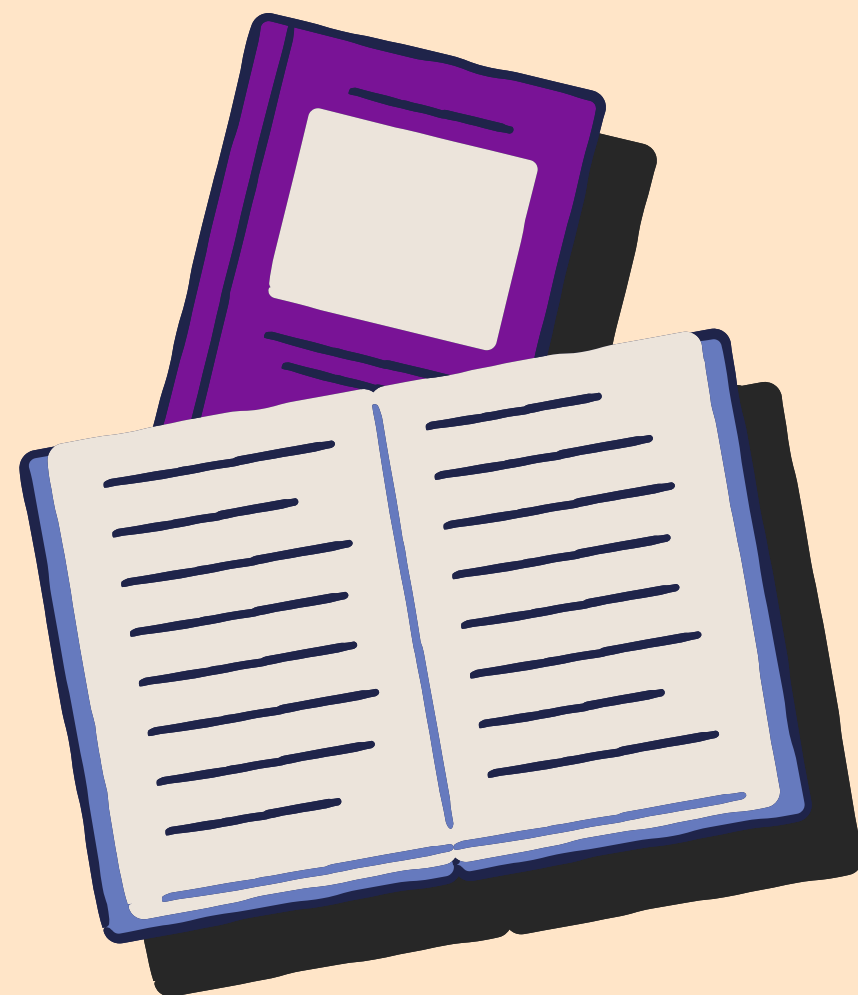




EduCollab

**Maria Eduarda Romana
Nicolas Romano
Rafael Sanches**



Objetivo

EduCollab é uma plataforma web desenvolvida para conectar professores e alunos em um ambiente de aprendizado colaborativo.

O objetivo do projeto é fornecer ferramentas para que professores possam criar e gerenciar conteúdo educacional, como bancos de questões e provas, enquanto os alunos podem utilizar esses recursos para estudar e testar seus conhecimentos.

Modelagem do Banco de Dados

```
models.py X
models.py > ...
1  from typing import Optional, List
2  from datetime import datetime
3  from sqlmodel import SQLModel, Field, Relationship, Column, JSON
4  from enum import Enum
5
6  # --- ENUMS ---
7  class AnswerOptions(str, Enum):
8      A = "A"
9      B = "B"
10     C = "C"
11     D = "D"
12
13 # --- USUÁRIOS ---
14 class UserBase(SQLModel):
15     name: str
16     phone: str
17     area: Optional[str] = None
18     level: Optional[str] = None
19
20 class User(UserBase, table=True):
21     # 'email' é chave primária lógica ou unique index
22     email: str = Field(primary_key=True, index=True)
23     hashed_password: str
24     user_type: str # 'student' ou 'teacher'
25
26 # --- QUESTÕES ---
27 class QuestionBase(SQLModel):
28     materia: str
29     enunciado: str
30     # SQLite não tem array nativo, usamos JSON type do SQLAlchemy
31     alternativas: List[str] = Field(sa_column=Column(JSON))
32     resposta: AnswerOptions
```

Integração back-BD

Autenticação e Segurança:

Persistência de usuários e verificação segura de credenciais via Banco de Dados. Busca otimizada com select, validação de hash de senha (Bcrypt) e geração de token JWT.

