## BACKEND – DJANGO REST FRAMEWORK

### 1. Criar o Projeto

bash

django-admin startproject investimento\_dashboard

cd investimento\_dashboard

python manage.py startapp carteira

### 2. Instalar Dependências

bash

pip install djangorestframework

pip install pyodbc

pip install django-cors-headers

pip install djangorestframework-simplejwt

### 3. Configurações no settings.py

python

INSTALLED\_APPS = [

...

'rest\_framework',

'carteira',

'corsheaders',

]

MIDDLEWARE = [

'corsheaders.middleware.CorsMiddleware',

...

]

CORS\_ALLOW\_ALL\_ORIGINS = True

REST\_FRAMEWORK = {

'DEFAULT\_AUTHENTICATION\_CLASSES': (

'rest\_framework\_simplejwt.authentication.JWTAuthentication',

)

}

DATABASES = {

'default': {

'ENGINE': 'sql\_server.pyodbc',

'NAME': 'nome\_do\_banco',

'USER': 'usuario',

'PASSWORD': 'senha',

'HOST': 'localhost',

'PORT': '1433',

'OPTIONS': {

'driver': 'ODBC Driver 17 for SQL Server',

},

}

}

### 4. Models – carteira/models.py

python

from django.db import models

from django.contrib.auth.models import User

class Fundo(models.Model):

nome = models.CharField(max\_length=100)

ticker = models.CharField(max\_length=10)

tipo = models.CharField(max\_length=50)

valor\_cota = models.DecimalField(max\_digits=10, decimal\_places=2)

def \_\_str\_\_(self):

return f"{self.nome} ({self.ticker})"

class Movimentacao(models.Model):

TIPO\_CHOICES = (('aporte', 'Aporte'), ('resgate', 'Resgate'))

fundo = models.ForeignKey(Fundo, on\_delete=models.CASCADE)

usuario = models.ForeignKey(User, on\_delete=models.CASCADE)

data = models.DateField()

valor = models.DecimalField(max\_digits=10, decimal\_places=2)

tipo = models.CharField(max\_length=10, choices=TIPO\_CHOICES)

cotas = models.DecimalField(max\_digits=10, decimal\_places=4)

def \_\_str\_\_(self):

return f"{self.tipo} de {self.valor} em {self.fundo.ticker}"

### 5. Serializers – carteira/serializers.py

python

from rest\_framework import serializers

from .models import Fundo, Movimentacao

from django.contrib.auth.models import User

class FundoSerializer(serializers.ModelSerializer):

class Meta:

model = Fundo

fields = '\_\_all\_\_'

class MovimentacaoSerializer(serializers.ModelSerializer):

class Meta:

model = Movimentacao

fields = '\_\_all\_\_'

class UserSerializer(serializers.ModelSerializer):

class Meta:

model = User

fields = ['id', 'username', 'email']

### 6. Views (API) – carteira/views.py

python

from rest\_framework import viewsets

from rest\_framework.permissions import IsAuthenticated

from .models import Fundo, Movimentacao

from .serializers import FundoSerializer, MovimentacaoSerializer

class FundoViewSet(viewsets.ModelViewSet):

queryset = Fundo.objects.all()

serializer\_class = FundoSerializer

permission\_classes = [IsAuthenticated]

class MovimentacaoViewSet(viewsets.ModelViewSet):

queryset = Movimentacao.objects.all()

serializer\_class = MovimentacaoSerializer

permission\_classes = [IsAuthenticated]

def get\_queryset(self):

return self.queryset.filter(usuario=self.request.user)

def perform\_create(self, serializer):

serializer.save(usuario=self.request.user)

### 7. URLs – carteira/urls.py

python

from rest\_framework.routers import DefaultRouter

from .views import FundoViewSet, MovimentacaoViewSet

router = DefaultRouter()

router.register(r'fundos', FundoViewSet)

router.register(r'movimentacoes', MovimentacaoViewSet)

urlpatterns = router.urls

### Principal URLConf – investimento\_dashboard/urls.py

python

from django.contrib import admin

from django.urls import path, include

from rest\_framework\_simplejwt.views import (

TokenObtainPairView,

TokenRefreshView,

)

urlpatterns = [

path('admin/', admin.site.urls),

path('api/', include('carteira.urls')),

path('api/token/', TokenObtainPairView.as\_view(), name='token\_obtain\_pair'),

path('api/token/refresh/', TokenRefreshView.as\_view(), name='token\_refresh'),

]

