participantes da UnAPI. Para seu correto funcionamento é necessário que sejam cadastrados os setores que oferecem tais oportunidades aos participantes.

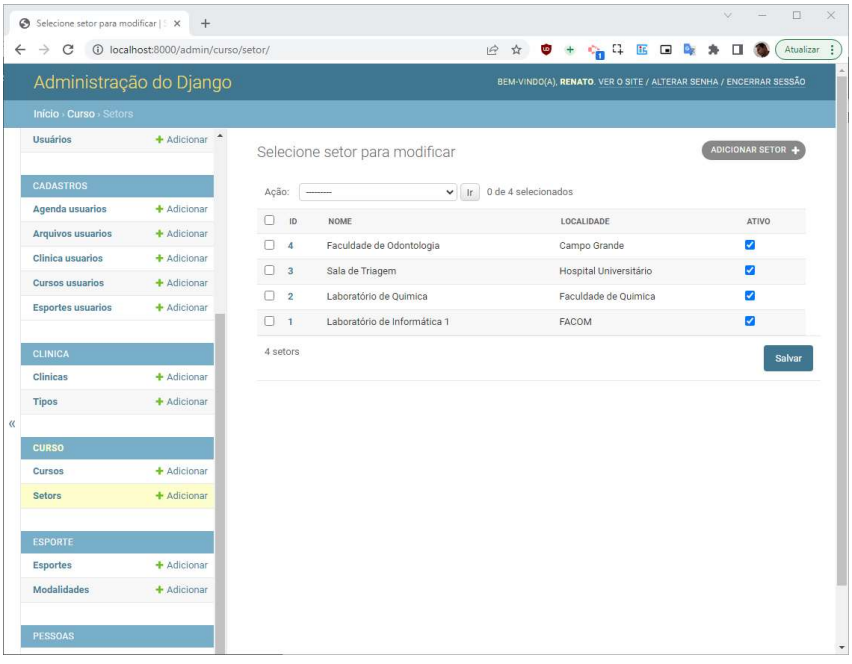
No *App Curso* / *Setors* é possível realizar o cadastramento dos setores envolvidos nessas oficinais e cursos oferecidos. Nesse *APP* também é possível realizar o cadastramento de todos os outros setores envolvidos em atividades junto a UnAPI. No teste foi realizado o cadastramento de quatro exemplo de setores: Laboratório de Informática I, Laboratório de Química, Sala de Triagem e Faculdade de Odontologia (Figura 50).

Figura 49 Cadastramento dos Tipos de Atendimentos Clínicos



Fonte: Autor, 2022

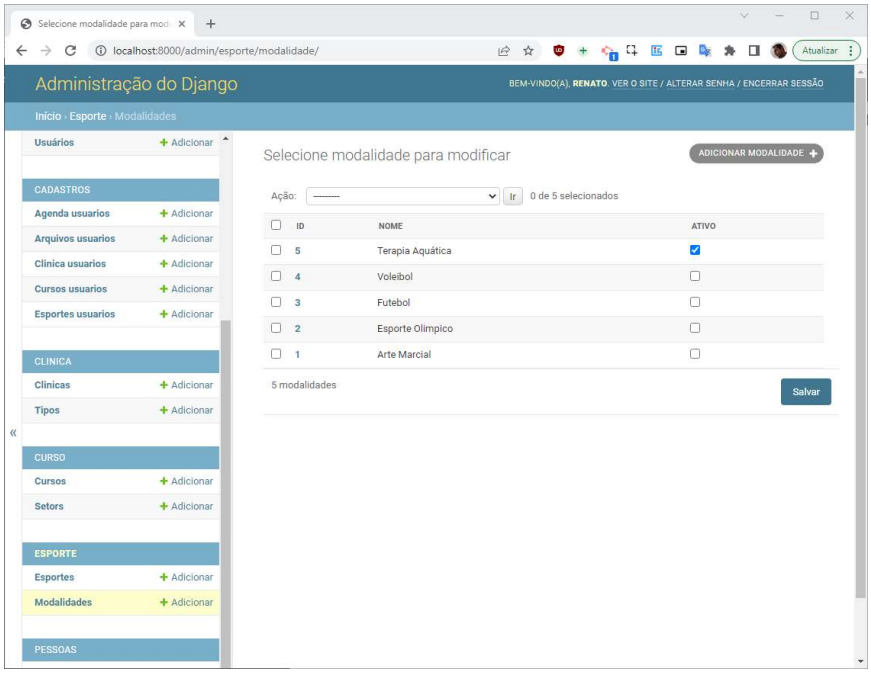
Figura 50 Cadastramento dos Setores Envolvidos



Fonte: Autor, 2022

No *App Esporte* são cadastradas as possíveis atividades esportivas oferecidas aos participantes da UnAPI. Esse *App* foi construído para ser classificado em modalidades e essas necessitam ser cadastradas previamente para a correta classificação. Nos testes foram efetuados o cadastramento das seguintes modalidades: Arte Marcial, Esporte Olímpico, Futebol, Voleibol e Terapia Aquática (Figura 51).

Figura 51 Cadastro de Modalidades Esportivas

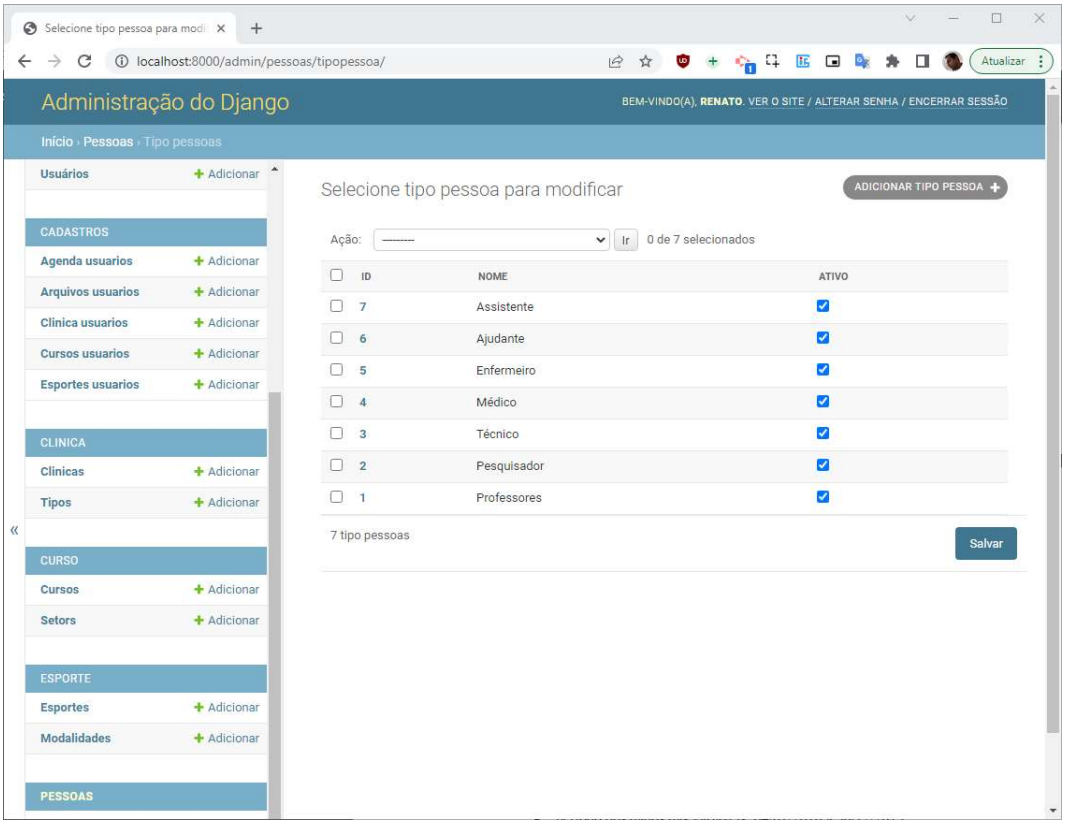


Fonte: Autor, 2022

O *App Pessoa* é responsável pelo cadastro de todas as pessoas envolvidas em atividades na UnAPI. Durante o levantamento dos requisitos do sistema, foi identificado a necessidade da classificação dos envolvidos nos projetos.

Essa classificação foi projetada para garantir que Pessoas que possuam envolvimento com a UnAPI consigam, de maneira rápida, filtrar suas atividades e os usuários envolvidos. No *App* em *Pessoas*/*Tipo pessoa* é possível realizar o cadastramento dos tipos de pessoas envolvidas nos projetos. Nos testes foram realizados os cadastros de tipos de pessoa: Professores, Pesquisador, Técnico, Médico, Enfermeiro, Ajudante e Assistente, conforme Figura 52.

Figura 52 Cadastro dos Tipos de Pessoas



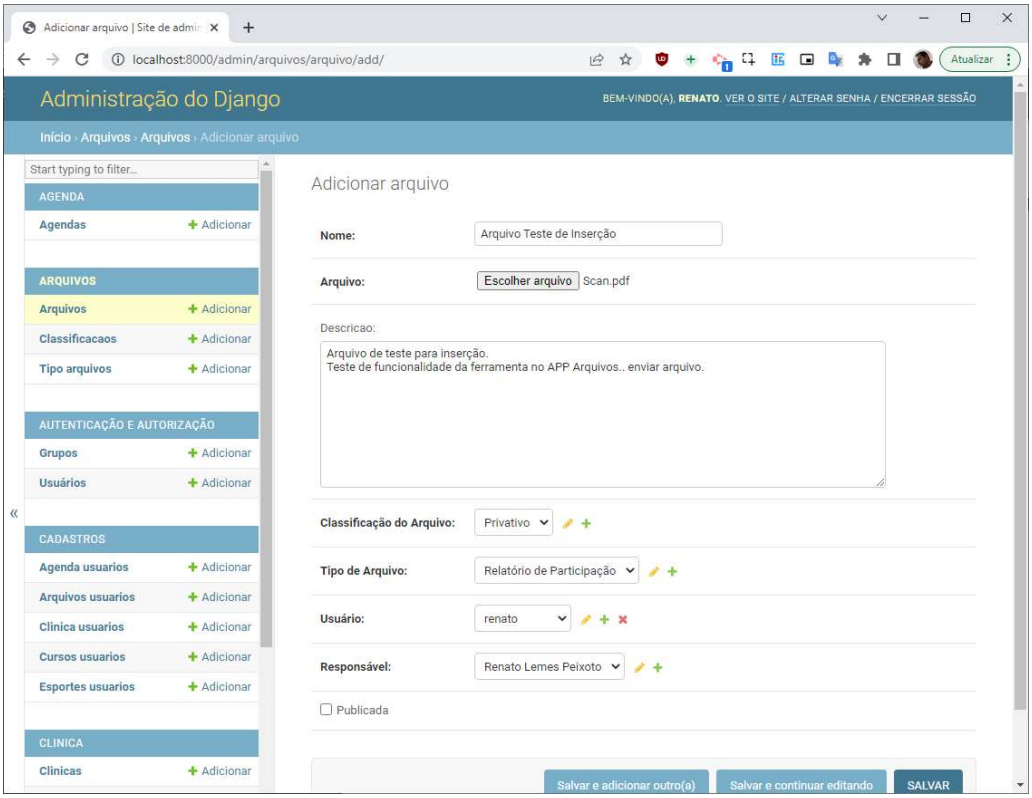
Fonte: Autor, 2022

O perfil administrativo do sistema habilita acesso a todos os módulos desenvolvidos e disponibilizados para utilização. Com esse perfil é possível realizar o cadastro e armazenamento no servidor de um novo arquivo para disponibilização futura.

No *App Arquivo* /*Arquivo* é possível realizar a inserção e cadastro de um novo arquivo dentro do sistema. Como houve o cadastramento anterior dos "tipos de arquivo", "classificação do arquivo" e "responsável" o *App* passa a ter sua funcionalidade totalmente disponível para o correto funcionamento e arquivamento, conforme a tela da Figura 53.

