Disciplina: Lab. WEB I

Aluno: IAN DOS SANTOS SILVA

Docente: Eduardo Teles

WORKOUT

Camaçari

2025

SUMÁRIO

1. APRESENTAÇÃO

Este projeto visa apresentar um programa de gerenciamento de treinos para facilitar o contato e montagem dos treinos entre alunos e professores de educação física, permitindo um acompanhamento personalizado para cada aluno. O projeto pode ser acessado através do seguinte repositório:

MillyChurch/Workout: Programa para gerenciamento de treinos em uma academia

https://github.com/MillyChurch/Workout

2. TECNOLOGIAS UTILIZADAS

Foi utilizado a linguagem de programação python em conjunto com o framework django e o SGBD utilizado foi o mysql.

- ✓ Linguagens de estilo e marcação:
  - HTML: Linguagem de marcação (estrutura do conteúdo);
  - CSS: Linguagem de estilo (aparência do conteúdo).
- ✓ Linguagens de programação: Python, javascript
- ✓ Frameworks ou plataformas: Django e vscode.
- ✓ Gerência de Dados: MySQL 8.0.41.

3. PLATAFORMA DE IMPLEMENTAÇÃO

Hardware:

Processador: Ryzen 7 5700x

Memória Ram: 16gb ram

Software:

Sistema operacional: Windows 10 64 bits

SGBD: MYSQL 8.0.41

Servidor local: django development server

Linguagem de programação: Pyhon 3.11.15

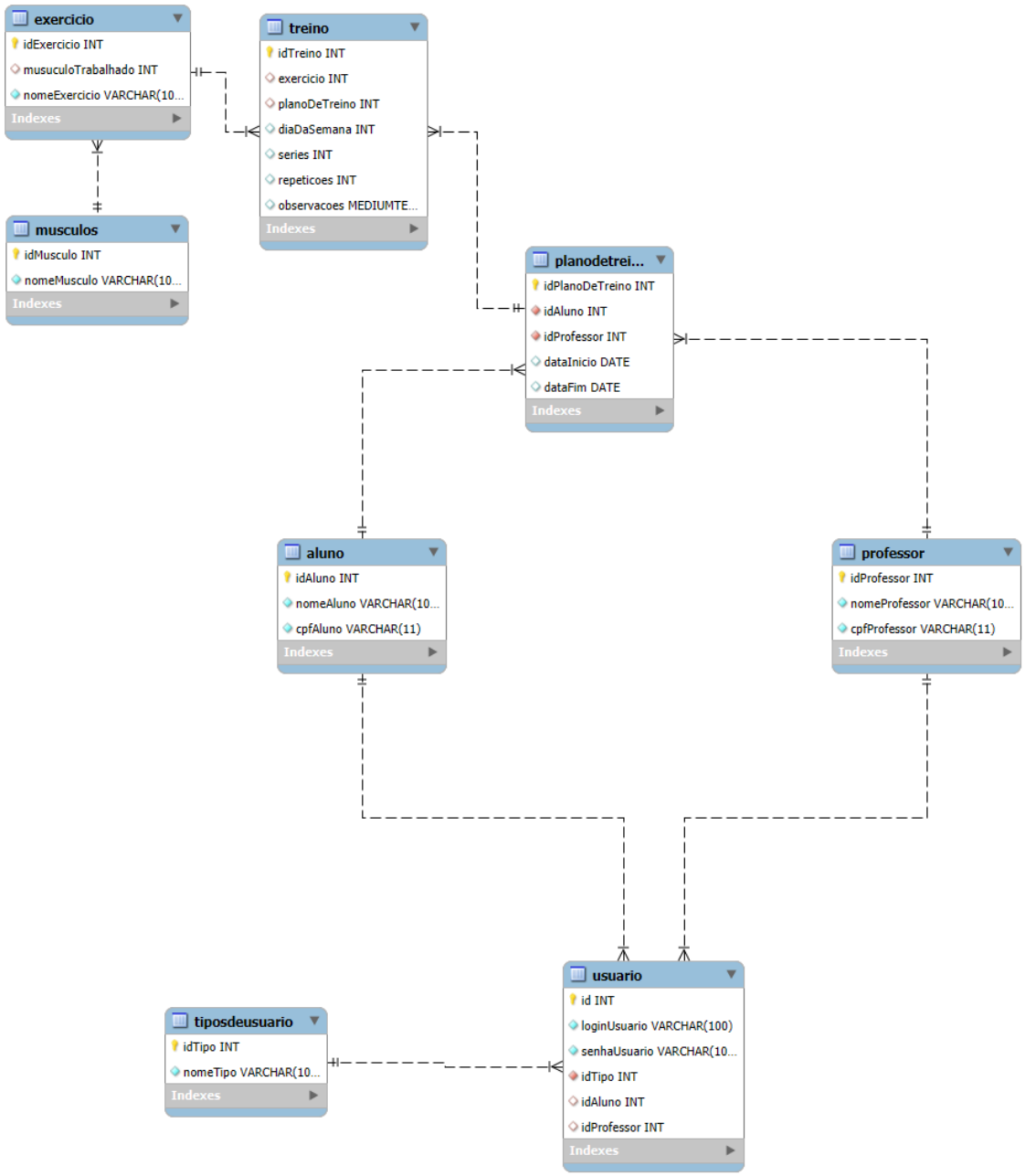
Framework: Django 5.2

Ferramentas de versionamento: GIT/GITHUB

4. SISTEMA

O banco de dados do sistema tem a seguinte organização dos dados:

Figura 1 – DER do Banco de Dados MySQL



Tipo de usuário: tabela que representa as permissões de cada usuário no sistema (ex: professor, aluno, administrador) associadas a um número

Usuário: É a tabela que representa as credenciais de entrada no sistema, e possui conexões com professor e aluno, podendo estes serem referenciados dependendo do tipo do usuário

Aluno: Representa os alunos, com nome, cpf e id (sendo possível adicionar novas informações no futuro)

Professor: Representa os professores, com nome, cpf e id (sendo possível adicionar novas informações no futuro)

Plano de treino: Responsável por conectar alunos aos seus professores responsáveis, é uma tabela de associcação.

