

# Portfolio Starter (Django + React + Docker)

Minimal, clean, and fast scaffold to showcase mixed academic + professional work. Local dev runs with Docker; deploy React on Netlify (or Vercel) and Django API on Railway/Render/Fly.

---

## Repo structure

```
portfolio/
├─ docker-compose.yml
├─ .env.example
├─ README.md
├─ backend/
│   ├─ Dockerfile
│   ├─ requirements.txt
│   ├─ manage.py
│   ├─ core/
│   │   ├─ __init__.py
│   │   ├─ settings.py
│   │   ├─ urls.py
│   │   └─ wsgi.py
│   └─ api/
│       ├─ __init__.py
│       ├─ admin.py
│       ├─ apps.py
│       ├─ models.py
│       ├─ serializers.py
│       ├─ views.py
│       └─ urls.py
└─ frontend/
    ├─ Dockerfile
    ├─ index.html
    ├─ package.json
    ├─ vite.config.ts
    ├─ tailwind.config.js
    └─ src/
        ├─ main.tsx
        ├─ App.tsx
        ├─ lib/api.ts
        ├─ components/
        │   ├─ Navbar.tsx
        │   ├─ Footer.tsx
        │   ├─ ProjectCard.tsx
        │   └─ TagPill.tsx
        └─ pages/
            ├─ Home.tsx
            └─ Projects.tsx
```

```
└─ ProjectDetail.tsx
└─ Research.tsx
└─ About.tsx
└─ Contact.tsx
```

## docker-compose.yml (dev)

```
version: "3.9"
services:
  db:
    image: postgres:15
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    volumes:
      - db_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  backend:
    build: ./backend
    command: bash -c "python manage.py migrate && python manage.py runserver 0.0.0.0:8000"
    environment:
      DJANGO_DEBUG: "1"
      DJANGO_SECRET_KEY: ${DJANGO_SECRET_KEY}
      DATABASE_URL: postgres://${POSTGRES_USER}:${POSTGRES_PASSWORD}@db:5432/${POSTGRES_DB}
      ALLOWED_HOSTS: "*"
      CORS_ALLOWED_ORIGINS: ${CORS_ALLOWED_ORIGINS}
    volumes:
      - ./backend:/app
    ports:
      - "8000:8000"
    depends_on:
      - db

  frontend:
    build: ./frontend
    volumes:
      - ./frontend:/app
      - /app/node_modules
    ports:
      - "5173:5173"
    environment:
      - VITE_API_BASE=http://localhost:8000
    command: ["npm", "run", "dev", "--", "--host"]
```

```
volumes:
  db_data:
```

## **.env.example**

```
POSTGRES_DB=portfolio
POSTGRES_USER=postgres
POSTGRES_PASSWORD=postgres
DJANGO_SECRET_KEY=change-me
CORS_ALLOWED_ORIGINS=http://localhost:5173
```

## **Backend (Django + DRF)**

### **backend/Dockerfile**

```
FROM python:3.12-slim
ENV PYTHONDONTWRITEBYTECODE=1 \
    PYTHONUNBUFFERED=1
WORKDIR /app
RUN apt-get update && apt-get install -y build-essential libpq-dev && rm -rf /var/lib/apt/lists/*
COPY requirements.txt /app/
RUN pip install --no-cache-dir -r requirements.txt
COPY . /app
```

### **backend/requirements.txt**

```
Django>=5.0
psycopg2-binary>=2.9
python-dotenv
whitenoise
django-rest-framework
django-filter
django-cors-headers
dj-database-url
```