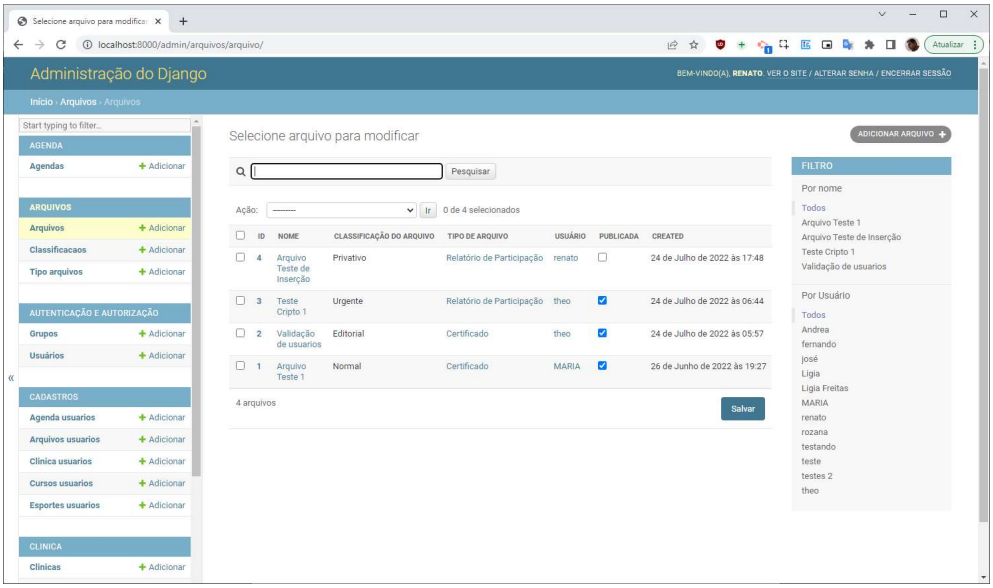
Durante o processo de inserção de uma informação na ferramenta, pode ser necessário o registro de uma nova classificação ou tipo de arquivo não existente. O *framework Django*, em seu perfil administrativo, fornece atalhos onde essa possibilidade é possível. Ao lado da caixa de seleção de cada item há a opção de alterar, incluir ou excluir . Ao selecionar um dos botões, uma nova caixa é apresentada para a confirmação da ação. Essa opção é disponibilizada no perfil administrativo da ferramenta (Figura 54).

Figura 53 Inserção de Novo Arquivo no servidor



Fonte: Autor, 2022

Figura 54 Arquivos inseridos durante os Testes de validação

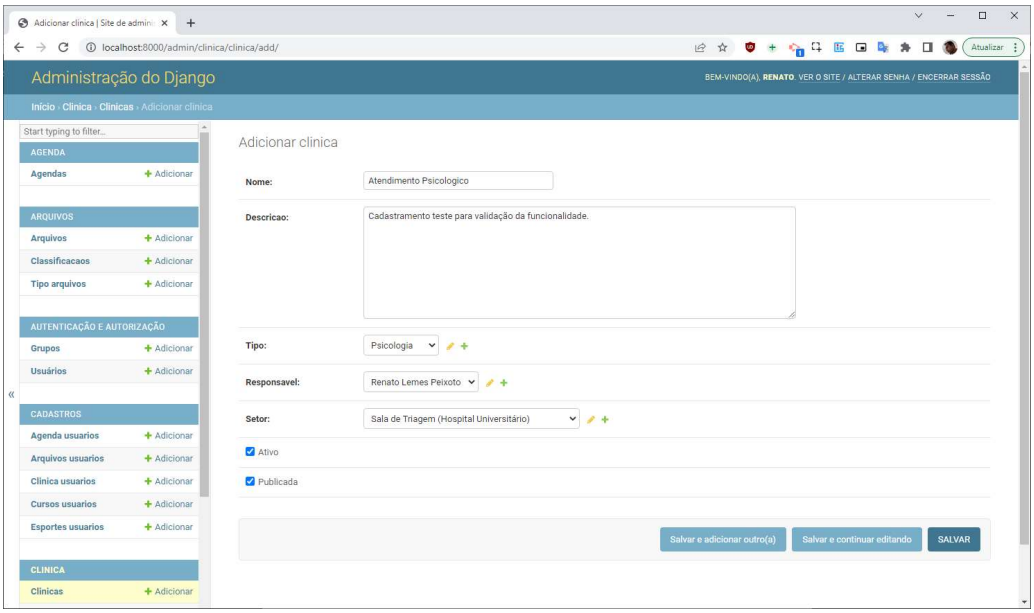


Fonte: Autor, 2022

No *App Clinica* é possível realizar o cadastramento dos atendimentos clínicos oferecidos aos usuários da UnAPI. Após o cadastramento dos Tipos possíveis de Atendimento, o usuário

administrador pode realizar a inserção de novos atendimentos disponíveis. Para os testes de validação foi realizado o cadastramento de um novo atendimento, conforme Figura 55.

Figura 55 Cadastramento de Atendimento Clínico



Fonte: Autor, 2022

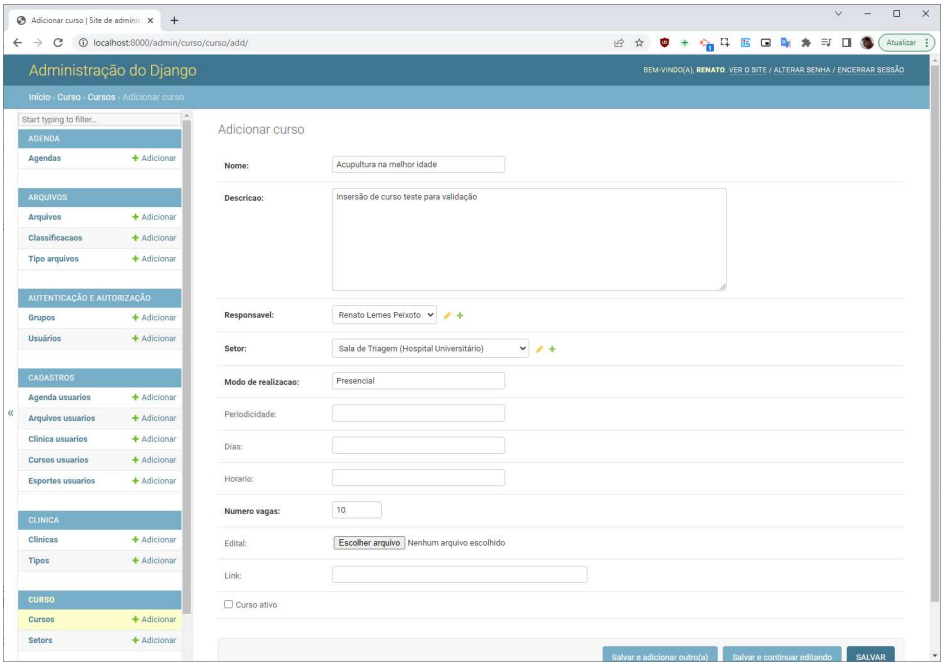
Durante o cadastramento do atendimento é possível definir o cadastro como ativo ou inativo, selecionado o *checkbox* relacionado e se o atendimento clínico será publicado, realizando a seleção no *checkbox* correspondente.

No *App Curso*, após a inclusão dos setores foi realizado os testes de validação com a inclusão de três novos cursos. Assim como o *App Clinica*, após o cadastramento dos setores, é possível realizar a inclusão dos cursos sem maiores dificuldades.

Durante o levantamento de requisitos, verificou-se a necessidade da inclusão de campos como periodicidade, dias e horário de realização do curso, bem como o número de vagas, um espaço para inclusão de um *link* externo e a opção para inclusão de um arquivo, que pode ser um edital ou um documento auxiliar, por exemplo. Os campos obrigatórios para inserção sempre são apresentados em destaque, sendo de fácil visualização durante os trabalhos com a ferramenta.

Assim como apresentado no *App Clinica*, há a opção de inserção, alteração ou exclusão de itens relacionados a outros módulos da ferramenta. O *framework Django* disponibiliza esse acesso de forma universal, dentro do perfil administrativo (Figura 56).

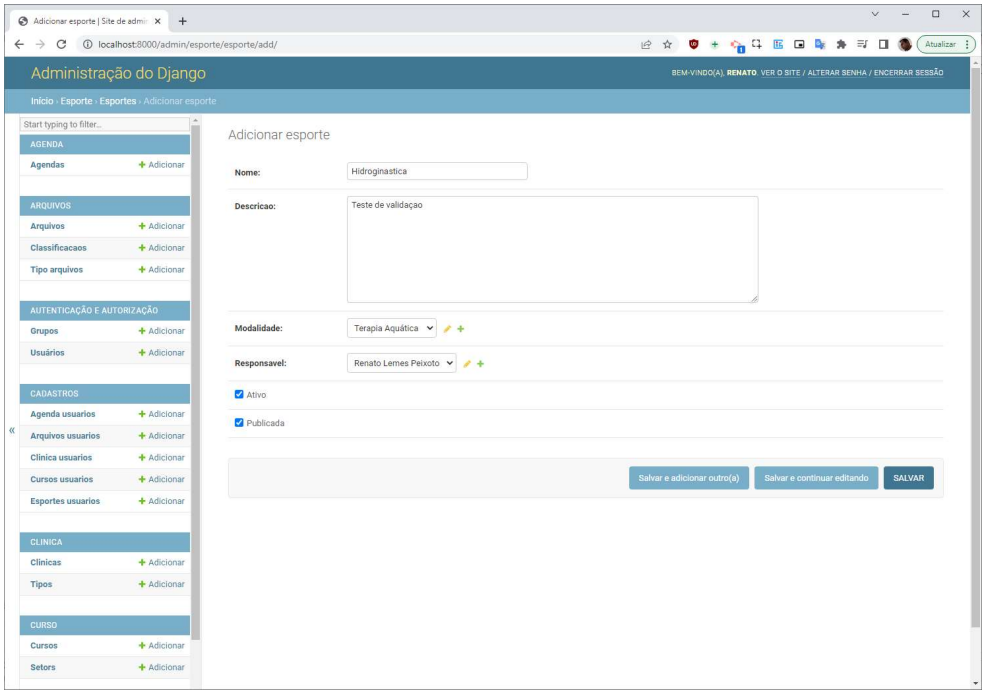
Figura 56 Inclusão de Curso no SISUNAPI.



Fonte: Autor, 2022

Semelhante ao *App Clinica*, o *App Esporte* disponibiliza a inclusão de atividades esportivas disponíveis aos usuários da UnAPI. Para os testes de validação, foram inclusas três novas atividades esportivas (Figura 57). Assim como demais *Apps* e funcionalidades disponíveis na ferramenta, o Responsável é elemento presente e essencial dentro do *App*. Como já mencionado anteriormente, o Responsável será filtro futuro para evidenciar os trabalhos dos pesquisadores e pessoas ligadas a UnAPI e registradas dentro do SISUNAPI.

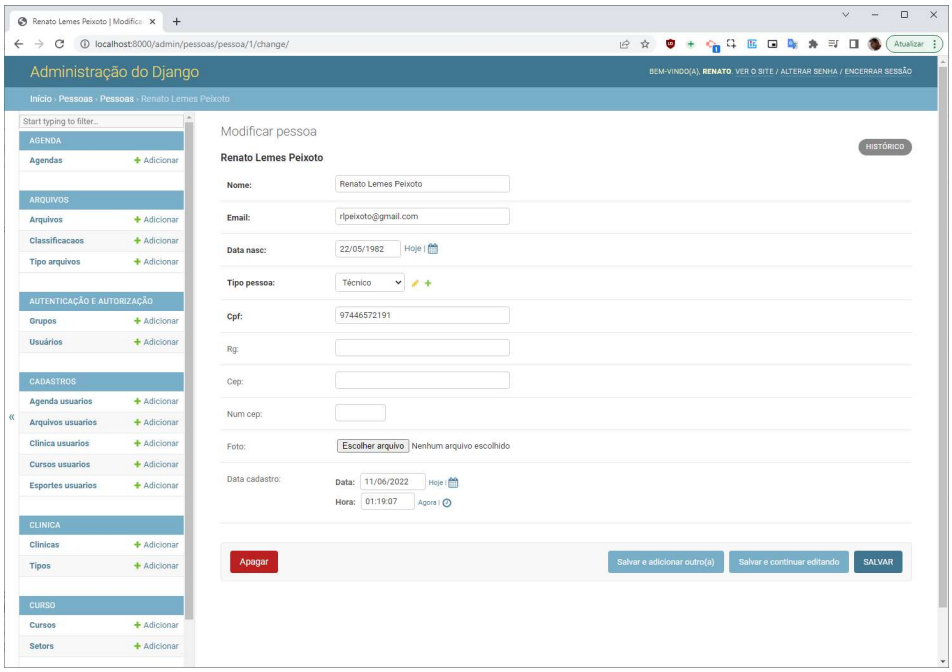
Figura 57 Inserção de Atividade Esportiva



Fonte: Autor, 2022

O *App Pessoa* é o responsável pelo cadastramento de todos os envolvidos em projetos, oficinas, atividades esportivas ou arquivos dentro da ferramenta. Durante os testes de validação foi realizado o cadastramento de um usuário de testes para validação (Figura 58).