Músculo: Responsável por armazenar os músculos do corpo humano

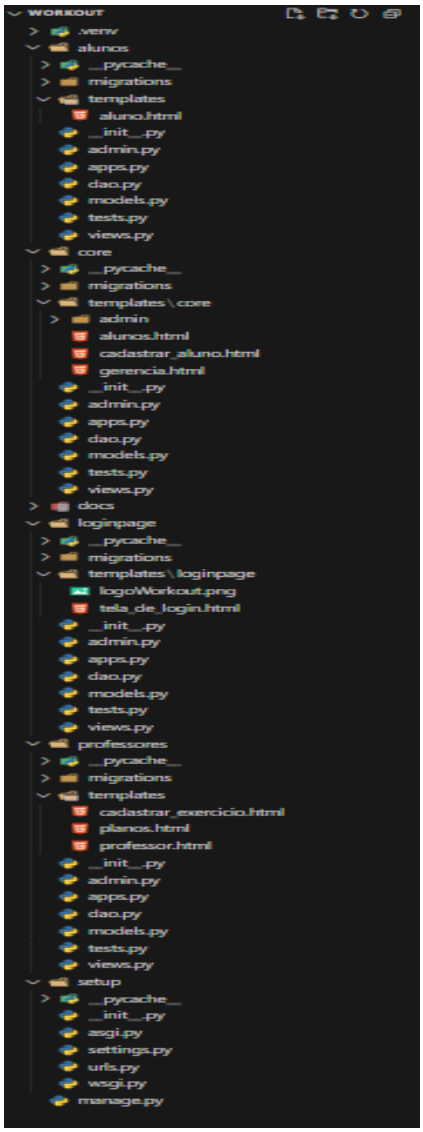
Exercicio: Responsável por guardar a informação de tipos específicos de exercício, que está associado a um músculo principal (ex: supino com halteres, cadeira adutora, afundo...).

Treino: Representa a execução de um exercício, com quantidade de séries, repetições, dia da semana do treino e observações do professor para o aluno, o trenio estará inserido no plano de treino de algum aluno

4.1 CAMADAS E PADRÃO DO PROJETO

Nesse projeto foi utilziado o padrão MTV (model, template, view), para uma abordagem mais modular da aplicação. Abaixo, apresentamos a Figura 2, com a árvore do projeto.

Figura 2 – Projeto implementado na IDE Eclipse



A seguir, apresentamos um descritivo da composição das camadas.

4.1.1 Model (Modelo)

A camada Model está representada no projeto nos arquivos:

- Professores/models.py
- Alunos/models.py
- Core/models.py
- Loginpage/models.py

O arquivo *Professores/models.py* é composto pelas seguintes classes:

Professor: Classe que representa um professor, com nome e cpf

O arquivo *Aluno/models.py* é composto pelas seguintes classes:

Aluno: Classe que representa um aluno, com nome e cpf

O arquivo *Core/models.py* é composto pelas seguintes classes:

Plano de Treino: Classe que associa um professor à um aluno, referencia um objeto de cada associado.

Musculo: Classe que contem o nome do musculo.

Exercicio: Classe que contem o nome do exercicio e referencia um objeto Musculo.

Treino: Classe que contem a quantidade de séries e repetições, observações e dia da semana para um objeto Exercício, essa classe também referencia um plano de treino a qual aquele treino pertence.

O arquivo *Loginpage/models.py* é composto pelas seguintes classes:

Tipo de usuário: Classe que descreve o tipo de um usuário com o nome do tipo

Usuario: classe que contem credenciais de um usuário, com login, senha, uma referencia a um tipo de usuário e uma possível referência a um professor e um aluno

4.1.2 Templates (Visão)

A camada de Templates, equivalente a camada "View" no modelo MVC, está representada no projeto nos arquivos:

Professores

- Professores/templates/cadastrar_exercicio.html
- Professores/templates/planos.html
- Professores/templates/professor.html

Alunos

- Alunos/templates/ aluno.html

Core

- Core/templates/alunos.html
- Core/templates/cadastrar_aluno.html
- Core/templates/gerencia.html
- Core/templates/admin/cadastrar_professor.html
- Core/templates/admin/profs.html

Loginpage

- Loginpage/templates/tela_de_login.html

Esta camada é responsável pela visualização final do usuário, o que ele verá depois de cada processamento de dados.

4.1.3 Views (Controlador)

A camada Views, equivalente ao “controller” no modelo MVC, está representada no projeto nos seguintes arquivos, com os seguintes métodos:

- **Core/views.py**
  - admin_screen():
    - Gerencia o acesso e as ações da interface administrativa, permitindo que usuários autenticados dos tipos administradores ou recepcionistas executem funções como cadastrar ou deletar professores e aluno, e editar planos de treino. Com base na URL e no tipo de requisição (GET ou POST), ela direciona para as páginas corretas ou executa as funções correspondentes.
  - profs()
    - Caso o usuário seja do tipo administrador, Retorna core/admin/profs.html com a lista de professores.
  - alunos()
    - Caso o usuário seja do tipo administrador ou recepcionista, Retorna core /alunos.html com a lista de alunos.
  - verifica_se_logado()
    - Método para verificar se o usuário está logado, utilizado em todos os views que possuem proteção por autenticação.
  - novo_professor()
    - Registra um novo professor no banco de dados. Obtém os dados do formulário, valida os campos e chama os métodos do pacote dao, que registram o professor e o usuario do aluno.
  - del_professor()
    - chama o método de deletar professor no DAO
  - novo_aluno()
    - Registra um novo aluno banco de dados. Obtém os dados do formulário, valida os campos e chama os métodos do pacote dao, que registram o aluno, usuario do aluno e plano de treino
  - del_aluno()
    - Recebe o número do ID de um objeto Professor e chama o método de deletar professor do pacote DAO
  - editar_plano()
    - Chama o DAO para alterar um professor em um plano de treino específcio

- **Alunos/ views.py**
  - alunoPage(request)
    - Recebe as informações de um aluno e envia os treinos daquele aluno para o template aluno.html.

- **Professores/ views.py**
  - professorPage()
    - Renderiza a página principal do professor, caso ele esteja logado e seja do tipo correto. Procura os dados do professor e os envia para o template professor.html.

