Departamento de Telemática

Prof. Antônio de Barros Serra

Projeto Final - Sistemas Web – 2025.1

Graduação - Tecnologia em Telemática

Aluna: Carla Beatriz da Silva Teixeira (individual)

Projeto Acadêmico e Objetivo

Projeto executado e testado em Visual Studio Code e salvo no GitHub. Abaixo os passos para abrir e localizar os arquivos do Projeto Final.

1) Elabore um banco de dados que permita aos Departamentos de Ensino de uma Universidade a:
a. Cadastrar todos os professores do departamento (matrícula, nome, email, telefone);
b. Cadastrar todas as disciplinas ministradas no departamento (código-disciplina, disciplina, carga-horária);
c. Registrar que disciplinas um professor é apto a ministrar.
d. Registrar todas as turmas nas quais ele já ministrou a disciplina (código-professor, código-turma, código-disciplina, semestre, número-alunos, horário: manhã, tarde ou noite);

2) Elaborar um Sistema WEB que tenha as seguintes funcionalidades:
a. Cadastrar (incluir, alterar, excluir, consultar) professores;
b. Cadastrar (incluir, alterar, excluir, consultar) disciplinas;
c. Registrar disciplinas que podem se ministradas por um professor ;
d. Registrar turmas nas quais um professor já ministrou uma determinada disciplina;
e. Elaborar uma consulta onde seja escolhida uma das disciplinas ministradas no departamento e que mostre todos os professores aptos a ministrar aquela disciplina;
f. Elaborar uma consulta onde seja escolhido um professor e que mostre todas as disciplinas que ele já ministrou, com a carga horária total dele ministrando essa disciplina e a quantidade total de alunos que participaram das turmas das disciplinas.

Rodando localmente

Clone o projeto

```
git clone https://github.com/Carla-beatriz22/SistemasWeb_2025.git
```