Figura 58 Cadastramento de Pessoas no SISUNAPI.

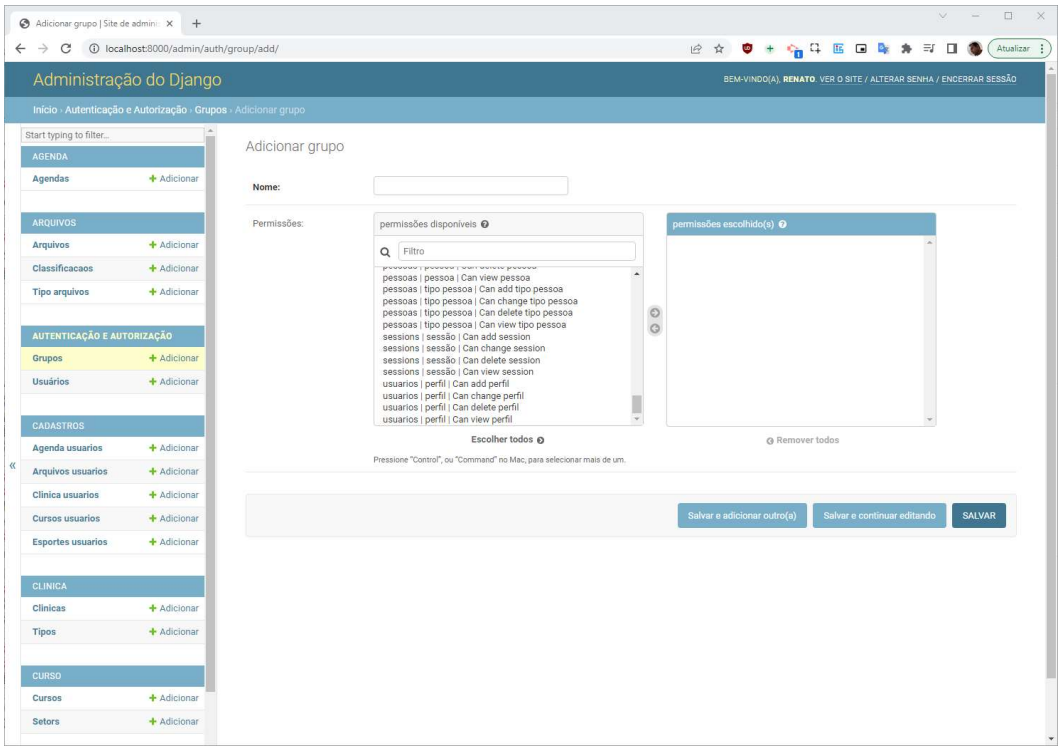


Fonte: Autor, 2022

O perfil administrativo do *framework Django*, por consequência, da ferramenta SISUNAPI oferece um gerenciamento de autenticação e autorização completo. Quando um usuário realiza o cadastro no sistema, ele é automaticamente cadastrado como perfil de usuário. Entretanto, esse perfil não tendo acesso ao painel de administração do *framework* e da ferramenta, e havendo a necessidade de liberar esse acesso, o procedimento é de fácil realização.

Acesso o painel principal do *framework Django*, no *App Autenticação e Autorização*, há dois módulos registrados, um *Grupos* e outro *Usuários* (Figura 59). Durante o levantamento dos requisitos do sistema, entendeu-se que os usuários registrados não necessitam de acesso profundo a ferramenta, sendo esse acesso limitado e autorizado a uma quantidade mínima de pessoas que farão a gestão das informações, cadastros e acessos dentro da ferramenta (Figura 59).

Figura 59 App Autenticação e Usuários - Módulo Grupos.

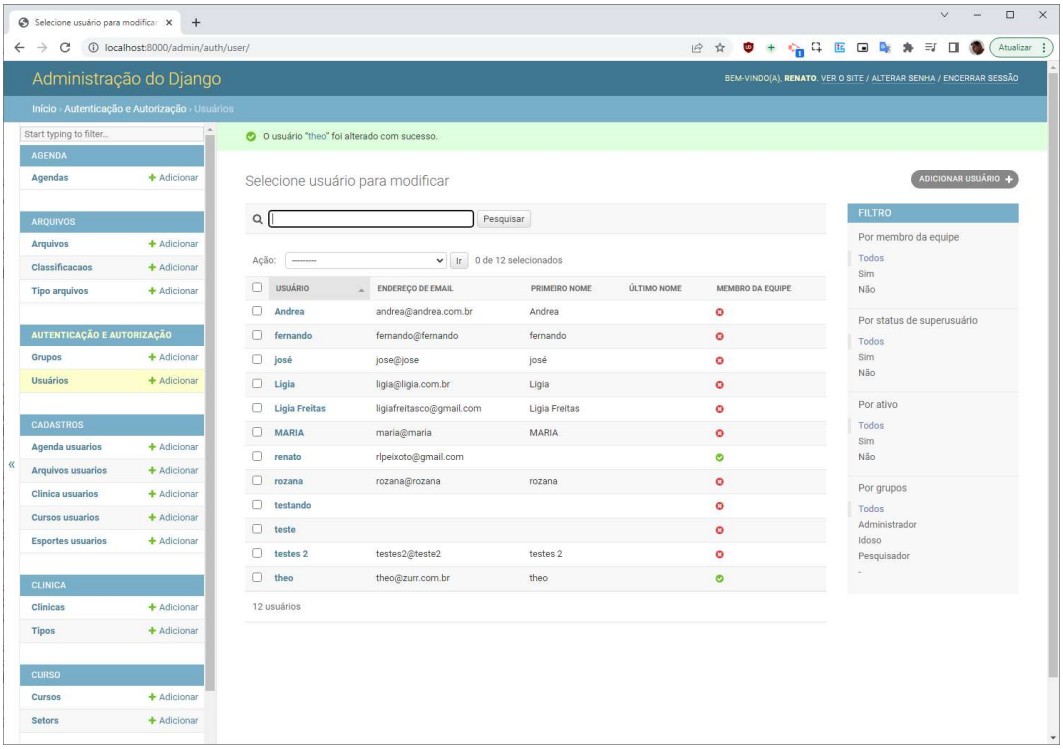


Fonte: Autor, 2022

Para os testes de validação, foram realizados diversos novos cadastros no sistema de usuários, como forma de validar formulários de entrada de dados e métodos de autenticação durante o desenvolvimento da ferramenta.

Durante os testes foram realizados a inclusão de todas as classificações necessárias para o funcionamento da ferramenta. Não houveram falhas para os itens descritos acima.

Figura 60 Cadastro de Usuários - Perfil Administrativo



Fonte: Autor, 2022

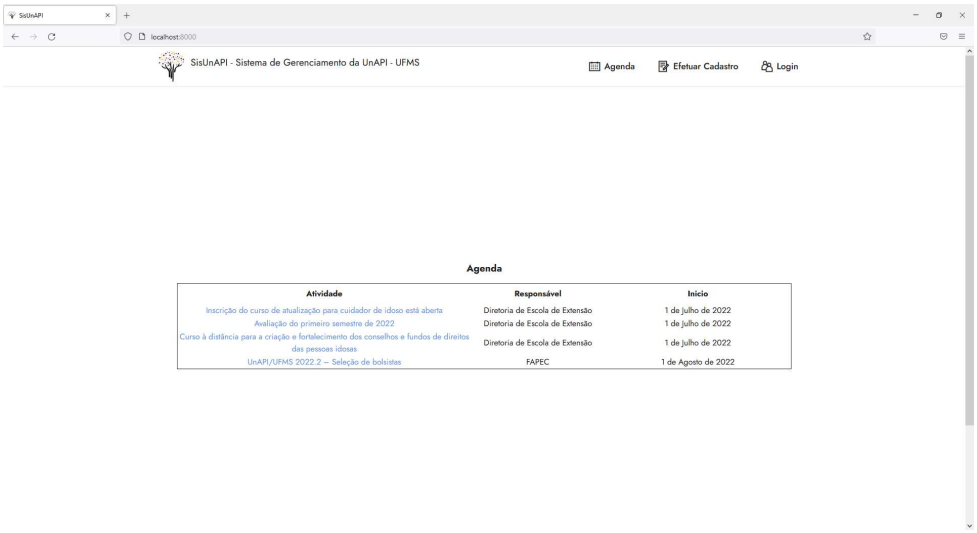
Os casos de uso para o perfil administrativo estão especificados nos Apêndices.

5.2 Utilização da Ferramenta como Usuário

A qualidade é um requisito básico da engenharia de *software*. A utilização de *frameworks*, de ferramentas adequadas e a capacidade de interpretar às necessidades no desenvolvimento de sistema, tornam processos manuais de grande complexidade simples com sistemas automatizados.

Para o usuário do sistema SISUNAP foi pensado uma ferramenta simples, de fácil acesso e usabilidade. Com essa premissa, durante o levantamento de requisitos foi proposto um ambiente de fácil aprendizado. Na página principal do sistema é apresentada a Agenda de atividades disponíveis cadastrados na ferramenta e também são apresentados ícones e texto para acesso aos menus. O menu contém: Agenda, Efetuar Cadastro e *Login* (Figura 61).

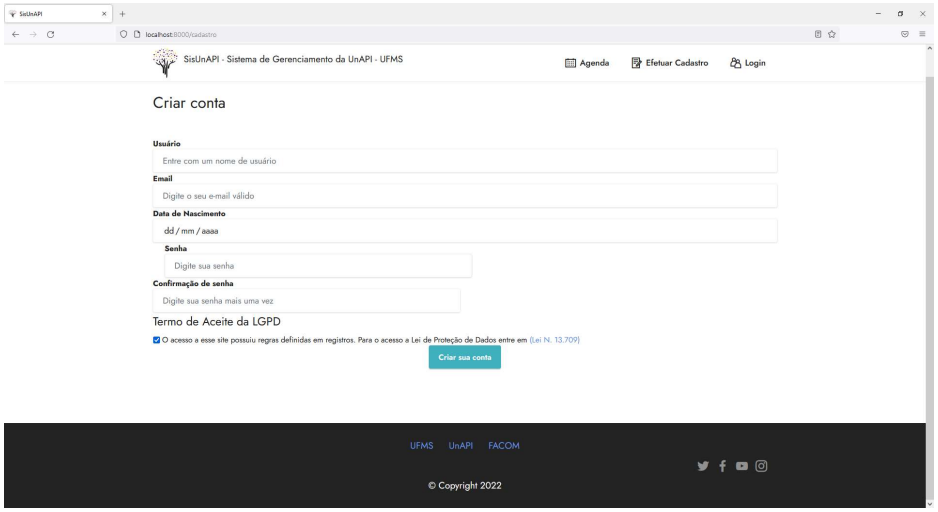
Figura 61 Tela principal do SISUNAPI



Fonte: Autor, 2022

A ferramenta é pública, permitindo que seja efetuado o cadastro por qualquer usuário da rede e que deseje fornecer seus dados para ter acesso ao conteúdo. O usuário para realizar o cadastro (no menu Efetuar Cadastro) deve fornecer dados mínimos ao acessar ao sistema (Figura 62).