    - cadastrar_exercicio_page()
        - Exibe a página para o professor cadastrar um novo exercício. Caso receba um POST, extrai os dados do formulário (nome do exercício e grupo muscular) e chama a função registrar_exercicio. Sempre envia a lista de músculos disponíveis para o template cadastrar_exercicio.html.
    - plano_de_treino_page()
        - Exibe a página de gerenciamento dos planos de treino dos alunos vinculados ao professor. Se a requisição for POST, verifica se se trata da criação, edição ou exclusão de um treino com base na URL. Também permite filtrar os treinos de um aluno em um dia específico, enviando os dados para o template planos.html.
    - novo_treino()
        - Registra um novo treino no plano de um aluno. Obtém os dados do formulário (aluno, exercício, série, repetições, observações e dia da semana), valida os campos e chama registrar_treino, método do pacote dao, que registra um treino no banco de dados, com os dados apropriados.
    - editar_treino()
        - Edita um treino existente, recebe os dados de um formulário do usuário. Atualiza os dados do treino com base nas informações do formulário, validando todos os campos.
    - deletar_treino
        - Exclui um treino com base no ID recebido do formulário.

- Loginpage/ views.py
    - Login_screen()
        - Esse método gerencia a lógica de login dos usuários da aplicação. Quando a requisição é GET, ele apenas renderiza a tela de login (loginpage/tela_de_login.html). Quando é POST, o sistema tenta autenticar o usuário com base nas credenciais informadas, redirecionando-o para sua página de usuário ou informando que as credenciais informadas estão incorretas.

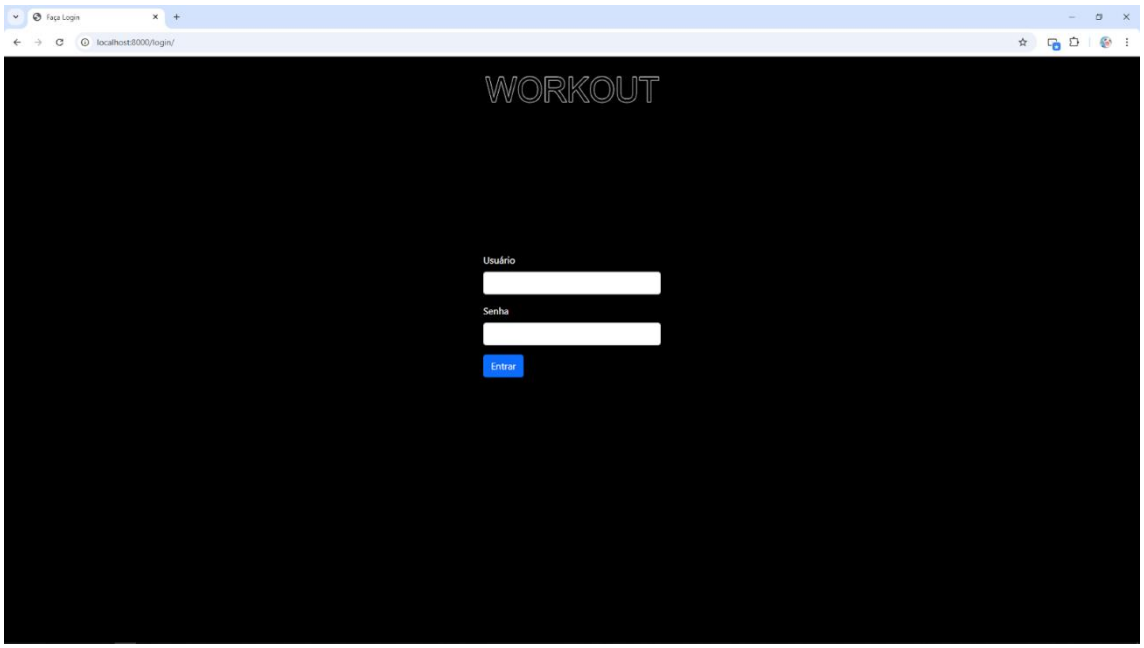
- Setups/ urls.py: Arquivo contendo todas as urls do sistemas e para qual views (controller) o sistema deverá direcionar a requisição.

Essa camada é responsável por fazer a “ponte” entre os dados (models) e sua exibição (templates), decidindo quais informações devem ser mostradas ao usuário quando uma requisição é feita.

4.2 TELAS

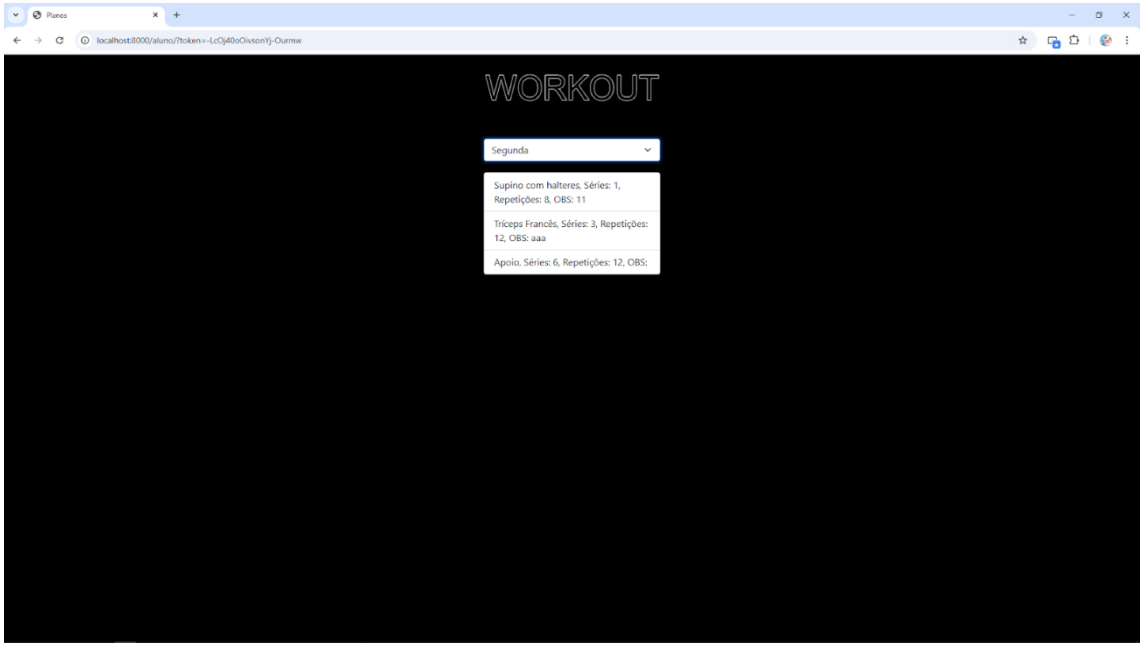
Este tópico apresenta o sistema XXXXXXX. As telas abaixo são apresentadas por ordem de usabilidade no sistema e são explicadas de forma detalhada após cada imagem.