Arquivo: security.py

```
@router.post("/token", response_model=Token)
async def login_for_access_token(
    form_data: OAuth2PasswordRequestForm = Depends(),
    session: Session = Depends(get_session)
):
    # Busca usuário no banco
    statement = select(User).where(User.email == form_data.username)
    user = session.exec(statement).first()

    if not user or not verify_password(form_data.password, user.hashed_password):
        raise HTTPException(
            status_code=401,
            detail="Email ou senha incorretos",
            headers={"WWW-Authenticate": "Bearer"},
        )

    access_token = create_access_token(
        data={"sub": user.email, "user_type": user.user_type},
        expires_delta=timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES)
    )

    return {
        "access_token": access_token,
        "token_type": "bearer",
        "user_type": user.user_type
    }
```

Integração back-BD

Relacionamentos no Fórum:

Gerenciamento de chaves estrangeiras (Foreign Key) e integridade referencial entre Tópicos e Respostas. Criar uma resposta vinculada a um tópico existente e recuperar esses dados aninhados.

Arquivo: forum.py

```
@router.post("/topics/{topic_id}/replies", response_model=ReplyDisplay)
async def add_reply_to_topic(
    topic_id: str,
    reply_data: ReplyBase,
    current_user: users.User = Depends(users.get_current_user),
    session: Session = Depends(get_session)
):
    context = 'teacher' if current_user.user_type == 'teacher' else 'student'
    statement = select(Topic).where(Topic.id == topic_id, Topic.context_type == context)
    topic = session.exec(statement).first()

    if not topic:
        raise HTTPException(status_code=404, detail="Tópico não encontrado")

    autor_nome = current_user.name

    new_reply = Reply(
        id=f"resp-{int(datetime.now().timestamp())}",
        topic_id=topic_id,
        texto=reply_data.texto,
        autor_nome=autor_nome,
        data=datetime.now()
    )

    session.add(new_reply)
    session.commit()
```


Integração back-BD

Persistência de Estruturas Complexas:

Armazenamento de listas em um banco relacional que espera tipos primitivos. Uso de tipos avançados do SQLAlchemy (JSON) dentro do SQLAlchemy para permitir que uma coluna armazene uma lista ["A", "B", "C"].