Figura 62 Tela de cadastro de usuário

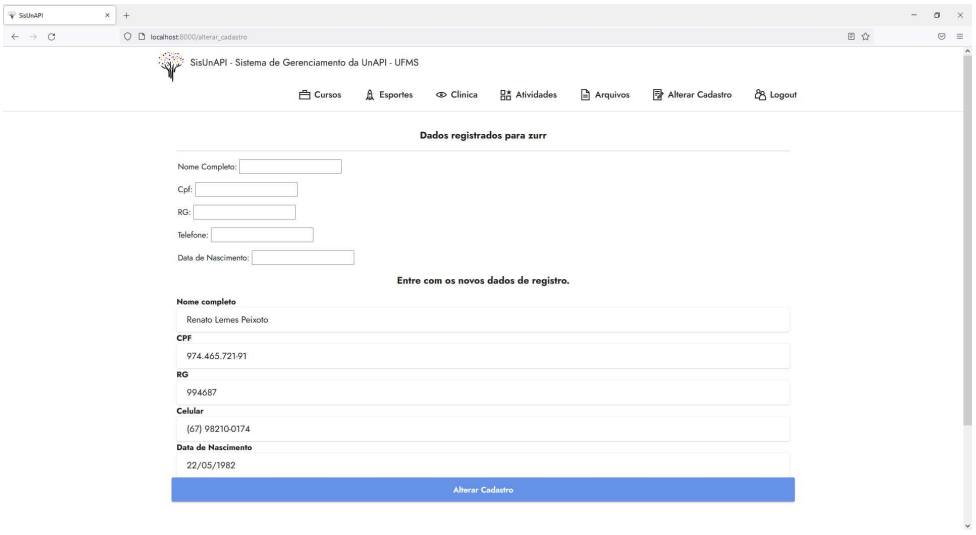


Fonte: Autor, 2022

Os dados são: nome de usuário, *e-mail*, data de nascimento e uma senha, que necessita ser inserida duas vezes para sua validação. Também é necessário aceitar o Termo de Aceite da Lei Geral de Proteção dos Dados (LGPD), após clicar no botão criar conta. Será realizada a validação dos dados inseridos, verificando se o nome de usuário ou e-mail já estão cadastrados no sistema. Estando tudo validado, uma conta é criada e é possível realizar o *login* no sistema.

Ao efetuar o primeiro *login* é apresentado na tela acesso para complementar as informações, que foram avaliadas durante o levantamento de requisitos, e que seriam de importância para projetos e pesquisas futuras. Ao clicar sobre o *link* "Alterar Cadastro" é apresentado uma tela onde são exibidos os dados já cadastrados e um espaço destinado a inserção dos novos dados para complemento e atualização (Figura 63).

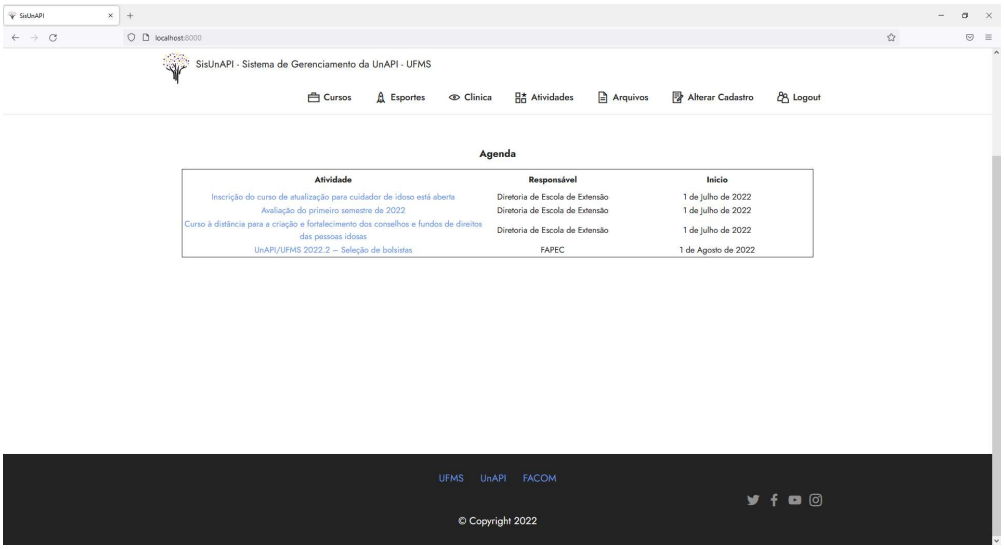
Figura 63 Tela para Inserção dos dados do usuário



Fonte: Autor, 2022

Ao realizar o complemento dos dados e clicando sobre o botão "Alterar Dados" o acesso ao sistema é liberado (Figura 64).

Figura 64 Tela de Acesso principal do SISUNAPI após login



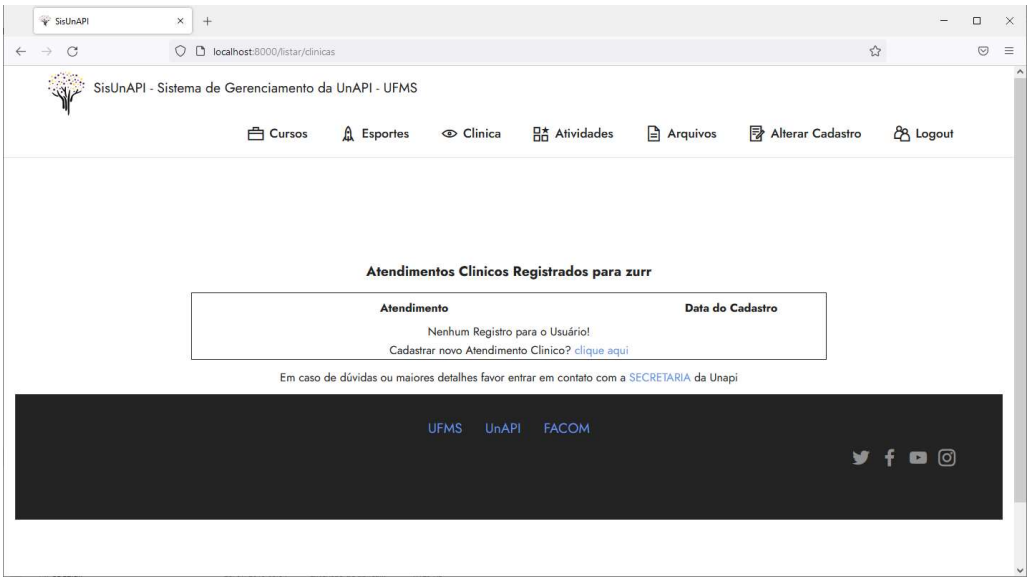
Fonte: Autor, 2022

Na tela de acesso principal, após o *login* do usuário, são apresentados no menu mais acessos a funcionalidades do SISUNAPI. O menu é composto pelos elementos: Cursos,

Esportes, Clínica, Atividades, Arquivos, Alterar Cadastro e *Logout*. Na Figura 65 é apresentado a tela de navegação pelo SISUNAP para atendimentos clínicos.

Assim como o perfil administrativo, cada item do Menu corresponde a um *App* do SISUNAPI, porém com as limitações de usuários implantadas em cada item.

Figura 65 Navegação pelo SISUNAPI, atendimentos clínicos.



Fonte: Autor, 2022

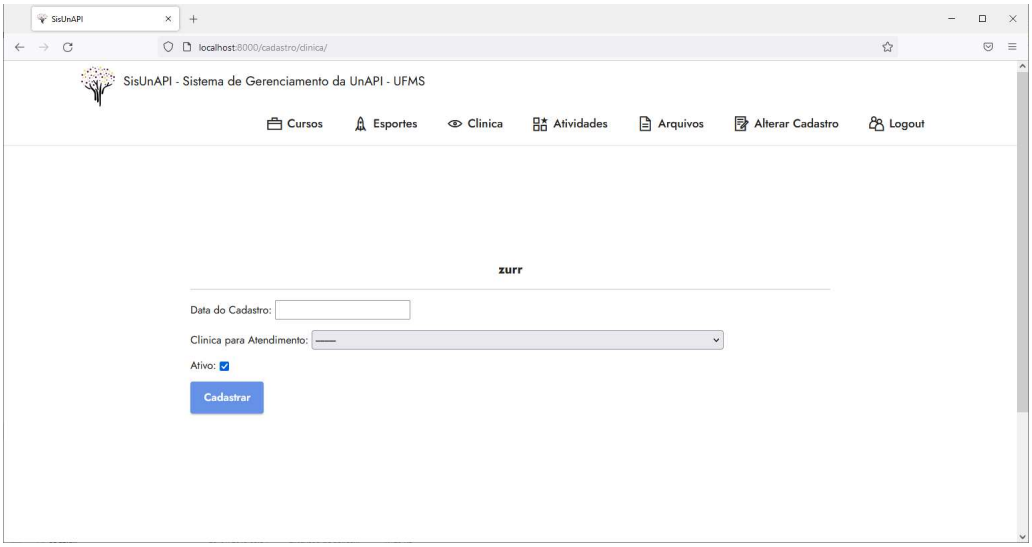
Ao selecionar um item do menu é possível verificar os itens registrados e vinculados ao usuário. Caso não haja nenhum item vinculado, é possível realizar o vínculo de um item, clicando sobre o atalho cadastrar novo. Esse vínculo será realizado com base nos itens previamente cadastrados pelo administrador do sistema, no perfil administrativo do sistema. Na Figura 66 é apresentado o vínculo de um atendimento Clínico ao usuário.

Em nível de usuário foram validados os acessos disponíveis na ferramenta e não foram encontrados problemas durante os testes realizados em ambiente de homologação. Os casos de uso para os perfis comuns de usuário estão especificados nos Apêndices.

Durante o desenvolvido de todo o projeto, foi verificado que muitas atividades que os participantes da UnAPI participam não estavam compartilhando informações. Muitas das atividades esportivas ou atendimentos clínicos não eram armazenados e com isso, deixavam de ser utilizados em pesquisas.

Com a ferramenta SISUNAPI foi possível realizar o armazenamento de dados clínicos e relatórios de atividades esportivas no banco de dados, bem como a organização geral para futuras pesquisas.

Figura 66 Vínculo de um Atendimento Clinico ao usuário.



Fonte: Autor, 2022

Capítulo 6

CONSIDERAÇÕES FINAIS

Nesse capítulo são apresentadas as considerações finais após o desenvolvimento do SISUNAPI, além de sugestões de trabalhos futuros para que o sistema seja cada vez mais aperfeiçoado e utilizado na UNAPI. Dessa forma, espera-se que seja feita a implementação do sistema e que seja utilizado de maneira satisfatória o mais breve possível. Espera-se também que os idosos tenham uma facilidade maior para encontrar as atividades que lhe proporcionam qualidade de vida.

6.1 – Conclusão

É desejo de todo o pesquisador preservar o seu trabalho e promover a divulgação para seus comuns, seja essa divulgação por meios acadêmicos ou na mídia tradicional. Também é comum do ser humano as relações interpessoais com o avanço da idade. As pessoas, com o passar dos anos unem-se em grupos para compartilhar experiências e buscar melhores condições de convivência.

Este trabalho teve como objetivo principal o registro dos trabalhos realizados com os idosos participantes da UnAPI, em seus mais diversos programas de apoio e vivência garantindo a salvaguarda desse material para futuros trabalhos e análises. O objetivo primário do trabalho era o armazenamento, de forma segura, de dados gerados pelas pesquisas dos alunos e professores ligados a UnAPI. Esses dados são gerados e armazenados de forma distribuída, muitas vezes deixados em dispositivos particulares de alunos. Com um repositório unificado, é seguro dizer que, os dados serão centralizados e disponibilizados para futuros projetos. O objetivo principal deste trabalho foi a criação de um repositório para armazenar as informações geradas pelos projetos e demais atividades desenvolvidas na UnAPI. Para desenvolver o

trabalho foi necessário o estudo de como são realizados esses projetos, trabalhos e demais atividades que fornecem os dados.

O protótipo, desenvolvido para testes, obteve de forma satisfatória o armazenamento dos dados em forma de arquivos digitais, conseguindo realizar o cadastro com diversas classificações. Houve também testes de vínculo de atividades com diversos usuários.

Durante o levantamento de requisitos, foi identificado que os usuários idosos poderiam estar vinculados a diversas oficinas dentro da UnAPI e da UFMS. O levantamento de requisitos identificou que o vínculo de um *login* único na busca por todas as informações seria primordial. Todavia, devido a pandemia de COVID-19 não houve possibilidade de reuniões para o alinhamento geral dos requisitos e apresentação dos dados.