Figura 3 – Tela de Login



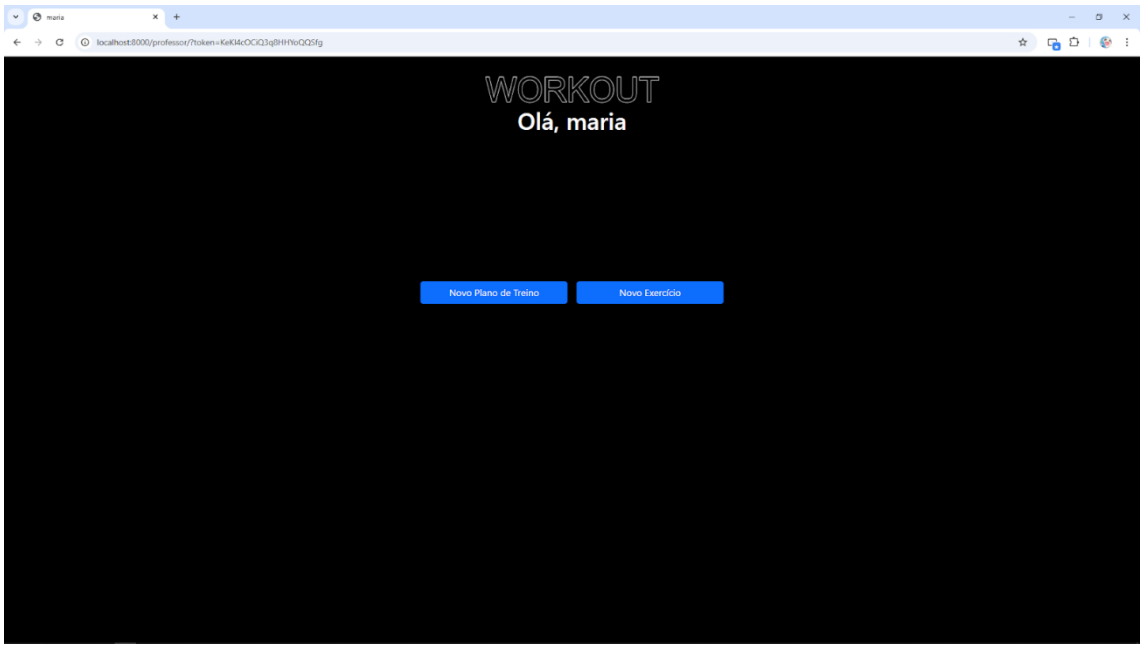
Esta tela tem como objetivo permitir ao usuário acessar a página referente a sua função no sistema, permitindo utilizar apenas a parte que compete a ele, garantindo segurança das informações

Figura 4 – Tela Principal - Aluno



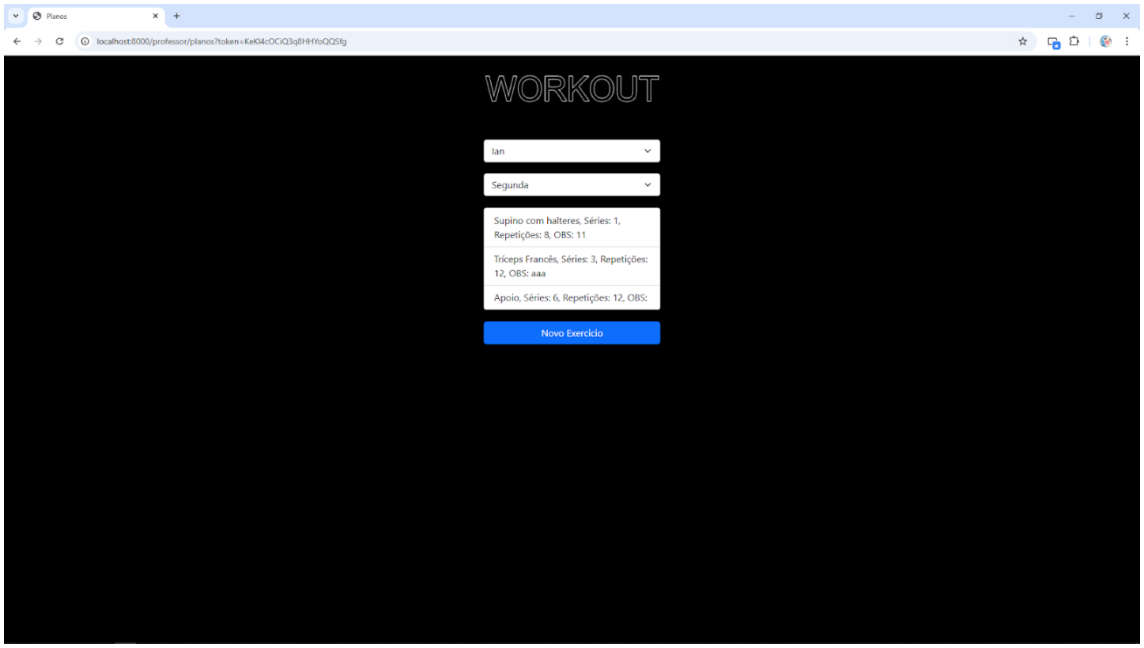
Esta tela é a tela inicial do usuário aluno, nela o aluno poderá escolher um dia da semana e fisualizar seus treinos

