```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_jwt_extended import JWTManager, create_access_token,
jwt_required, get_jwt_identity
from flask_cors import CORS
from datetime import timedelta
from werkzeug.security import generate_password_hash,
check_password_hash
import os

app = Flask(__name__)
CORS(app)

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///questoes.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['JWT_SECRET_KEY'] = os.environ.get('JWT_SECRET_KEY',
'sua_chave_secreta_segura')
app.config['JWT_ACCESS_TOKEN_EXPIRES'] = timedelta(days=1)

db = SQLAlchemy(app)
jwt = JWTManager(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
    nome = db.Column(db.String(120), nullable=False)
    tipo_usuario = db.Column(db.String(20), nullable=False)

class Question(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    enunciado = db.Column(db.Text, nullable=False)
    alternativa_a = db.Column(db.Text, nullable=False)
    alternativa_b = db.Column(db.Text, nullable=False)
    alternativa_c = db.Column(db.Text, nullable=False)
    alternativa_d = db.Column(db.Text, nullable=False)
    alternativa_e = db.Column(db.Text)
```

```
    resposta_correta = db.Column(db.String(1), nullable=False)
    explicacao = db.Column(db.Text)
    criado_por = db.Column(db.Integer, db.ForeignKey('user.id'))

@app.route('/auth/register', methods=['POST'])
def register():
    data = request.json
    if User.query.filter_by(email=data['email']).first():
        return jsonify({"msg": "Email já cadastrado"}), 409
    hashed = generate_password_hash(data['password'])
    user = User(email=data['email'],
                password_hash=hashed,
                nome=data['nome'],
                tipo_usuario=data['tipo_usuario'])
    db.session.add(user)
    db.session.commit()
    return jsonify({"msg": "Usuário criado com sucesso"}), 201

@app.route('/auth/login', methods=['POST'])
def login():
    data = request.json
    user = User.query.filter_by(email=data['email']).first()
    if not user or not check_password_hash(user.password_hash,
data['password']):
        return jsonify({"msg": "Credenciais inválidas"}), 401
    token = create_access_token(identity=user.id)
    return jsonify({"token": token, "nome": user.nome})

@app.route('/questions', methods=['GET'])
@jwt_required()
def listar_questoes():
    questoes = Question.query.limit(10).all()
    return jsonify([{
        "id": q.id,
        "enunciado": q.enunciado,
        "alternativa_a": q.alternativa_a,
        "alternativa_b": q.alternativa_b,
        "alternativa_c": q.alternativa_c,
```

```
        "alternativa_d": q.alternativa_d,
        "alternativa_e": q.alternativa_e,
    } for q in questoes])

@app.route('/questions/<int:qid>/answer', methods=['POST'])
@jwt_required()
def responder_questao(qid):
    data = request.json
    questao = Question.query.get_or_404(qid)
    correta = (questao.resposta_correta.lower() ==
data['resposta_usuario'].lower())
    return jsonify({"correta": correta, "explicacao": questao.explicacao})

@app.route('/questions', methods=['POST'])
@jwt_required()
def criar_questao():
    data = request.json
    user_id = get_jwt_identity()
    q = Question(
        enunciado=data['enunciado'],
        alternativa_a=data['alternativa_a'],
        alternativa_b=data['alternativa_b'],
        alternativa_c=data['alternativa_c'],
        alternativa_d=data['alternativa_d'],
        alternativa_e=data.get('alternativa_e'),
        resposta_correta=data['resposta_correta'],
        explicacao=data.get('explicacao'),
        criado_por=user_id
    )
    db.session.add(q)
    db.session.commit()
    return jsonify({"msg": "Questão criada"}), 201

if __name__ == '__main__':
    db.create_all()
    port = int(os.environ.get("PORT", 5000))
    app.run(debug=True, host="0.0.0.0", port=port)
```

```
import requests

API_URL = 'http://localhost:5000'
EMAIL = 'admin@example.com'
PASSWORD = '123456'

def registrar_usuario():
    res = requests.post(f'{API_URL}/auth/register', json={
        'email': EMAIL,
        'password': PASSWORD,
        'nome': 'Admin',
        'tipo_usuario': 'medico'
    })
    print('Registro:', res.status_code)

def fazer_login():
    res = requests.post(f'{API_URL}/auth/login', json={
        'email': EMAIL,
        'password': PASSWORD
    })
    print('Login:', res.status_code)
    return res.json()['token']
```

```
def criar_questao(token, enunciado, alternativas, correta, explicacao):
    headers = {'Authorization': f'Bearer {token}'}
    payload = {
        'enunciado': enunciado,
        'alternativa_a': alternativas['a'],
        'alternativa_b': alternativas['b'],
        'alternativa_c': alternativas['c'],
        'alternativa_d': alternativas['d'],
        'alternativa_e': alternativas.get('e'),
        'resposta_correta': correta,
        'explicacao': explicacao
    }
    res = requests.post(f'{API_URL}/questions', json=payload,
headers=headers)
    print('Questão criada:', res.status_code)

if __name__ == '__main__':
    registrar_usuario()
    token = fazer_login()
    criar_questao(token,
        "Qual é o principal neurotransmissor envolvido na doença de
Parkinson?",
        {'a': 'Serotonina', 'b': 'Dopamina', 'c': 'Acetilcolina', 'd': 'Noradrenalina',
'e': 'Glutamato'},
        'b',
        "A dopamina é o neurotransmissor cuja deficiência está associada à
doença de Parkinson."
    )
    criar_questao(token,
        "Qual dos seguintes antibióticos é uma cefalosporina de terceira
geração?",
        {'a': 'Cefalexina', 'b': 'Cefazolina', 'c': 'Ceftriaxona', 'd': 'Cefadroxila', 'e':
'Cefuroxima'},
        'c',
        "A ceftriaxona é uma cefalosporina de terceira geração amplamente
utilizada."
    )
```

```
Flask
Flask-SQLAlchemy
Flask-JWT-Extended
Flask-CORS
Werkzeug
requests


services:
  - type: web
    name: backend-questoes
    env: python
    plan: free
    buildCommand: "pip install -r requirements.txt"
    startCommand: "python backend_flask_api.py"
    envVars:
      - key: FLASK_ENV
        value: production
```