A troca do *framework* incialmente estudado para o *framework Django* foi primordial, diante do avanço do tempo e pela facilidade de manutenibilidade ou mesmo adição de novas funcionalidades que vierem a ser solicitadas no SISUNAPI.

O painel administrativo, fornecido pelo *framework*, atende de maneira ágil e adota as melhores práticas de segurança.

Com o avanço dos trabalhos realizados, foi identificado que muitas atividades que são realizadas na UnAPI não eram registradas. Atividades na área de Educação Física para os idosos simplesmente eram deixadas de lado. No SISUNAPI foi criado uma área específica para o registro dessas atividades. Cursos e atendimentos clínicos também passaram a ter importância e seus registros serão realizados.

Foi incluído, na base do SISUNAPI, todos os módulos necessários para a implantação dos relatórios e manipulação de dados.

Sobre o trabalho desenvolvido, acredita-se que é primordial para o início de uma nova etapa para a UnAPI, pois não havia um sistema centralizador de informações e muitas informações geradas por inúmeros projetos foram perdidos. O armazenamento de informações será importante para a UnAPI, com dados de projetos dos professores e dados de todos alunos envolvidos.

Ainda há muito a ser desenvolvido no sistema SISUNAPI e espera-se que este trabalho incentive outros e seja fonte de conhecimento e expansão de novas atividades dentro da UnAPI e da UFMS em seus diversos campos e faculdades. Essa dissertação pretende ser a base para os demais subsistemas que venham a ser implementados.

6.2 Trabalhos Futuros

Como pode ser observado, o correto armazenamento das informações é de importância tanto quanto as pesquisas, os processos de estudo e análise realizados. Também é observado que, a atenção a melhor idade deve estar presente em todas as ações realizadas dentro das pesquisas, tanto da UnAPI quanto de outros setores da UFMS. Um ponto a ser estudado é a união da ferramenta SISUNAPI com o SISCAD – Sistema Acadêmico, pois muitos dos participantes da UnAPI são alunos ligados a cursos de graduação da UFMS. A união dos dois sistemas agregará facilidade de uso para os idosos e professores.

Outro ponto a ser melhorado é com relação ao *framework* do *Front-End*. Hoje a UFMS possui diversos sistemas e *sites* com um *layout* próprio, a modificação do *layout* do protótipo é uma forma de integrar o SISUNAPI ao ecossistema da UFMS. A criação de uma aplicação *mobile*, ligada ao banco de dados, agilizaria ainda mais o acesso por parte dos idosos ligados a UnAPI.

O desenvolvimento de relatórios para facilitar a busca pelas informações é outro ponto a ser considerado em trabalhos futuros.

A escolha pelo *framework Django*, de fácil compreensão e manutenção, foi realizada como forma de agilizar todas as mudanças que forem necessárias nos projetos futuros.

A escolha pela linguagem Python foi motivada para a adição futura de ferramentas de manipulação de dados e tratamento de informações. A ferramenta SISUNAPI foi projetada de forma a facilitar a inclusão de ferramentas de manipulação de dados. Os pacotes necessários para essas manipulações já estão incluídos na base do sistema.

REFERÊNCIAS

4Linux (2022) – **O que é PostgreSQL**. 2022 Disponível em https://4linux.com.br/o-que-e-postgresql/ acessado em 06/07/2022.

AGOSTINHO, B. M.; SCHREINER, G. A.; GOMES, F. O.; PINTO, A. S. R.; DANTAS, M. A.R.**Unificação de Dados de Saúde Através do Uso de *Blockchain* e *SmartContracts***– Programa de Pós-Graduação em Ciência da Computação, UFSC, Florianópolis, 2018SC. Ver se consegue todos os autores

AZRAOUI, M.; Önen, M.; Molva, R. ***Framework for Searchable Encryption with SQL Databases***– *8th International Conference on Cloud Computing and Services Science* (CLOSER 2018), pages 57-67.

BARBOSA, S. R. M.; GUIMARÃES, C. P.; PENHA, R. M.; MEZA, E. R.**Universidade Aberta à Pessoa Idosa: Transformando Realidades**. Revista Barbaquá/UEMS – Dourados – MS, vol. 01, n. 01, p.41-46, jan-jun 2017 ISSN: 2526-9461 (*online*).

BARCELLOS, C. D; MUSA, D. L.; BRANDÃO, A. L.; WARPECHOWSKI, M.**Sistema de Recomendação Acadêmico para Apoio a Aprendizagem**– *Novas Tecnologias na Educação* (CINTED-UFRGS 2007) V. 5 Nº 2, Dezembro, 2007.

BYLEARN (2020) – **Conheça as vantagens do Django, o framework Python para desenvolvimento web**. 2020 Disponível em https://dojo.bylearn.com.br/python/vantagens-do-django-desenvolvimento-web/ acessado em 06/07/2022.

CARROMEU, C. **Linha de Produtos de Software no Processo de Geração de Sistemas Web de Apoio a Gestão de Fomento de Projetos.** (Dissertação de Mestrado). Departamento de Computação e Estatística, UFMS. Campo Grande, MS, Brasil, 2007

CARROMEU, C. (2016).***TitanFrameworkCookbook.*** 2016 Disponível em: <http://www.titanframework.com/Cookbook.pdf>

Coordenadoria de Bibliotecas – UFMS em https://bibliotecas.ufms.br/

Django Documentation - https://docs.djangoproject.com/en/4.0/

Google.com em http://www.google.com.br

Google Acadêmico em https://scholar.google.com.br/

IBGE – Expectativa de Vida do Brasileiro. Disponível em <https://agenciadenoticias.ibge.gov.br/agencia-sala-de-imprensa/2013-agencia-de-noticias/releases/29502-em-2019-expectativa-de-vida-era-de-76-6-anos> acessado em 26/11/2020.

IBICT – Instituto Brasileiro de Informação em Ciência e Tecnologia em https://www.ibict.br/

JOGET (2022) em https://www.joget.org/get-started/

LAMOUNIER, L. (2018). **O Guia Definitivo da Tecnologia Blockchain: Uma Revolução Para Mudar o Mundo**. 2018 Disponível em: <https://101blockchains.com/pt/tecnologia-blockchain-guia/#prettyPhoto> acessado em 12/11/2020.

LIMA, G. (2021). **Bootstrap: O que é, Documentação, como e quando usar**. 2021 Disponível em: <https://www.alura.com.br/artigos/bootstrap> acessado em 18/07/2022.

MACHADO, L. R.; GRANDE, T. P. F.; BEHAR, P. A.; LUNA, F. M. R. (2016). **Mapeamento de Competências Digitais: A inclusão social dos idosos**. ETC – Educação Temática Digital, V. 18, n.4 p. 903 – 921, out./dez.2016. http://dx.doi.org/10.20396/etd.v18i4.8644207

MAGALHÃES, Hudson (2018). **Arquivos Estáticos no Django**. Disponível em < http://www.hmagalhaes.eti.br/?p=587> acessado em 05/08/2022.

Mobirise (2022) em https://mobirise.com/

Nata.House (2022) **O que é um bom software? Identifique essas 5 caracteristicas!** Disponível em: <https://natahouse.com/pt/o-que-e-um-bom-software-identifique-essas-5-caracteristicas> acessado em 19/07/2022.

PostgreSQL – *Advanced Open Source Relational Database* - https://www.postgresql.org/

PRESSMAN, Roger. S. **Engenharia de *Softwares***. 5 ed.Rio de Janeiro: McGraw-Hill, 2002. 844p.

PUBLISH OR PERISH– Disponível em https://harzing.com/resources/publish-or-perish

Python 3.10.5 Documentation - https://docs.python.org/3/

SALES, G. L.; PAZ, R. P.; FRANÇA, A. B. (2014). **Sig@LV: Sistema de Gerenciamento Acadêmico *LearningVectors***. Programa de Pós-Graduação em Ciência da Computação. 3 Congresso Brasileiro de Informática na Educação (CBIE, 2014) Workshops (WCBIE 2014), Fortaleza, 2014, CE.

SILVA, J. A.; MODESTO, A. F.; BRITTO, F. R. P.; GUARIENTI, J. S. C.; de FARIAS, N. C. F., **Universidade Aberta À Pessoa Idosa (UnAPI/UFMS) e as Estratégias Propostas para o Envelhecimento Ativo**, Anais do IX Seminário Regional de Extensão Universitária da Região Centro Oeste**, 2018** ISBN:978-85-99880-66-1

SILVA, T. R.; BUZ, D. F.; PESSOA, N. S. M.; RAMOS, A. J.; PALHANO, M. B.; BARBOSA, A. C. G.; GARCIA, M. C. M., **Inclusão Digital da Terceira Idade**, Grupo de Pesquisa em Inteligência Computacional Aplicada UNESC, Criciúma,SC**.** 2014.

SOKHRANYAEVA T. V. (2015). ***Elderly People Education: Social Effectiveness and Personal Significance.****Chair of Philosophy of Education*, Lomonosov State University, Moscow, Russia, 2015. DOI: 10.15372/PHE20150319

SPÍNOLA, R. O. (2008). **Artigo Engenharia de Software – Introdução à Engenharia de Requisistos.** 2008 Disponível em: <https://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-engenharia-de-requisitos/8034> acessado em 25/11/2020.

SUCUROVIC, S. (2007). ***Implementing security in a distributed web-based ehcr***. *International Journal of Medical Informatics*, 76(5):491 – 496.

TITAN FRAMEWORK – **Um *Framework* PHP para gerenciadores de conteúdo** - http://www.titanframework.com/

VAZ, J. C. T.; DOURADO, R. A.; GOMES, A. S.; RODRIGUES, R. L. (2016) **Requisitos de Dados para uso de Técnicas de Análise Quantitativa de Dados Educacionais em AVAs.** *VII Congresso Brasileiro de Informática na Educação (CBIE 2018). Anais do XXIX Simpósio Brasileiro de Informática na Educação (SBIE 2018)*. Fortaleza, CE, 2018

VECCHIA, R. D.; RUIZ, T.; BOCCHI, S. C. M.; CORRENTE, J. E. (2005). **Qualidade de vida na terceira idade: um conceito subjetivo.** Ver. Bras. Epidemiol. [online]. 2005, vol. 8, n.3, pp.246-252. ISSN 1980-5497 https://doi.org/10.1590/S1415-790X2005000300006

VIANA, Gabriela (2012). **O que é um Host?** Techtudo. Disponivel em < https://www.techtudo.com.br/noticias/2012/02/o-que-e-um-host.ghtml> acessado em 06/08/2022.

WESCHTER, E. O. **Arquitetura do Gerador de Aplicação Web Baseado no *Framework* Titan.** (Dissertação de Mestrado). Departamento de Computação e Estatística, UFMS. Campo Grande, MS, Brasil, 2008

YUAN, Y. and WANG, F. (2018). ***Blockchain and cryptocurrencies: Model, techniques, and applications***. *IEEE Transactionson Systems, Man, andCybernetics: Systems, 48(9):*1421–1428.