Figura 5 – Tela Principal – Professor



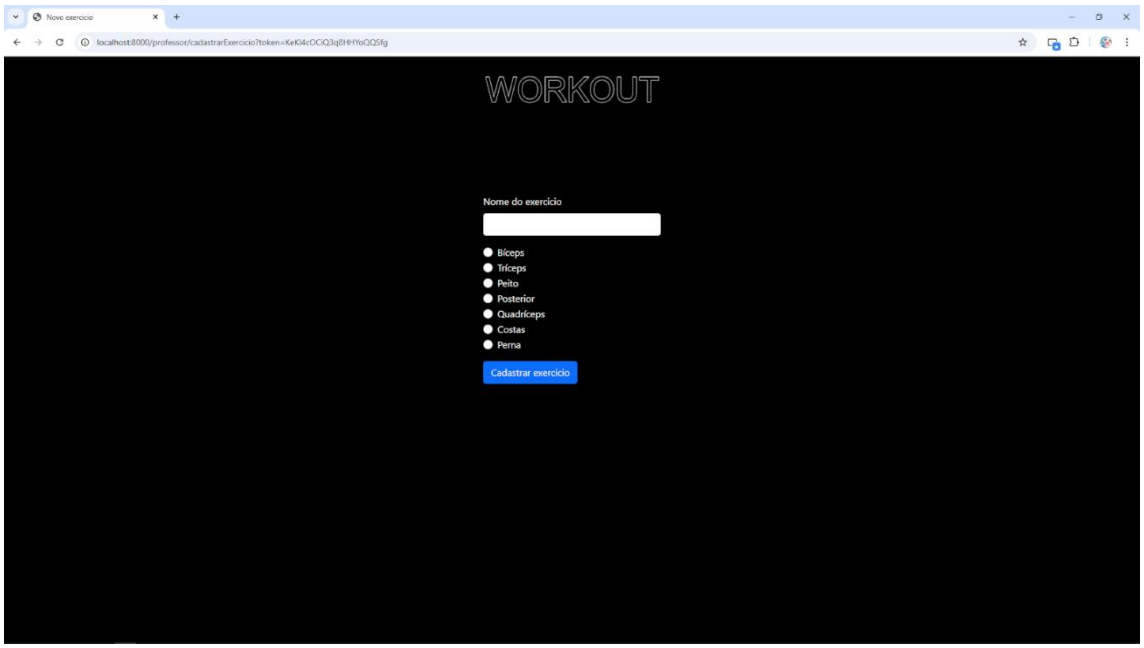
Esta tela é a tela inicial do professor, onde o professor poderá escolher criar um novo exercício ou gerencial o plano de treino de um aluno

Figura 6 – Tela De gerenciar plano de treino - Professor



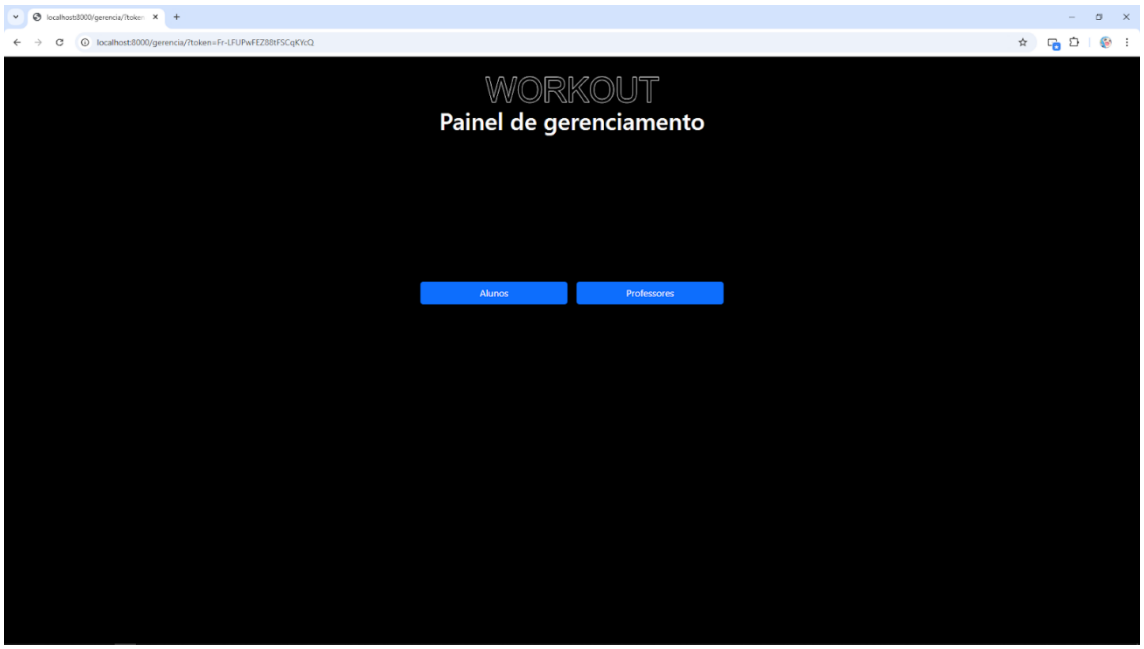
Esta é a tela de gerênciar plano de treino de um professor, onde ele poderá visualizar os treinos de um aluno, editar, deletar e criar novos exercícios para aquele aluno nos dias selecionados