## 🌐 FRONTEND – REACT

### 1. Criar Projeto React

bash

npx create-react-app investimento-dashboard-frontend

cd investimento-dashboard-frontend

npm install axios react-router-dom

### 2. Estrutura Sugerida

pgsql

src/

├── components/

│ ├── FundoForm.jsx

│ ├── MovimentacaoForm.jsx

│ ├── Dashboard.jsx

│ └── Login.jsx

├── services/

│ └── api.js

├── App.js

└── index.js

### 3. Exemplo de Serviço API – services/api.js

javascript

import axios from 'axios';

const api = axios.create({

baseURL: 'http://localhost:8000/api/',

});

export const login = (credentials) => api.post('token/', credentials);

export const getFundos = (token) => api.get('fundos/', {

headers: { Authorization: `Bearer ${token}` }

});

export const postMovimentacao = (data, token) => api.post('movimentacoes/', data, {

headers: { Authorization: `Bearer ${token}` }

});

export default api;

### 4. Dashboard – components/Dashboard.jsx

jsx

import React, { useEffect, useState } from 'react';

import { getFundos } from '../services/api';

const Dashboard = ({ token }) => {

const [fundos, setFundos] = useState([]);

useEffect(() => {

getFundos(token).then(res => setFundos(res.data));

}, [token]);

return (

<div>

<h2>Fundos de Investimento</h2>

<ul>

{fundos.map(f => (

<li key={f.id}>{f.nome} - {f.valor\_cota}</li>

))}

</ul>

</div>

);

};

export default Dashboard;

Vamos criar o esqueleto inicial completo para o desafio técnico "Dashboard de Investimento" utilizando **Flask RESTful**, **React** e **SQL Server (SQLExpress)**. Este guia inclui backend, frontend, autenticação JWT, painel de controle e dockerização, com foco na clareza do código e estruturação adequada.

## 🛠️ Backend – Flask RESTful + SQLAlchemy + SQL Server

### 1. Estrutura do Projeto

pgsql

CopiarEditar

investimento-dashboard/

├── backend/

│ ├── app/

│ │ ├── \_\_init\_\_.py

│ │ ├── models.py

│ │ ├── schemas.py

│ │ ├── resources.py

│ │ ├── auth.py

│ │ └── config.py

│ ├── Dockerfile

│ ├── requirements.txt

│ └── run.py

└── frontend/

├── src/

│ ├── components/

│ │ ├── Dashboard.jsx

│ │ ├── FundoForm.jsx

│ │ ├── MovimentacaoForm.jsx

│ │ └── Login.jsx

│ ├── services/

│ │ └── api.js

│ ├── App.js

│ └── index.js

├── Dockerfile

└── package.json

### 2. Backend – Flask + SQLAlchemy + Flask-RESTful

#### requirements.txt

nginx

CopiarEditar

Flask

Flask-SQLAlchemy

Flask-RESTful

Flask-JWT-Extended

pyodbc

Flask-Cors

#### run.py

python

CopiarEditar

from app import create\_app

app = create\_app()

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

#### app/\_\_init\_\_.py

python

CopiarEditar

from flask import Flask

from flask\_sqlalchemy import SQLAlchemy

from flask\_restful import Api

from flask\_jwt\_extended import JWTManager

from flask\_cors import CORS

db = SQLAlchemy()

api = Api()

jwt = JWTManager()

def create\_app():

app = Flask(\_\_name\_\_)

app.config.from\_object('app.config.Config')

db.init\_app(app)

api.init\_app(app)

jwt.init\_app(app)

CORS(app)

return app

#### app/config.py

python

class Config:

SQLALCHEMY\_DATABASE\_URI = 'mssql+pyodbc://username:password@localhost:1433/db\_name?driver=SQL+Server+Native+Client+11.0'

SQLALCHEMY\_TRACK\_MODIFICATIONS = False

JWT\_SECRET\_KEY = 'your\_jwt\_secret\_key'

#### app/models.py

python

from datetime import datetime

from app import db

class Fundo(db.Model):

id = db.Column(db.Integer, primary\_key=True)

nome = db.Column(db.String(100), nullable=False)

ticker = db.Column(db.String(10), unique=True, nullable=False)

tipo = db.Column(db.String(50), nullable=False)

valor\_cota = db.Column(db.Float, nullable=False)

class Movimentacao(db.Model):

id = db.Column(db.Integer, primary\_key=True)

fundo\_id = db.Column(db.Integer, db.ForeignKey('fundo.id'), nullable=False)

data = db.Column(db.Date, default=datetime.utcnow)

valor = db.Column(db.Float, nullable=False)

tipo = db.Column(db.String(10), nullable=False)

cotas = db.Column(db.Float, nullable=False)

fundo = db.relationship('Fundo', backref=db.backref('movimentacoes', lazy=True))

#### app/schemas.py

python

from marshmallow import Schema, fields

class FundoSchema(Schema):

id = fields.Int(dump\_only=True)

nome = fields.Str()

ticker = fields.Str()

tipo = fields.Str()

valor\_cota = fields.Float()

class MovimentacaoSchema(Schema):

id = fields.Int(dump\_only=True)

fundo\_id = fields.Int()

data = fields.Date()

valor = fields.Float()

tipo = fields.Str()

cotas = fields.Float()

#### app/resources.py

python

CopiarEditar

from flask\_restful import Resource

from flask\_jwt\_extended import jwt\_required

from app import db

from app.models import Fundo, Movimentacao

from app.schemas import FundoSchema, MovimentacaoSchema

class FundoResource(Resource):

@jwt\_required()

def get(self, fundo\_id):

fundo = Fundo.query.get\_or\_404(fundo\_id)

return FundoSchema().dump(fundo)

@jwt\_required()

def put(self, fundo\_id):

fundo = Fundo.query.get\_or\_404(fundo\_id)

# Atualizar fundo

db.session.commit()

return FundoSchema().dump(fundo)

@jwt\_required()

def delete(self, fundo\_id):

fundo = Fundo.query.get\_or\_404(fundo\_id)

db.session.delete(fundo)

db.session.commit()

return '', 204

class MovimentacaoResource(Resource):

@jwt\_required()

def get(self, movimentacao\_id):

movimentacao = Movimentacao.query.get\_or\_404(movimentacao\_id)

return MovimentacaoSchema().dump(movimentacao)

@jwt\_required()

def post(self):

# Criar movimentacao

db.session.commit()

return MovimentacaoSchema().dump(movimentacao), 201

#### app/auth.py

python

CopiarEditar

from flask\_jwt\_extended import create\_access\_token

from app import db

from app.models import User

def login\_user(username, password):

user = User.query.filter\_by(username=username).first()

if user and user.check\_password(password):

access\_token = create\_access\_token(identity=user.id)

return {'access\_token': access\_token}, 200

return {'message': 'Invalid credentials'}, 401

#### app/resources.py (continuação)

python

from flask\_restful import Resource

from flask\_jwt\_extended import jwt\_required

from app import db

from app.models import Fundo, Movimentacao

from app.schemas import FundoSchema, MovimentacaoSchema

from app.auth import login\_user

class LoginResource(Resource):

def post(self):

# Lógica de login

return login\_user(username, password)

### 3. Frontend – React

#### package.json

json

{

"name": "frontend",

"version": "0.1.0",

"private": true,

"dependencies": {

"axios": "^0.21.1",

"react": "^17.0.2",

"react-dom": "^17.0.2",

"react-router-dom": "^5.1.0"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test",

"eject": "react-scripts eject"

}

}

#### src/services/api.js

javascript

CopiarEditar

import axios from 'axios';

::contentReference[oaicite:0]{index=0}