Apêndice A

DOCUMENTO DE REQUISITOS

a. Requisitos Funcionais do Sistema

Nome	Descrição	Oculto?
F1 - Registrar Pessoas	O sistema deve registrar pessoas vinculadas a UnAPI. Essas pessoas são responsáveis diretas por funções dentro do sistema, seja ela um professor, pesquisador, aluno, assistente, etc.	Não
F2 - Cadastramento de Agenda de Atividades	O sistema deve permitir o cadastramento de Agendas de Atividades	Não
F3 - Cadastramento de usuários	O sistema deve permitir o cadastramento de Usuários	Não
F4 - Desativar usuário	Como o sistema não pode permitir exclusão de um usuário, deve então permitir a desativação do mesmo.	Não
F5 - Alterar usuário	O sistema deve permitir a alteração de dados de usuário.	Não
F6 - Armazenamento de Arquivos	O sistema deve permitir o armazenamento de arquivos classificados	Não
F7 - Classificar Arquivos	O sistema deve permitir realizar uma classificação dos arquivos, antes e após a sua inclusão	Não
F8 - Cadastramento de Atendimentos Clínicos	O sistema deve permitir o cadastramento de atendimentos clínicos	Não
F9 - Cadastrar Tipos de Atendimentos Clínicos	O sistema deve permitir classificar os atendimentos clínicos antes e após a sua inclusão	Não
F10 - Cadastramento de Cursos	O sistema deve permitir o cadastramento de cursos	Não
F11 - Cadastramento de atividades esportivas	O sistema deve permitir o cadastramento de atividades esportivas	Não

F12 - Cadastramento de Modalidades Esportivas	O sistema deve permitir o cadastramento de modalidades esportivas	Não
F13 - Cadastramento de Setores	O sistema deve permitir o cadastramento de setores vinculados a UNAPI	Não
F14 - Cadastramento de Tipos de Pessoas	O sistema deve permitir o cadastramento de tipos de atividades em que as Pessoas estejam envolvidas	Não
F15 - Gerenciamento de acessos	O sistema deve permitir o gerenciamento de acessos	SIM
F16 - Vínculo de atividade esportiva	O sistema deve permitir o vínculo de uma atividade esportiva a um usuário	Não
F17 - Vínculo de Curso	O sistema deve permitir o vínculo de um curso a um usuário	Não
F18 - Vínculo de Atendimento Clínico	O sistema deve permitir o vínculo de um atendimento clínico a um usuário	Não
F19 - Vínculo de Atividades	O sistema deve permitir o vínculo de uma atividade a um usuário	Não
F20 - Exclusão de Dados	O sistema NÃO DEVE permitir a exclusão de dados	SIM

b. Requisitos Não Funcionais do Sistema

Nome	Descrição	Categoria
NF1 - Interface amigável	O sistema deve ter uma interface amigável e de fácil aprendizado.	Usabilidade
NF2 - Mensagens de Erro	O sistema deve prover mensagens autoinstrutivas como forma de melhorar a experiência do usuário	Usabilidade
NF3 - Prevenção de erros	O sistema deve prover mecanismos de validação de entradas de dados e padronização de informação de campos.	Usabilidade
NF4 - Atalhos	O sistema deve prover teclas de atalho para agilizar as tarefas de usuários mais experientes.	Usabilidade
NF5 - Acesso Restrito	O sistema deverá possuir acesso a informações restritas liberadas mediante a cadastro e acesso com usuários e senhas válidos.	Funcional
NF6 - Perfis de usuário	Os perfis de usuário para acesso ao sistema são: 1. Administrador – pode efetuar todas as operações; 2. Usuário – pode efetuar vínculo de atividades e acessos aos ambientes ao qual está vinculado; 3. Professor/pesquisador – pode efetuar consultas e inserção de dados e vincular usuários aos arquivos inseridos	Funcional
NF7 - Criptografia	Todas as senhas de usuário devem passar por um processo de criptografia.	Funcional
NF8 - Tempo de Resposta	Uma consulta ao banco de dados deve ter um tempo de resposta máximo.	Desempenho
NF9 - Backup	O sistema deve fazer backups regulares dos dados para restaurações que se fizerem necessárias.	Confiabilidade

NF10 - Log	O sistema deve manter arquivos de Log atualizados constantemente, para auxiliar no processo de recuperação de falhas.	Confiabilidade
NF11 - Validação de dados	O sistema deverá possuir validações de campos de entrada de dados a fim de evitar erros.	Confiabilidade
NF12 - Armazenamento de dados	A camada de persistência deve ser implementada de forma que diferentes tecnologias de banco de dados possam vir a ser utilizadas no futuro.	Facilidade de Suporte
NF13 - Documentação do sistema	A documentação do sistema deve se manter completa e atualizada e seus códigos-fonte devidamente comentados a fim de auxiliar a futuras manutenções e/ou evoluções.	Facilidade de Suporte
NF14 - Portabilidade	O sistema deve ser multiplataforma, ou seja, poderá ser utilizado em vários ambientes, com diferentes sistemas operacionais e arquitetura.	Facilidade de Suporte

Apêndice B

CASOS DE USO

a. Casos de Uso – perfil Administrativo e perfil Usuário

Caso de Uso: 1 – Registrar Pessoa
Atores: Administrador do Sistema
Interessados: professor, pesquisador, assistente, aluno
Pré-condições: Deve estar vinculado a UnAPI e Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Poderá ter arquivos e projetos vinculados em seu nome no sistema.
Requisitos correlacionados: F1
Fluxo Principal: 1. A pessoa solicita a inclusão de seus dados no sistema. 2. A pessoa fornece os dados pessoais para cadastro no sistema. 3. O administrador acessa o sistema com o seu login e senha. 4. O administrador acessa a área de registro de pessoas. 5. O administrador realiza o registro do usuário no sistema. 6. O administrador verifica se a função da pessoa está registrada. 7. Caso esteja seleciona a opção, caso contrário [F14]

Caso de Uso: 2 – Cadastramento de Agenda de Atividades
Atores: Administrador do Sistema
Interessados: professor, pesquisador, assistente, aluno, usuário, público em geral
Pré-condições: Ter um acesso ao sistema com o perfil administrativo.
Pós-condições: Evento com possibilidade de vínculo dos usuários.
Requisitos correlacionados: F2
Fluxo Principal: 1. É solicitado a inclusão de uma nova atividade no sistema. 2. O administrador acessa o sistema com o seu login e senha. 3. O administrador acessa a área de registro de atividades. 4. O administrador realiza o registro da atividade no sistema.

Caso de Uso: 3 – Cadastramento de usuários
Atores: Público em Geral
Interessados: público em geral
Pré-condições: Não possuir acesso ao sistema, possuir e-mail válido
Pós-condições: Cadastro de dados complementares solicitados após o primeiro *login* no sistema.
Requisitos correlacionados: F3
Fluxo Principal: 1. A pessoa interessa acessa a página para cadastro. 2. A pessoa interessa entra com os dados solicitados validando o formulário. 3. A pessoa confirma os dados e realiza o cadastro.

Caso de Uso: 4 – Desativar usuário
Atores: Administrador do sistema
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao administrativo ao sistema.
Pós-condições: Ter o seu acesso bloqueado
Requisitos correlacionados: F4
Fluxo Principal: 1. O administrador do sistema realiza o login e senha. 2. O administrador entre na página administrativa do sistema e localiza o cadastro do usuário. 3. O administrador bloqueia o acesso do usuário, inativando o mesmo.

Caso de Uso: 5 – Alterar usuário
Atores: usuários do sistema, administrador, professores, etc.
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com perfil administrativo.
Pós-condições: Ter o seu cadastro alterado conforme inclusões.
Requisitos correlacionados: F5
Fluxo Principal: 1. O administrador acessa o sistema com o seu login e senha. 2. O administrador acessa a área de alteração de dados. 3. O administrador insere os dados que deseja alterar. 4. O sistema retorna a mensagem informando sobre a alteração dos dados.

Caso de Uso: 6 – Armazenamento de Arquivos
Atores: administrador, professores, pesquisadores, alunos e assistentes
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Arquivo incluso para vínculo aos usuários
Requisitos correlacionados: F6
Fluxo Principal: 1. O administrador acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O administrador acessa a área para inserção de arquivos. 3. O administrador insere os dados solicitados e inclui o arquivo para inserção; 4. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 7 – Classificar Arquivos
Atores: administrador, professores, pesquisadores, alunos e assistentes
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil com essa posição
Pós-condições: Classificação de arquivos
Requisitos correlacionados: F7
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de tipos de classificação de arquivos. 3. O usuário insere os dados solicitados. 4. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 8 – Cadastramento de Atendimentos Clínicos
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização de atendimentos clínicos para vínculo dos usuários
Requisitos correlacionados: F8
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de atendimentos clínicos. 3. O usuário insere os dados solicitados. 4. O usuário selecionado o tipo de atendimento, caso não encontre [F9]. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 9 – Cadastrar Tipos de Atendimentos Clínicos
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização de tipos de atendimentos clínicos
Requisitos correlacionados: F9
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de tipos de atendimentos clínicos. 3. O usuário insere os dados solicitados. 4. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 10 – Cadastramento de Cursos
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização de cursos para vínculo dos usuários
Requisitos correlacionados: F10
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de cursos. 3. O usuário insere os dados solicitados. 4. O usuário seleciona o setor responsável, caso não esteja cadastrado [F13]. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 11 – Cadastramento de Atividade Esportiva
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização de atividades esportivas para vínculo dos usuários
Requisitos correlacionados: F10
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de atividades esportivas. 3. O usuário insere os dados solicitados. 4. O usuário seleciona o setor responsável, caso não esteja cadastrado [F13]. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 12 – Cadastramento de Modalidades Esportiva
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização de atividades esportivas para vínculo dos usuários
Requisitos correlacionados: F12
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área de atividades esportivas. 3. O usuário acessa a área para inserção de modalidades esportivas 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 13 – Cadastramento de Setores
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização de setores para cadastro
Requisitos correlacionados: F13
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de setores administrativos 3. O usuário insere os dados solicitados. 4. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 14 – Cadastramento de Tipos de Pessoas
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização do cadastro dos responsáveis pelas atividades
Requisitos correlacionados: F14
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área para inserção de pessoas. 3. O usuário insere os dados referente ao tipo de classificação das pessoas (professor, pesquisador, estudante, etc.) 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 15 – Gerenciamento de Acessos
Atores: administrador
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema com o perfil administrativo
Pós-condições: Disponibilização perfis de acesso personalizados
Requisitos correlacionados: F15
Fluxo Principal: 1. O usuário acessa o sistema, no perfil administrativo, com o seu login e senha. 2. O usuário acessa a área de autenticação e gerenciamento de usuários. 3. O usuário acessa o perfil de acessos. 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 16 – Vínculo de atividade Esportiva
Atores: usuário
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema
Pós-condições: Vincular o seu perfil junto a uma atividade esportiva
Requisitos correlacionados: F16
Fluxo Principal: 1. O usuário acessa o sistema com o seu login e senha. 2. O usuário acessa a área de atividades esportivas. 3. O usuário vincula o seu perfil a atividade desejada 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 17 – Vínculo de cursos
Atores: usuário
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema
Pós-condições: Vincular o seu perfil junto a um curso
Requisitos correlacionados: F17
Fluxo Principal: 1. O usuário acessa o sistema com o seu login e senha. 2. O usuário acessa a área de cursos. 3. O usuário vincula o seu perfil a um curso desejado. 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 18 – Vínculo de Atendimento Clínico
Atores: usuário
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema
Pós-condições: Vincular o seu perfil vinculado a um atendimento clínico
Requisitos correlacionados: F18
Fluxo Principal: 1. O usuário acessa o sistema com o seu login e senha. 2. O usuário acessa a área de atendimento clínico 3. O usuário vincula o seu perfil ao atendimento desejada 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