Figura 7 – Tela de adicionar novo exercício



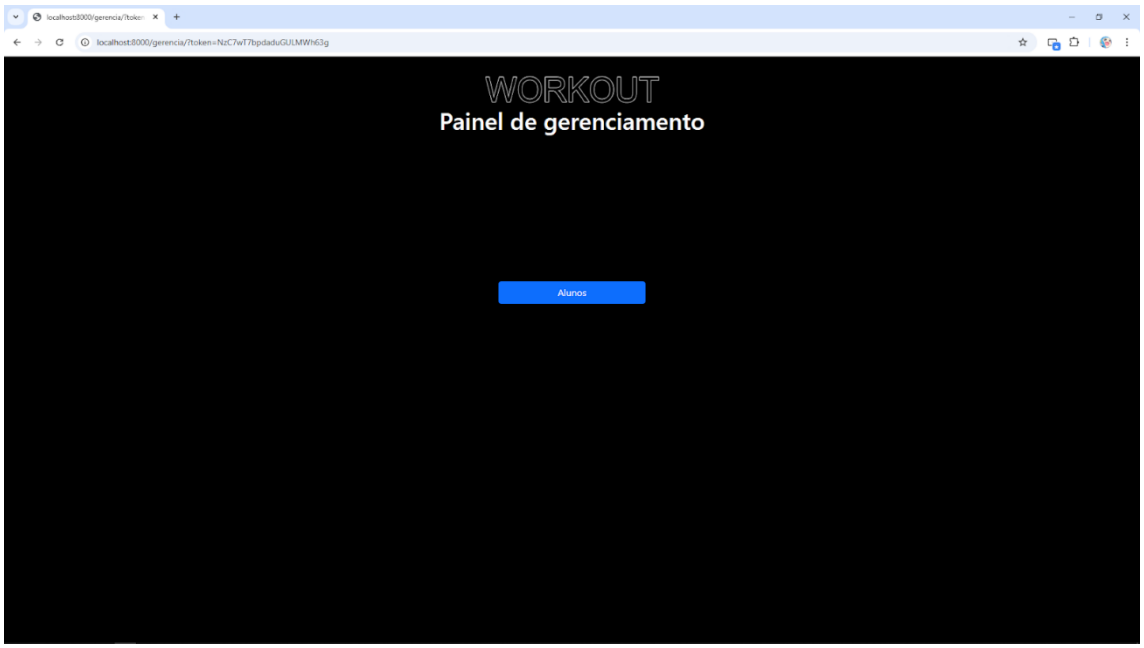
Esta tela permite o professor cadastrar um novo exercício no sistema

Figura 8 – Tela Principal - Administrador



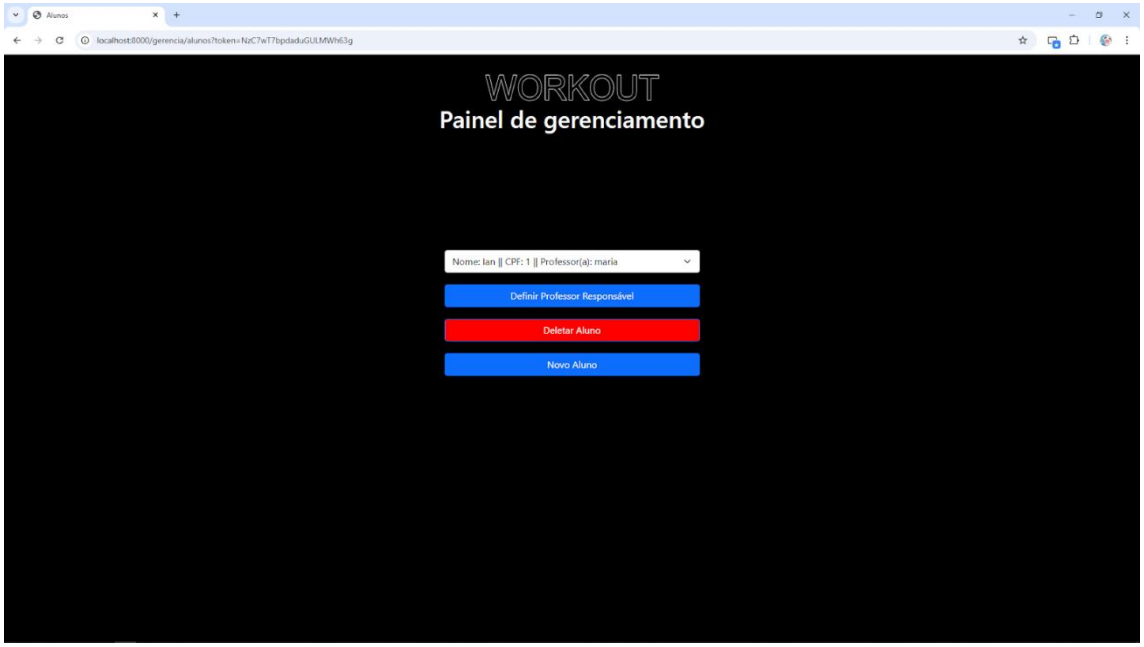
Esta tela é a tela inicial do administrador, onde o adm poderá escolher gerenciar alunos ou professores

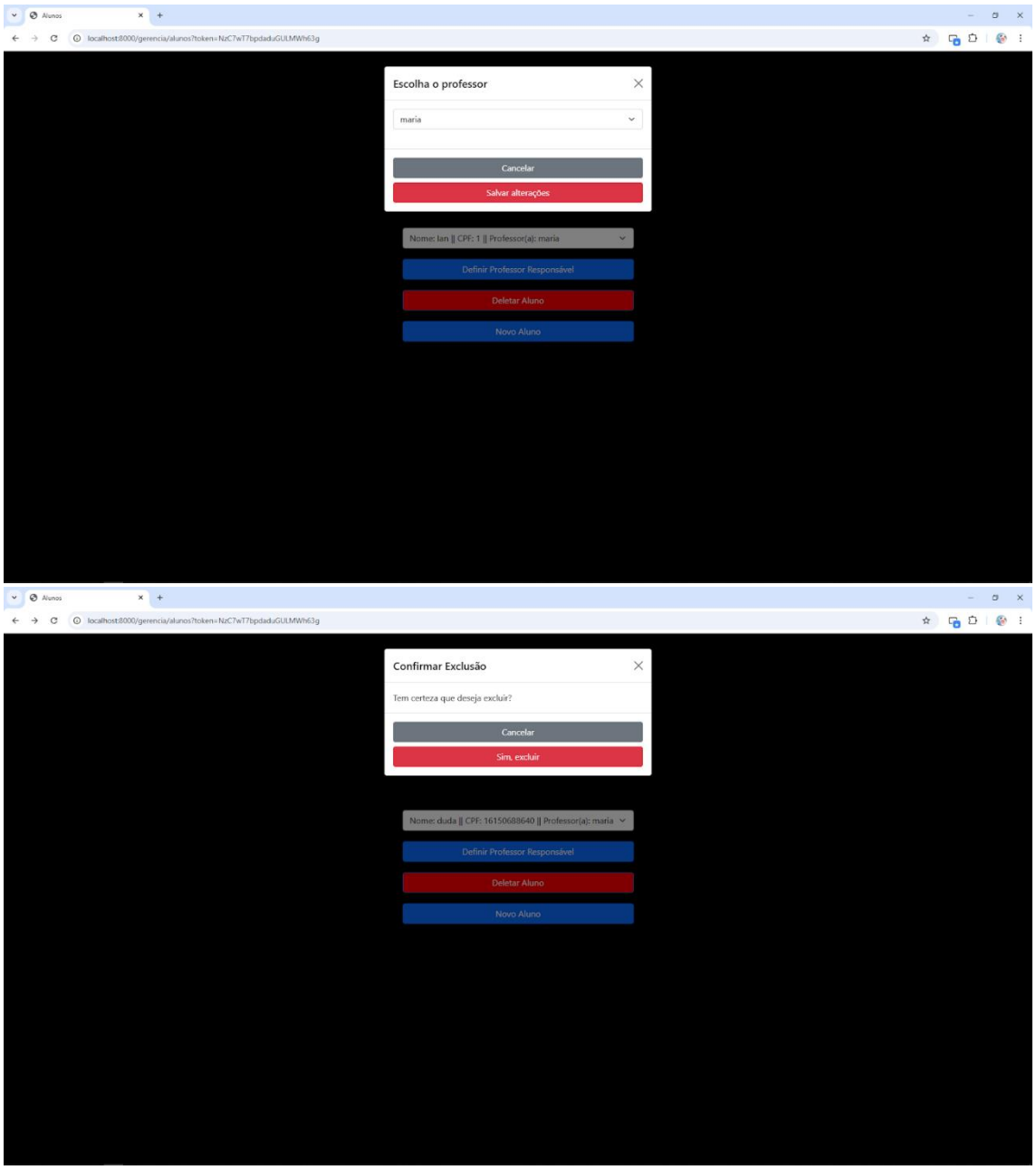
Figura 9 – Tela Principal - Recepcionista