### **backend/core/settings.py**

```
import os
from pathlib import Path
import dj_database_url

BASE_DIR = Path(__file__).resolve().parent.parent
```

```

SECRET_KEY = os.getenv("DJANGO_SECRET_KEY", "dev")
DEBUG = os.getenv("DJANGO_DEBUG", "0") == "1"
ALLOWED_HOSTS = os.getenv("ALLOWED_HOSTS", "*").split(",")

INSTALLED_APPS = [

    'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions',
    'rest_framework', 'django_filters', 'corsheaders', 'api',
]
MIDDLEWARE =
['corsheaders.middleware.CorsMiddleware', 'django.middleware.security.SecurityMiddleware', 'django.contrib.sessions.middleware.SessionMiddleware', 'django.middleware.common.CommonMiddleware', 'django.middleware.csrf.CsrfViewMiddleware', 'django.contrib.auth.middleware.AuthenticationMiddleware', 'django.contrib.messages.middleware.MessageMiddleware']
ROOT_URLCONF = 'core.urls'
TEMPLATES = [{
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [],
    'APP_DIRS': True,
    'OPTIONS': {'context_processors':
        ['django.template.context_processors.debug', 'django.template.context_processors.request', 'django.contrib.auth.context_processors.auth', 'django.contrib.messages.context_processors.messages']
    }]
WSGI_APPLICATION = 'core.wsgi.application'

DATABASES = {
    'default': dj_database_url.parse(os.getenv('DATABASE_URL', 'postgres://postgres:postgres@localhost:5432/portfolio'))
}

AUTH_PASSWORD_VALIDATORS = []

LANGUAGE_CODE = 'pt-br'
TIME_ZONE = 'America/Recife'
USE_I18N = True
USE_TZ = True

STATIC_URL = 'static/'
STATIC_ROOT = BASE_DIR / 'staticfiles'

REST_FRAMEWORK = {
    'DEFAULT_FILTER_BACKENDS':
        ['django_filters.rest_framework.DjangoFilterBackend'],
    'DEFAULT_PAGINATION_CLASS':
        'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 12,
}

CORS_ALLOWED_ORIGINS = os.getenv("CORS_ALLOWED_ORIGINS", "http://localhost:5173").split(",")

```

## backend/core/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('api.urls')),
]
```

## backend/api/models.py

```
from django.db import models

class Tag(models.Model):
    name = models.CharField(max_length=50, unique=True)
    def __str__(self): return self.name

class Project(models.Model):
    CATEGORY_CHOICES = [
        ("software", "Software"), ("hardware", "Hardware"),
        ("research", "Pesquisa"), ("competition", "Competição"), ("data", "Dados/IA"),
    ]
    title = models.CharField(max_length=200)
    slug = models.SlugField(unique=True)
    category = models.CharField(max_length=20, choices=CATEGORY_CHOICES)
    short_desc = models.TextField(max_length=400)
    long_desc = models.TextField(blank=True)
    tech_stack = models.CharField(max_length=300, blank=True)
    start_date = models.DateField(null=True, blank=True)
    end_date = models.DateField(null=True, blank=True)
    featured = models.BooleanField(default=False)
    repo_url = models.URLField(blank=True)
    demo_url = models.URLField(blank=True)
    cover_url = models.URLField(blank=True)
    tags = models.ManyToManyField(Tag, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['-featured', '-created_at']

    def __str__(self): return self.title

class Publication(models.Model):
    title = models.CharField(max_length=300)
    venue = models.CharField(max_length=200, blank=True)
    year = models.IntegerField(null=True, blank=True)
    url = models.URLField(blank=True)
    authors = models.CharField(max_length=400, blank=True)
```

```

        highlight = models.BooleanField(default=False)
        def __str__(self): return self.title

class Achievement(models.Model):
    """Competitions/awards/certifications"""
    title = models.CharField(max_length=200)
    org = models.CharField(max_length=200, blank=True)
    year = models.IntegerField(null=True, blank=True)
    url = models.URLField(blank=True)
    def __str__(self): return self.title

```

## backend/api/serializers.py

```

from rest_framework import serializers
from .models import Project, Tag, Publication, Achievement

class TagSerializer(serializers.ModelSerializer):
    class Meta:
        model = Tag
        fields = ['id', 'name']

class ProjectSerializer(serializers.ModelSerializer):
    tags = TagSerializer(many=True, read_only=True)
    class Meta:
        model = Project
        fields = '__all__'

class PublicationSerializer(serializers.ModelSerializer):
    class Meta:
        model = Publication
        fields = '__all__'

class AchievementSerializer(serializers.ModelSerializer):
    class Meta:
        model = Achievement
        fields = '__all__'

```

## backend/api/views.py

```

from rest_framework import viewsets, filters
from .models import Project, Tag, Publication, Achievement
from .serializers import ProjectSerializer, TagSerializer,
PublicationSerializer, AchievementSerializer

class ProjectViewSet(viewsets.ModelViewSet):
    queryset = Project.objects.all()
    serializer_class = ProjectSerializer
    filter_backends = [filters.SearchFilter, filters.OrderingFilter]
    search_fields =

```