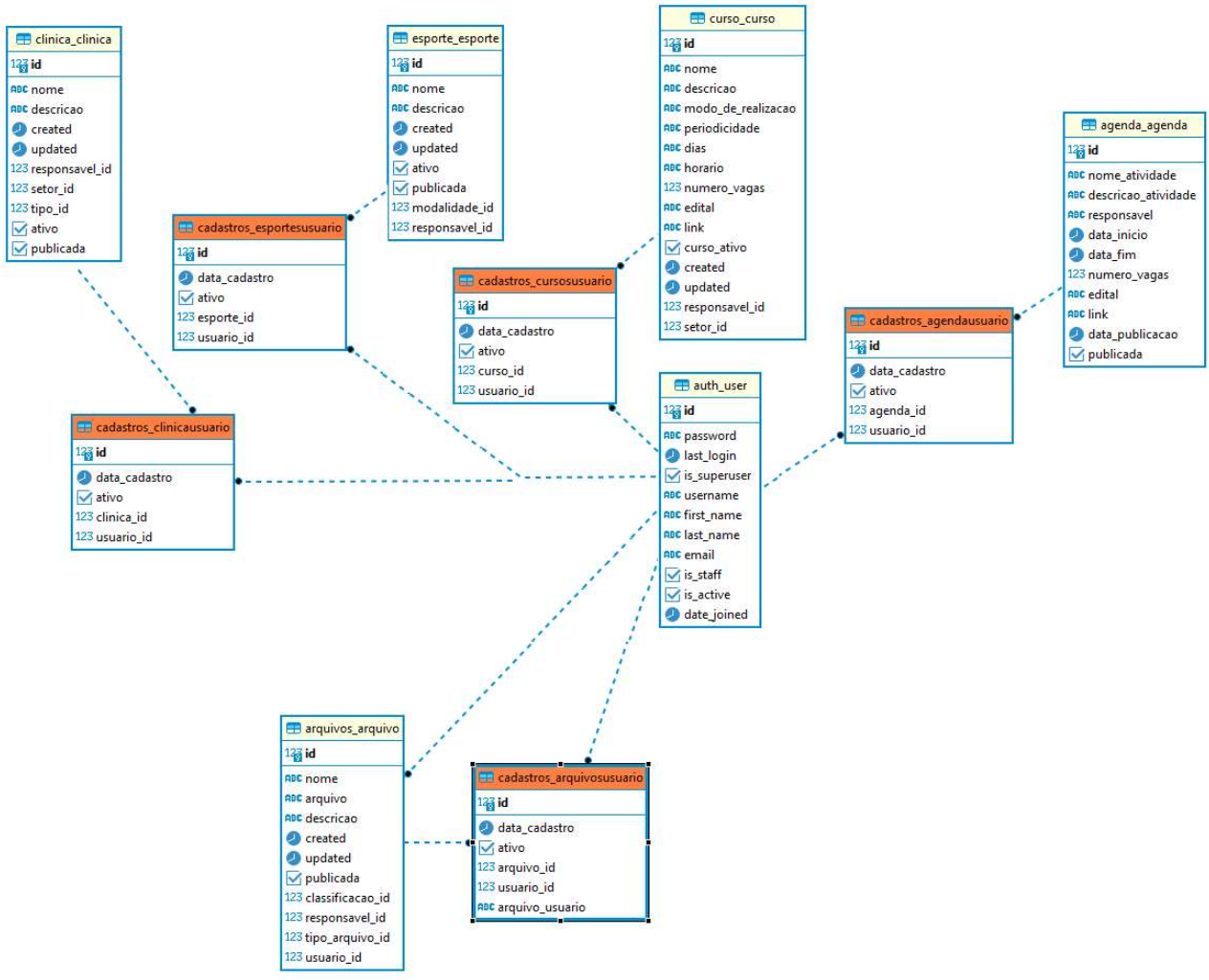
Caso de Uso: 19 – Vínculo de atividades - Agenda
Atores: usuário
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema
Pós-condições: Vincular o seu perfil junto a uma atividade - Agenda
Requisitos correlacionados: F19
Fluxo Principal: 1. O usuário acessa o sistema com o seu login e senha. 2. O usuário acessa a área de atividades - Agenda. 3. O usuário vincula o seu perfil a atividade desejada. 4. O usuário insere os dados solicitados. 5. O sistema retorna a mensagem informando sobre a inclusão dos dados.

Caso de Uso: 20 – Exclusão de Dados
Atores: usuário
Interessados: usuários do sistema, administradores, professores, etc.
Pré-condições: Ter um acesso ao sistema
Pós-condições: Usuário não deve excluir dados do sistema
Requisitos correlacionados: F20
Fluxo Principal: 1. O usuário acessa o sistema com o seu login e senha. 2. O usuário solicita a exclusão dos dados. 3. O sistema retorna a mensagem informando que não é possível realizar a exclusão dos dados.

Apêndice C

MODELO CONCEITUAL

a. Modelo Conceitual – Estrutura do Banco de Dados



Apêndice D

SCRIPTS DE GERAÇÃO DE TABELAS

a. Scripts SQL de Geração das Tabelas no Banco de Dados

```
CREATE TABLE public.agenda_agenda (
	id bigserial NOT NULL,
	nome_atividade varchar(200) NOT NULL,
	descricao_atividade text NOT NULL,
	responsavel varchar(150) NOT NULL,
	data_inicio date NOT NULL,
	data_fim date NOT NULL,
	numero_vagas int4 NOT NULL,
	edital varchar(100) NOT NULL,
	link varchar(200) NULL,
	data_publicacao timestamptz NULL,
	publicada bool NOT NULL,
	CONSTRAINT agenda_agenda_pkey PRIMARY KEY (id)
);


CREATE TABLE public.arquivos_arquivo (
	id bigserial NOT NULL,
	nome varchar(255) NOT NULL,
	arquivo varchar(100) NOT NULL,
	descricao text NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	publicada bool NOT NULL,
	classificacao_id int8 NOT NULL,
	responsavel_id int8 NOT NULL,
	tipo_arquivo_id int8 NOT NULL,
	usuario_id int4 NOT NULL,
	CONSTRAINT arquivos_arquivo_pkey PRIMARY KEY (id)
);
```

```
CREATE TABLE public.arquivos_classificacao (
	id bigserial NOT NULL,
	nome varchar(150) NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	ativo bool NOT NULL,
	CONSTRAINT arquivos_classificacao_pkey PRIMARY KEY (id)
);


CREATE TABLE public.arquivos_tipoarquivo (
	id bigserial NOT NULL,
	nome varchar(150) NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	ativo bool NOT NULL,
	CONSTRAINT arquivos_tipoarquivo_pkey PRIMARY KEY (id)
);


CREATE TABLE public.clinica_tipo (
	id bigserial NOT NULL,
	nome varchar(150) NOT NULL,
	ativo bool NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	CONSTRAINT clinica_tipo_pkey PRIMARY KEY (id)
);


CREATE TABLE public.curso_setor (
	id bigserial NOT NULL,
	nome varchar(255) NOT NULL,
	localidade varchar(250) NOT NULL,
	ativo bool NOT NULL,
	CONSTRAINT curso_setor_pkey PRIMARY KEY (id)
);


CREATE TABLE public.esporte_modalidade (
	id bigserial NOT NULL,
	nome varchar(150) NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	ativo bool NOT NULL,
	CONSTRAINT esporte_modalidade_pkey PRIMARY KEY (id)
);


CREATE TABLE public.pessoas_tipopessoa (
	id bigserial NOT NULL,
	nome varchar(200) NOT NULL,
	ativo bool NOT NULL,
	CONSTRAINT pessoas_tipopessoa_pkey PRIMARY KEY (id)
);
```

```
CREATE TABLE public.cadastros_agendausuario (
	id bigserial NOT NULL,
	data_cadastro timestamptz NOT NULL,
	ativo bool NOT NULL,
	agenda_id int8 NOT NULL,
	usuario_id int4 NOT NULL,
	CONSTRAINT cadastros_agendausuario_agenda_id_usuario_id_6c8573ec_uniq UNIQUE (agenda_id, usuario_id),
	CONSTRAINT cadastros_agendausuario_pkey PRIMARY KEY (id)
);
CREATE INDEX cadastros_agendausuario_agenda_id_04a5b271 ON public.cadastros_agendausuario USING btree (agenda_id);
CREATE INDEX cadastros_agendausuario_usuario_id_2f421fd9 ON public.cadastros_agendausuario USING btree (usuario_id);


CREATE TABLE public.cadastros_arquivosusuario (
	id bigserial NOT NULL,
	data_cadastro timestamptz NOT NULL,
	ativo bool NOT NULL,
	arquivo_id int8 NOT NULL,
	usuario_id int4 NOT NULL,
	arquivo_usuario varchar(100) NOT NULL,
	CONSTRAINT cadastros_arquivosusuario_arquivo_id_usuario_id_3e812d7c_uniq UNIQUE (arquivo_id, usuario_id),
	CONSTRAINT cadastros_arquivosusuario_pkey PRIMARY KEY (id)
);
CREATE INDEX cadastros_arquivosusuario_arquivo_id_4b120a3d ON public.cadastros_arquivosusuario USING btree (arquivo_id);
CREATE INDEX cadastros_arquivosusuario_usuario_id_f8af65b3 ON public.cadastros_arquivosusuario USING btree (usuario_id);


CREATE TABLE public.cadastros_clinicausuario (
	id bigserial NOT NULL,
	data_cadastro timestamptz NOT NULL,
	ativo bool NOT NULL,
	clinica_id int8 NOT NULL,
	usuario_id int4 NOT NULL,
	CONSTRAINT cadastros_clinicausuario_clinica_id_usuario_id_827c1714_uniq UNIQUE (clinica_id, usuario_id),
	CONSTRAINT cadastros_clinicausuario_pkey PRIMARY KEY (id)
);
CREATE INDEX cadastros_clinicausuario_clinica_id_89ab6241 ON public.cadastros_clinicausuario USING btree (clinica_id);
CREATE INDEX cadastros_clinicausuario_usuario_id_d714197e ON public.cadastros_clinicausuario USING btree (usuario_id);
```

```
CREATE TABLE public.cadastros_cursosusuario (
	id bigserial NOT NULL,
	data_cadastro timestamptz NOT NULL,
	ativo bool NOT NULL,
	curso_id int8 NOT NULL,
	usuario_id int4 NOT NULL,
	CONSTRAINT cadastros_cursosusuario_curso_id_usuario_id_b5ce0f4b_uniq UNIQUE (curso_id, usuario_id),
	CONSTRAINT cadastros_cursosusuario_pkey PRIMARY KEY (id)
);
CREATE INDEX cadastros_cursosusuario_curso_id_a06d55ed ON public.cadastros_cursosusuario USING btree (curso_id);
CREATE INDEX cadastros_cursosusuario_usuario_id_360abfbc ON public.cadastros_cursosusuario USING btree (usuario_id);


CREATE TABLE public.cadastros_esportesusuario (
	id bigserial NOT NULL,
	data_cadastro timestamptz NOT NULL,
	ativo bool NOT NULL,
	esporte_id int8 NOT NULL,
	usuario_id int4 NOT NULL,
	CONSTRAINT cadastros_esportesusuario_esporte_id_usuario_id_04e5e465_uniq UNIQUE (esporte_id, usuario_id),
	CONSTRAINT cadastros_esportesusuario_pkey PRIMARY KEY (id)
);
CREATE INDEX cadastros_esportesusuario_esporte_id_02e72157 ON public.cadastros_esportesusuario USING btree (esporte_id);
CREATE INDEX cadastros_esportesusuario_usuario_id_3f8e7357 ON public.cadastros_esportesusuario USING btree (usuario_id);

CREATE TABLE public.clinica_clinica (
	id bigserial NOT NULL,
	nome varchar(255) NOT NULL,
	descricao text NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	responsavel_id int8 NOT NULL,
	setor_id int8 NOT NULL,
	tipo_id int8 NOT NULL,
	ativo bool NOT NULL,
	publicada bool NOT NULL,
	CONSTRAINT clinica_clinica_pkey PRIMARY KEY (id)
);
```

```
CREATE INDEX clinica_clinica_responsavel_id_19ea0c34 ON public.clinica_clinica USING btree (responsavel_id);
CREATE INDEX clinica_clinica_setor_id_6357fb27 ON public.clinica_clinica USING btree (setor_id);
CREATE INDEX clinica_clinica_tipo_id_021e8534 ON public.clinica_clinica USING btree (tipo_id);


CREATE TABLE public.curso_curso (
	id bigserial NOT NULL,
	nome varchar(255) NOT NULL,
	descricao text NOT NULL,
	modo_de_realizacao varchar(250) NOT NULL,
	periodicidade varchar(250) NULL,
	dias varchar(250) NULL,
	horario varchar(250) NULL,
	numero_vagas int4 NOT NULL,
	edital varchar(100) NOT NULL,
	link varchar(200) NULL,
	curso_ativo bool NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	responsavel_id int8 NOT NULL,
	setor_id int8 NOT NULL,
	CONSTRAINT curso_curso_pkey PRIMARY KEY (id)
);
CREATE INDEX curso_curso_responsavel_id_ff9db8d8 ON public.curso_curso USING btree (responsavel_id);
CREATE INDEX curso_curso_setor_id_4c994f58 ON public.curso_curso USING btree (setor_id);


CREATE TABLE public.esporte_esporte (
	id bigserial NOT NULL,
	nome varchar(255) NOT NULL,
	descricao text NOT NULL,
	created timestamptz NOT NULL,
	updated timestamptz NOT NULL,
	ativo bool NOT NULL,
	publicada bool NOT NULL,
	modalidade_id int8 NOT NULL,
	responsavel_id int8 NOT NULL,
	CONSTRAINT esporte_esporte_pkey PRIMARY KEY (id)
);
```

```
CREATE INDEX esporte_esporte_modalidade_id_37f2b0e3 ON public.esporte_esporte USING btree (modalidade_id);
CREATE INDEX esporte_esporte_responsavel_id_8f0353be ON public.esporte_esporte USING btree (responsavel_id);


CREATE TABLE public.pessoas_pessoa (
	id bigserial NOT NULL,
	nome varchar(200) NOT NULL,
	email varchar(200) NOT NULL,
	data_nasc date NOT NULL,
	cpf varchar(14) NOT NULL,
	rg varchar(50) NULL,
	cep varchar(10) NULL,
	num_cep int4 NULL,
	foto varchar(100) NULL,
	data_cadastro timestamptz NULL,
	tipo_pessoa_id int8 NOT NULL,
	CONSTRAINT pessoas_pessoa_pkey PRIMARY KEY (id)
);


CREATE TABLE public.usuarios_perfil (
	id bigserial NOT NULL,
	nome_completo varchar(250) NULL,
	cpf varchar(14) NULL,
	rg varchar(50) NULL,
	telefone varchar(16) NULL,
	data_nasc date NULL,
	usuario_id int4 NOT NULL,
	CONSTRAINT usuarios_perfil_pkey PRIMARY KEY (id),
	CONSTRAINT usuarios_perfil_usuario_id_key UNIQUE (usuario_id)
);
```