Esta tela é a tela inicial do recepcionista, onde o mesmo poderá escolher gerenciar alunos ou professores

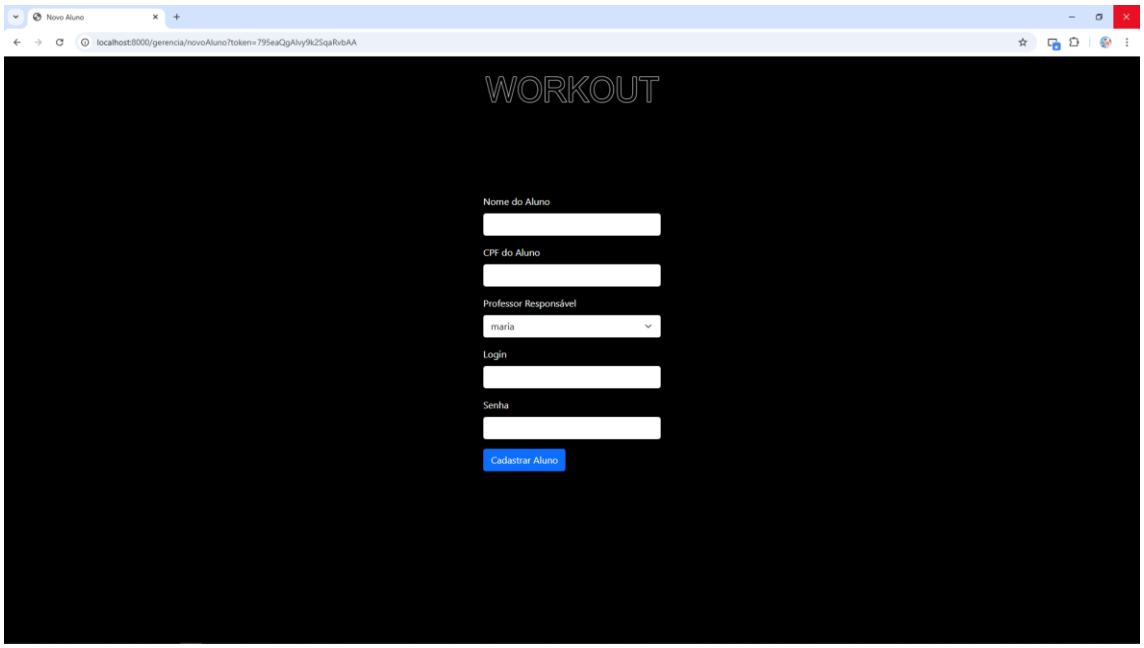
Figura 10 – Tela de gerenciamento de alunos (ADM ou Recepcionista)





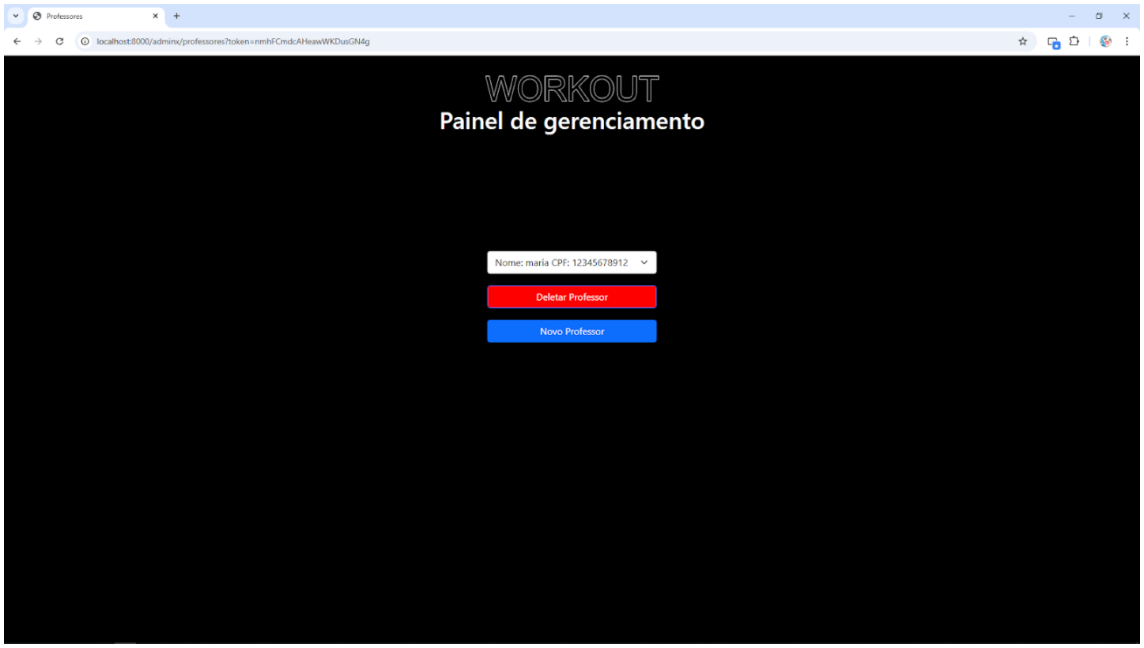
Esta tela é onde o administrador e recepcionista poderão gerenciar o cadastro de alunos e seus vínculos com professores