Arquivo: questions.py

```
@router.post("/question", response_model=QuestionDisplay, status_code=status.HTTP_201_CREATED)
async def create_question(
    question_data: QuestionCreate,
    current_user: users.User = Depends(users.get_current_user),
    session: Session = Depends(get_session)
):
    if current_user.user_type != 'teacher':
        raise HTTPException(status_code=404, detail="Apenas professores podem criar perguntas")

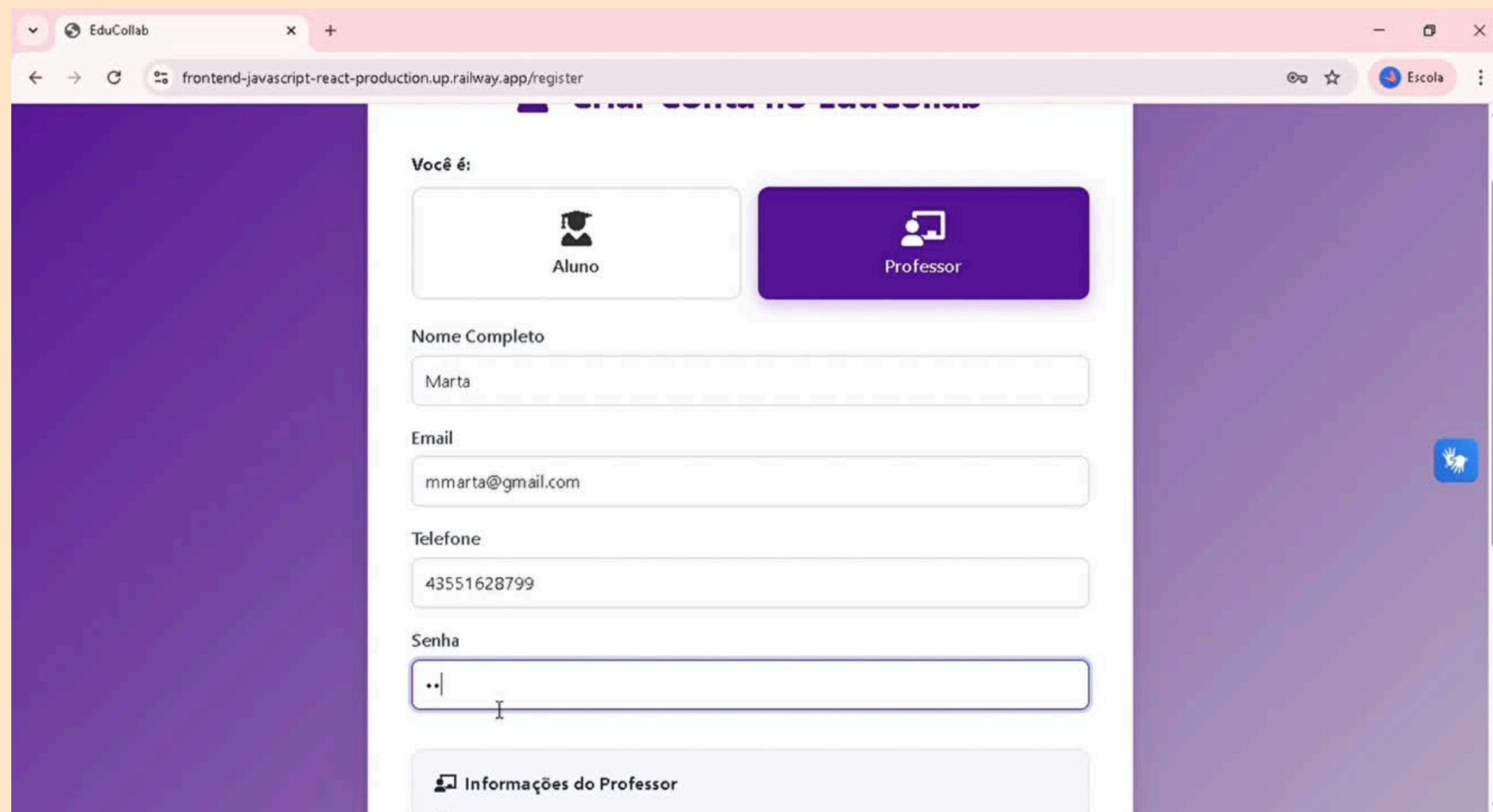
    # SQLAlchemy lida com o mapeamento para JSON automaticamente via models.py
    new_question = Question.model_validate(question_data)

    session.add(new_question)
    session.commit()
    session.refresh(new_question)

    return new_question

@router.delete("/question/{question_id}", response_model=QuestionDisplay)
async def delete_question(
    question_id: int,
    current_user: users.User = Depends(users.get_current_user),
    session: Session = Depends(get_session)
):
    if current_user.user_type != 'teacher':
        raise HTTPException(status_code=404, detail="Apenas professores podem deletar perguntas")
```

Demonstração



The screenshot shows a web browser window with the address bar displaying 'frontend-javascript-react-production.up.railway.app/register'. The page title is 'EduCollab'. The registration form is titled 'Você é:' and has two options: 'Aluno' (Student) and 'Professor' (Teacher). The 'Professor' option is selected. Below the options, there are input fields for 'Nome Completo' (Full Name), 'Email', 'Telefone' (Phone), and 'Senha' (Password). The 'Nome Completo' field contains 'Marta', the 'Email' field contains 'mmarta@gmail.com', and the 'Telefone' field contains '43551628799'. The 'Senha' field is currently empty and has a password strength indicator. At the bottom of the form, there is a section titled 'Informações do Professor' (Teacher Information).

Você é:

☐ Aluno

☒ Professor

Nome Completo

Marta

Email

mmarta@gmail.com

Telefone

43551628799

Senha

••

Informações do Professor

[vídeo completo](#)

The background is a purple grid. In the top-left corner, there are three books: a yellow one at the bottom, a blue one in the middle, and a purple one on top. In the top-right corner, there is an open notebook with a purple cover and a yellow pencil resting on it. In the bottom-left corner, there is a white speech bubble with a yellow cloud-like shape inside it. In the bottom-right corner, there are three books: a blue one on the left, a yellow one in the middle, and a purple one on the right.

OBRIGADO !

[GitHub do projeto](#)