### ****Planejamento das Entidades****

#### Fundos de Investimento

python

class Fundo(models.Model):

nome = models.CharField(max\_length=255)

codigo = models.CharField(max\_length=20) # Ticker

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

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

#### Movimentações

python

class Movimentacao(models.Model):

TIPO\_CHOICES = [

('aporte', 'Aporte'),

('resgate', 'Resgate')

]

fundo = models.ForeignKey(Fundo, 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)

### 🔄 ****Lógica de Carteira****

Você pode calcular o saldo da carteira dinamicamente com base nas movimentações registradas:

python

# Exemplo simplificado de lógica

from django.db.models import Sum, Case, When, F

def calcular\_saldo():

aportes = Movimentacao.objects.filter(tipo='aporte').aggregate(total=Sum('valor'))['total'] or 0

resgates = Movimentacao.objects.filter(tipo='resgate').aggregate(total=Sum('valor'))['total'] or 0

return aportes - resgates

### 📡 ****API Endpoints (DRF)****

1. GET /fundos/ – Listar fundos
2. POST /fundos/ – Criar fundo
3. GET /movimentacoes/ – Listar movimentações
4. POST /movimentacoes/ – Criar movimentação
5. GET /carteira/saldo/ – Retornar saldo atual

### 💻 ****Frontend com React****

No React, você pode ter 3 páginas/componentes principais:

1. **Cadastro de Fundos** – Formulário simples
2. **Cadastro de Movimentações** – Formulário com select para fundo e tipo
3. **Dashboard** – Tabela de movimentações + saldo exibido no topo

Use fetch ou axios pra consumir as rotas da API. Pode usar useEffect pra buscar os dados ao carregar os componentes.

### 🧼 ****Boas Práticas****

* Separação clara entre camadas: models, views, serializers
* Código limpo e comentado
* Uso adequado de Serializer e ModelSerializer
* Componentização no React para manter o código limpo

## 📁 Estrutura do Projeto

bash

CopiarEditar

investimentos/

├── backend/

│ ├── manage.py

│ ├── investimentos/ # Projeto Django

│ └── api/ # App Django com a lógica

├── frontend/

│ └── (React app aqui)

└── README.md

## 🧱 1. Setup Backend (Django + DRF + SQLite)

### 1.1 Criar o projeto

bash

CopiarEditar

mkdir investimentos && cd investimentos

python -m venv venv

source venv/bin/activate # no Windows: venv\Scripts\activate

pip install django djangorestframework

django-admin startproject backend .

cd backend

python manage.py startapp api

### 1.2 Configurações do projeto

#### backend/settings.py

* Adicionar 'rest\_framework' e 'api' no INSTALLED\_APPS

python

CopiarEditar

INSTALLED\_APPS = [

...

'rest\_framework',

'api',

]

* Definir CORS se for rodar React separado:

bash

CopiarEditar

pip install django-cors-headers

python

CopiarEditar

# settings.py

INSTALLED\_APPS += ['corsheaders']

MIDDLEWARE.insert(0, 'corsheaders.middleware.CorsMiddleware')

CORS\_ALLOW\_ALL\_ORIGINS = True # ou configure especificamente

## 🧩 Modelos Iniciais

### api/models.py

python

from django.db import models

class Fundo(models.Model):

nome = models.CharField(max\_length=255)

codigo = models.CharField(max\_length=20)

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

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

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

return self.nome

class Movimentacao(models.Model):

TIPO\_CHOICES = [

('aporte', 'Aporte'),

('resgate', 'Resgate')

]

fundo = models.ForeignKey(Fundo, 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)

## 🔄 Serializers

### api/serializers.py

python

from rest\_framework import serializers

from .models import Fundo, Movimentacao

class FundoSerializer(serializers.ModelSerializer):

class Meta:

model = Fundo

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

class MovimentacaoSerializer(serializers.ModelSerializer):

class Meta:

model = Movimentacao

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

## 🔁 Views e Rotas

### api/views.py

python

CopiarEditar

from rest\_framework import viewsets

from .models import Fundo, Movimentacao

from .serializers import FundoSerializer, MovimentacaoSerializer

class FundoViewSet(viewsets.ModelViewSet):

queryset = Fundo.objects.all()

serializer\_class = FundoSerializer

class MovimentacaoViewSet(viewsets.ModelViewSet):

queryset = Movimentacao.objects.all().order\_by('-data')

serializer\_class = MovimentacaoSerializer

### backend/urls.py

python

CopiarEditar

from django.contrib import admin

from django.urls import path, include

from rest\_framework.routers import DefaultRouter

from api.views import FundoViewSet, MovimentacaoViewSet

router = DefaultRouter()

router.register(r'fundos', FundoViewSet)

router.register(r'movimentacoes', MovimentacaoViewSet)

urlpatterns = [

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

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

]

## ⚛️ 2. Setup Frontend (React)

### 2.1 Criar o app React

bash

cd ..

npx create-react-app frontend

cd frontend

npm install axios

### 2.2 Estrutura sugerida

css

CopiarEditar

frontend/

├── src/

│ ├── components/

│ │ ├── FundosForm.js

│ │ ├── MovimentacoesForm.js

│ │ └── Dashboard.js

│ ├── services/

│ │ └── api.js

│ ├── App.js

### 2.3 Exemplo de consumo da API

#### services/api.js

javascript

CopiarEditar

import axios from 'axios';

const api = axios.create({

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

});

export default api;

#### Dashboard.js

jsx

CopiarEditar

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

import api from '../services/api';

export default function Dashboard() {

const [movimentacoes, setMovimentacoes] = useState([]);

const [saldo, setSaldo] = useState(0);

useEffect(() => {

api.get('movimentacoes/').then(res => setMovimentacoes(res.data));

// Você pode calcular o saldo aqui ou ter um endpoint específico

}, []);

return (

<div>

<h2>Dashboard</h2>

<p>Saldo: R$ {saldo.toFixed(2)}</p>

<table>

<thead>

<tr>

<th>Data</th><th>Tipo</th><th>Valor</th><th>Fundo</th>

</tr>

</thead>

<tbody>

{movimentacoes.map(m => (

<tr key={m.id}>

<td>{m.data}</td>

<td>{m.tipo}</td>

<td>R$ {m.valor}</td>

<td>{m.fundo}</td>

</tr>

))}

</tbody>

</table>

</div>

);

}

## Docker (opcional)

backend/Dockerfile simples:

Dockerfile

FROM python:3.11

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

## ✅ CI/CD (GitHub Actions - opcional)

.github/workflows/deploy.yml:

yaml

name: CI/CD

on:

push:

branches: [ main ]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

- name: Set up Python

uses: actions/setup-python@v4

with:

python-version: '3.11'

- name: Install dependencies

run: |

pip install -r backend/requirements.txt

- name: Run tests

run: |

cd backend

python manage.py test

- name: Build Docker image

run: docker build -t merito-backend ./backend

- name: Simulate deploy

run: echo "Deploy 🎉"