Apêndice E

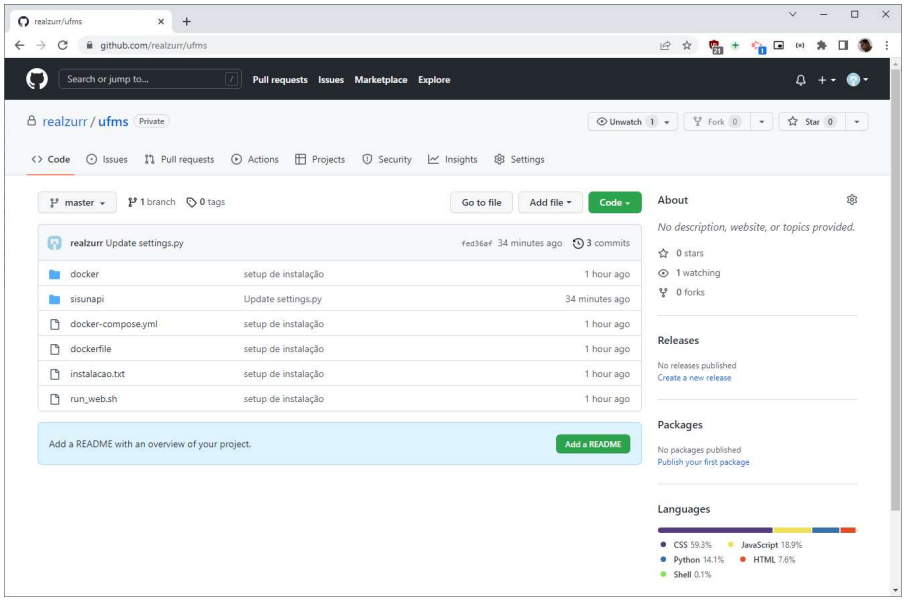
ROTEIRO DE INSTALAÇÃO

O SISUNAPI foi idealizado para facilitar sua implantação e desenvolvimento futuro. Com isso todo o seu núcleo inicial está presente no *Github*.

O *Github* é uma espécie de "rede social para programadores" (Ramos, 2021). O serviço é um site onde programadores compartilham seus desenvolvimentos e publicam, através da plataforma, seus trabalhos para a comunidade. O *Git* é um projeto aberto utilizado para manter o controle de alterações realizadas. O *Github* é um repositório online onde os códigos são armazenados para a distribuição.

O núcleo do SISUNAPI está presente no diretório online do desenvolver, no endereço git@github.com:realzurr/ufms.git (Figura 67). Realizando a clonagem do repositório é possível realizar o teste de todo o sistema inicialmente projetado.

Figura 67 GitHub do Projeto SISUNAPI

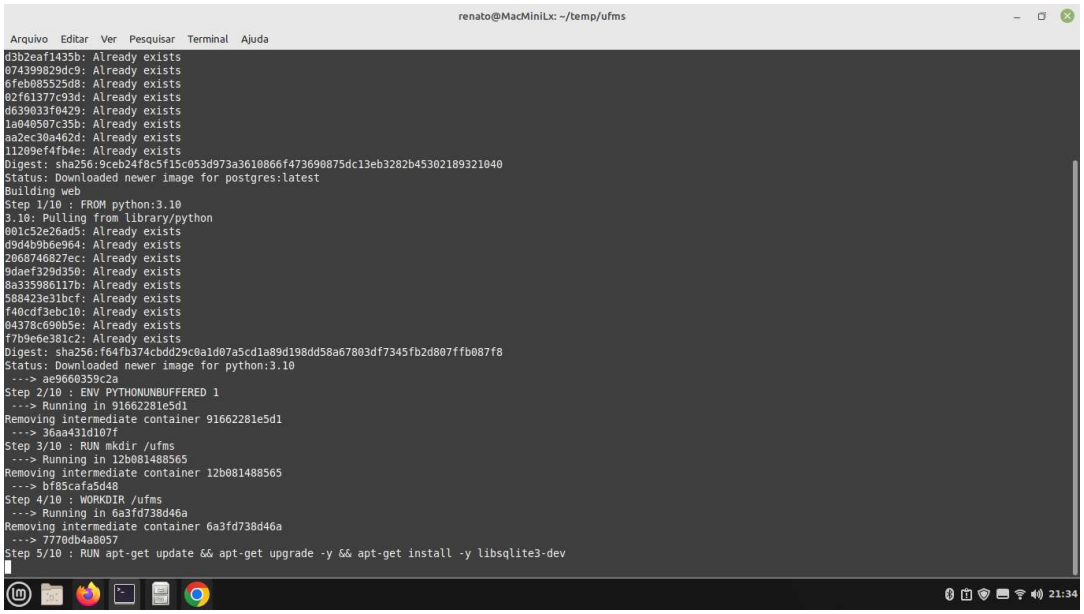


Fonte: Autor, 2022

Para realizar o teste é necessário que o sistema operacional da máquina tenha instalado o sistema *docker* e *docker-compose* para a correta instalação e execução do sistema (Figura 68).

Após a clonagem do repositório e o download de todo o seu conteúdo basta executar o comando, na pasta onde se encontram os arquivos, *docker-compose up -d*

Figura 68 Execução do Comando no Terminal

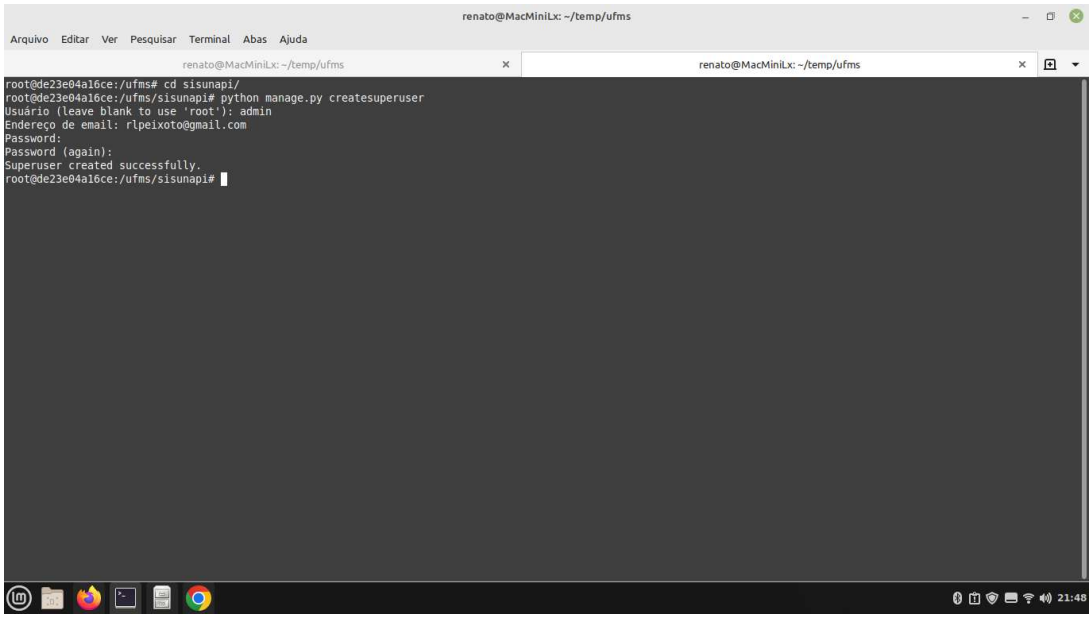


Fonte: Autor, 2022

No terminal do sistema operacional, após a conclusão da instalação, é necessário realizar a criação do *superusuário*.

Serão os únicos comandos necessários. Entre com o comando *docker-compose exec web bash*. Após entre no diretório do SISUNAPI com o comando *cd sisunapi*. Após execute o comando *python manager.py createsuperuser* e siga as orientações apresentadas na tela, entre com o usuário, o endereço de e-mail válido e uma senha válida. Após a inserção dos dados será apresentado uma mensagem de usuário criado com sucesso (Figura 69).

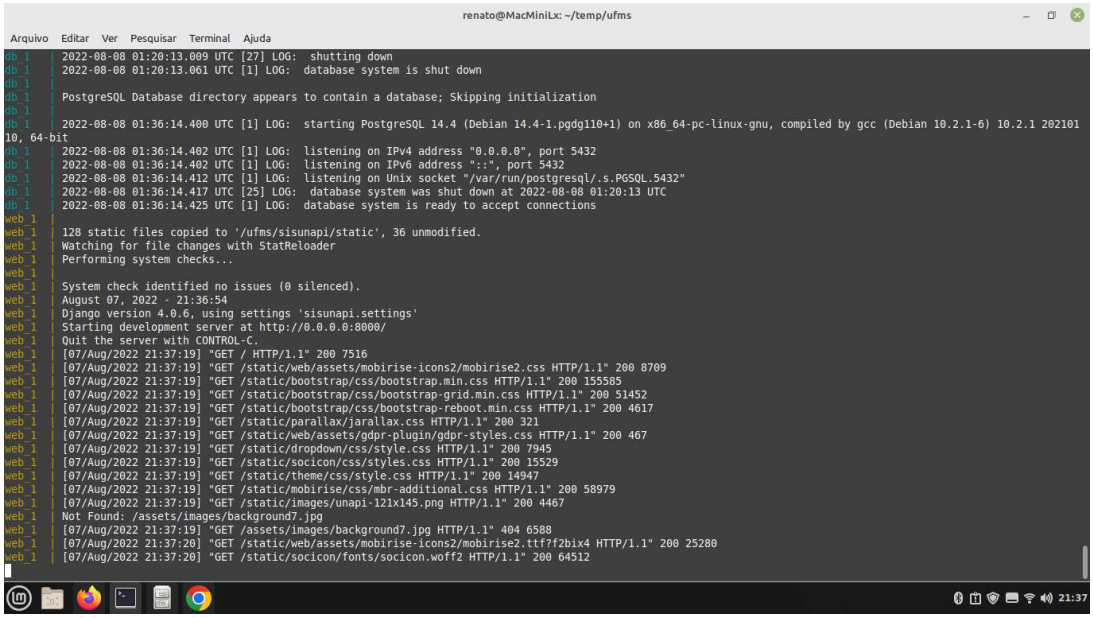
Figura 69 Criação de Super Usuário



Fonte: Autor, 2022

Para acesso a área administrativa do sistema, no navegador, entre com o endereço: *http://endereço-do-servidor:8000/admin* e será aberta a tela de login da Administração do Django (Figura 70) (O endereço do servidor é o endereço IP adotado pela máquina que está testando o ambiente. Caso seja a máquina local, deve-se ser substituído por *localhost*)

Figura 70 Terminal executando o SISUNAPI



Fonte: Autor, 2022