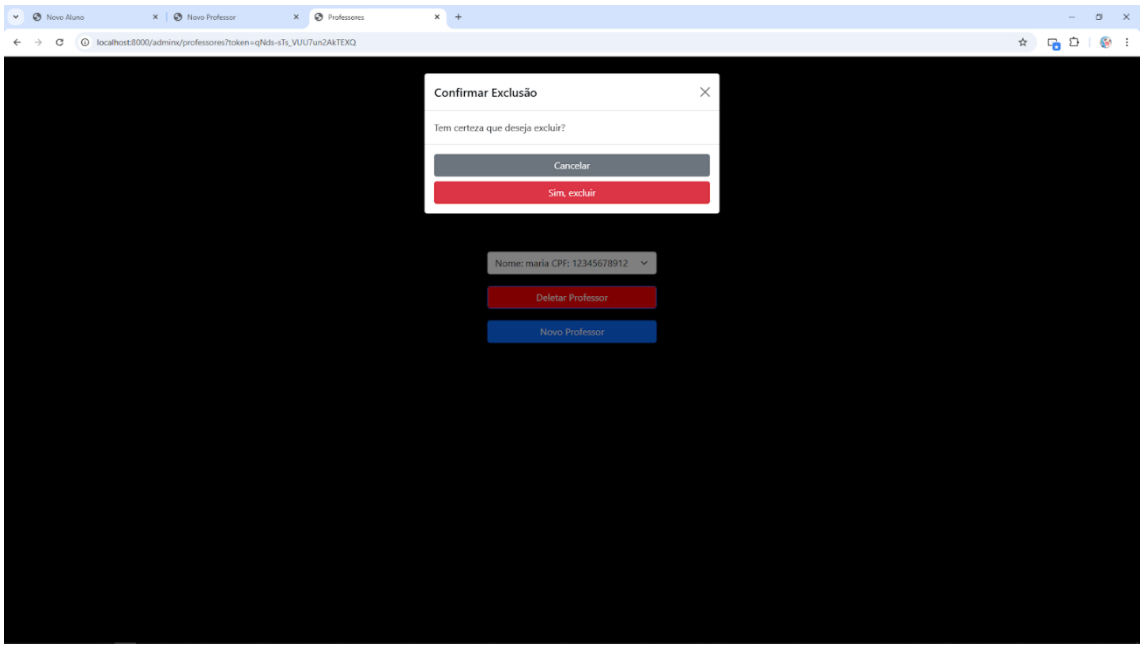
Figura 11 – Tela de adicionar novo aluno - (ADM ou Recepcionista)



Esta tela é onde o usuário (administrador ou recepcionista) poderá adicionar novos alunos

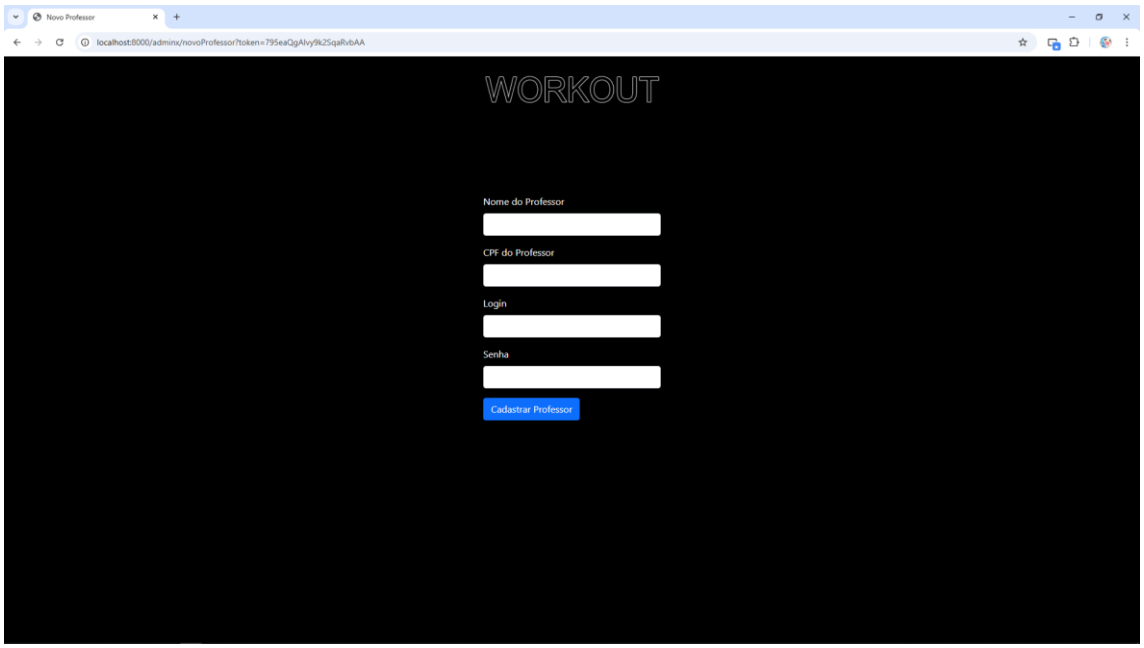
Figura 12 – Tela de gerenciamento de professores - ADM





Esta tela é onde o usuário (administrador) poderá gerenciar o cadastro de professores

Figura 13 – Tela de adicionar novo professor - ADM



Esta tela é onde o usuário (administrador) poderá adicionar novos professores