Entre no diretório do projeto

```
cd SistemasWeb_2025
```

Crie uma variavel de ambiente

```
python -m venv .venv
```

Instale as dependências

```
pip install -r requirements.txt
```

Configure .env
Inicie o servidor
python manage.py runserver

Código – Visual Studio Code

```
from django.db import models
from django.contrib.auth.models import User

class Notas(models.Model):
    matricula= models.ForeignKey("Matricula", models.CASCADE, related_name="matricula", blank=False, null=False)
    avaliacao= models.CharField(max_length=50, verbose_name="Avaliacao")
    nota= models.DecimalField(max_digits=5, verbose_name="Nota",decimal_places=2)
    data_avaliacao= models.DateField(verbose_name="Data avaliacao", default=None, blank=True, null=True)
    created_at= models.DateTimeField(auto_now_add=True, verbose_name="Criado em")
    updated_at= models.DateTimeField(auto_now=True, verbose_name="Atualizado em")

    class Meta:
        db_table="notas"
        verbose_name="notas"

    def __str__(self):
        return self.avaliacao


class Matricula(models.Model):
    turma= models.ForeignKey("Turma", models.CASCADE, related_name="turma_matricula", blank=False, null=False)
    aluno= models.ForeignKey("Aluno", models.CASCADE, related_name="aluno_matricula", blank=False, null=False)
    data_matricula= models.DateField(verbose_name="Data de Matricula", default=None, blank=True, null=True)
    created_at= models.DateTimeField(auto_now_add=True, verbose_name="Criado em")
    updated_at= models.DateTimeField(auto_now=True, verbose_name="Atualizado em")

    class Meta:
        db_table="matriculas"
        verbose_name="matriculas"


class Turma(models.Model):
    nome= models.CharField(max_length=50)
    professor= models.ForeignKey("Professor", models.CASCADE, related_name="professores_turma", blank=False, null=False)
    disciplina= models.ForeignKey("Disciplina", models.CASCADE, related_name="disciplina_turma", blank=False, null=False)
    data_inicio= models.DateField(verbose_name="Data de Matricula", default=None, blank=True, null=True)
```

```
    data_fim= models.DateField(verbose_name="Data de Matricula", default=None, blank=True, null=True)
    carga_horaria= models.IntegerField(verbose_name="Carga horario", blank=True, null=True)
    created_at= models.DateTimeField(auto_now_add=True, verbose_name="Criado em")
    updated_at= models.DateTimeField(auto_now=True, verbose_name="Atualizado em")

    class Meta:
        db_table="turmas"
        verbose_name="turmas"

    def __str__(self):
        return self.nome


class Aluno(models.Model):
    instituicao= models.ForeignKey("Instituicao", models.CASCADE, related_name="instituicao_aluno",
blank=False, null=False)
    user= models.ForeignKey(User, on_delete=models.PROTECT, null=True, verbose_name="Aluno")
    cpf= models.CharField(max_length=11, verbose_name="Cpf")
    email= models.CharField(max_length=40, verbose_name="Email")
    telefone= models.CharField(max_length=15, verbose_name="Telefone")
    data_nacimento= models.DateField(verbose_name="Data de Nascimento", default=None, blank=True,
null=True)
    created_at= models.DateTimeField(auto_now_add=True, verbose_name="Criado em")
    updated_at= models.DateTimeField(auto_now=True, verbose_name="Atualizado em")

    class Meta:
        db_table="alunos"
        verbose_name="alunos"

    def __str__(self):
        return self.user.first_name


class Instituicao(models.Model):
    nome= models.CharField(max_length=40, verbose_name="Nome")
    endereco= models.CharField(max_length=11, verbose_name="Endereco")
    telefone= models.CharField(max_length=15, verbose_name="Telefone")
    cnpj= models.CharField(max_length=14, verbose_name="Cnpj")

    class Meta:
        db_table="instituicao"
        verbose_name="instituicao"

    def __str__(self):
        return self.nome


class Professor(models.Model):
    instituicao= models.ForeignKey(Instituicao, models.CASCADE, related_name="instituicao_professor",
blank=False, null=False)
```

```
    user= models.ForeignKey(User, on_delete=models.PROTECT, null=True, verbose_name="Professor")
    cpf= models.CharField(max_length=11, verbose_name="name")
    especializacao= models.CharField(max_length=40, verbose_name="email")

    class Meta:
        db_table="professores"
        verbose_name="professores"

    def __str__(self):
        return self.user.first_name


class ProfessorDisciplina(models.Model):
    professor= models.ForeignKey(Professor, models.CASCADE, related_name="professor_disciplina",
blank=False, null=False)
    disciplina= models.ForeignKey("Disciplina", models.CASCADE, related_name="disciplina_disciplina",
blank=False, null=False)
    created_at= models.DateTimeField(auto_now_add=True, verbose_name="Ministrou em")
    updated_at= models.DateTimeField(auto_now=True, verbose_name="Disponivel em")

    class Meta:
        db_table="disciplina"
        verbose_name="disciplina"

    def __str__(self):
        return self.professor.user.first_name


class Disciplina(models.Model):
    nome= models.CharField(max_length=30, verbose_name="Disciplina")
    created_at= models.DateTimeField(auto_now_add=True, verbose_name="Criado em")
    updated_at= models.DateTimeField(auto_now=True, verbose_name="Atualizado em")

    class Meta:
        db_table="disciplina"
        verbose_name="disciplina"

    def __str__(self):
        return self.nome

class Historico(models.Model):
    TURNO_CHOICES = [
        ('Manha', 'Manha'),
        ('Tarde', 'Tarde'),
        ('Noite', 'Noite'),
    ]

    professor = models.ForeignKey(Professor, models.CASCADE, related_name="professores_historico",
blank=False, null=False)
    disciplina = models.ForeignKey(Disciplina, models.CASCADE, related_name="disciplina_historico",
```

```
blank=False, null=False)
    turma = models.ForeignKey(Turma, models.CASCADE, related_name="turma_historico", blank=False,
null=False)
    semestre = models.CharField(max_length=10, verbose_name="Semestre")  # antes era FloatField
    horario = models.CharField(max_length=10, choices=TURNO_CHOICES, verbose_name="Horario")  #
antes era CharField comum
    carga_horaria = models.IntegerField(verbose_name="Carga horaria", blank=True, null=True)
    quantidade_aluno = models.IntegerField(verbose_name="Quantidade de alunos")
    created_at = models.DateTimeField(auto_now_add=True, verbose_name="Criado em")
    updated_at = models.DateTimeField(auto_now=True, verbose_name="Atualizado em")

    class Meta:
        db_table = "historico"
        verbose_name = "historico"

    def __str__(self):
        return f"{self.professor.user.first_name} - {self.disciplina.nome} - {self.semestre}"
```

Tecnologias usadas

- Django (framework backend);
- Python (linguagem principal);
- HTML5, CSS3, JavaScript (frontend).

Funcionalidades principais

- CRUD de Professores;
- CRUD de Alunos;
- CRUD de Disciplinas;
- CRUD de Turmas;
- Registro de Matrículas;
- Registro de Notas;
- Consulta de Disciplinas por Professor;
- Consulta de Professores por Disciplina;
- Histórico de Disciplinas Ministradas.

Vantagens deste projeto

- Para se criar uma interface no próprio sistema (formulário HTML acessível via frontend) para cadastrar professores, em vez de depender apenas da interface de administração;

- No painel admin, é necessário garantir que o “Professor” esteja corretamente registrado no admin.py com os campos essenciais;

O que pode ser melhorado a partir do que está sendo entregue no projeto

- Implementar busca/filtragem nas listagens;
- Adicionar CSS/Bootstrap para visual mais profissional;
- Implementar autenticação (login obrigatório);
- Implementar sistema de validação de name, e-mail e telefone dos professores nos formulários.