```

['title', 'short_desc', 'long_desc', 'tech_stack', 'category', 'tags__name']
    ordering_fields = ['created_at', 'start_date']

class TagViewSet(viewsets.ModelViewSet):
    queryset = Tag.objects.all()
    serializer_class = TagSerializer

class PublicationViewSet(viewsets.ModelViewSet):
    queryset = Publication.objects.all()
    serializer_class = PublicationSerializer

class AchievementViewSet(viewsets.ModelViewSet):
    queryset = Achievement.objects.all()
    serializer_class = AchievementSerializer

```

### backend/api/urls.py

```

from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import ProjectViewSet, TagViewSet, PublicationViewSet,
AchievementViewSet

router = DefaultRouter()
router.register('projects', ProjectViewSet)
router.register('tags', TagViewSet)
router.register('publications', PublicationViewSet)
router.register('achievements', AchievementViewSet)

urlpatterns = [
    path('api/', include(router.urls)),
]

```

### backend/api/admin.py

```

from django.contrib import admin
from .models import Project, Tag, Publication, Achievement

@admin.register(Project)
class ProjectAdmin(admin.ModelAdmin):
    list_display = ("title", "category", "featured", "created_at")
    list_filter = ("category", "featured", "tags")
    search_fields = ("title", "short_desc", "tech_stack")
    prepopulated_fields = {"slug": ("title",)}

admin.site.register(Tag)
admin.site.register(Publication)
admin.site.register(Achievement)

```

## Frontend (React + Vite + Tailwind)

### frontend/Dockerfile

```
FROM node:20-slim
WORKDIR /app
COPY package.json package-lock.json* pnpm-lock.yaml* yarn.lock* ./
RUN npm ci || npm install
COPY . .
EXPOSE 5173
```

### frontend/package.json

```
{
  "name": "portfolio",
  "private": true,
  "version": "0.1.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview --host"
  },
  "dependencies": {
    "axios": "^1.7.7",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.26.1"
  },
  "devDependencies": {
    "@types/react": "^18.2.66",
    "@types/react-dom": "^18.2.22",
    "@vitejs/plugin-react": "^4.3.1",
    "autoprefixer": "^10.4.19",
    "postcss": "^8.4.41",
    "tailwindcss": "^3.4.10",
    "typescript": "^5.5.4",
    "vite": "^5.4.1"
  }
}
```

### frontend/tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
export default {
  content: ["../index.html", "./src/**/*.{ts,tsx}"],
  theme: { extend: {} },
}
```



```
  plugins: [],  
}
```

### frontend/vite.config.ts

```
import { defineConfig } from 'vite'  
import react from '@vitejs/plugin-react'  
export default defineConfig({ plugins: [react()] })
```

### frontend/src/main.tsx

```
import React from 'react'  
import ReactDOM from 'react-dom/client'  
import { BrowserRouter, RouterProvider } from 'react-router-dom'  
import './index.css'  
import App from './App'  
import Home from './pages/Home'  
import Projects from './pages/Projects'  
import ProjectDetail from './pages/ProjectDetail'  
import Research from './pages/Research'  
import About from './pages/About'  
import Contact from './pages/Contact'  
  
const router = createBrowserRouter([  
  { path: '/', element: <App/>, children: [  
    { index: true, element: <Home/> },  
    { path: 'projects', element: <Projects/> },  
    { path: 'projects/:slug', element: <ProjectDetail/> },  
    { path: 'research', element: <Research/> },  
    { path: 'about', element: <About/> },  
    { path: 'contact', element: <Contact/> },  
  ] },  
)  
  
ReactDOM.createRoot(document.getElementById('root')!).render(  
  <React.StrictMode>  
    <RouterProvider router={router}/>  
  </React.StrictMode>  
)
```

### frontend/src/App.tsx

```
import { Outlet } from 'react-router-dom'  
import Navbar from './components/Navbar'  
import Footer from './components/Footer'  
  
export default function App(){  
  return (  

```

```

    <div className="min-h-screen flex flex-col bg-neutral-950 text-
neutral-100">
      <Navbar/>
      <main className="flex-1 max-w-5xl w-full mx-auto px-4 py-8">
        <Outlet/>
      </main>
      <Footer/>
    </div>
  )
}

```

### frontend/src/lib/api.ts

```

import axios from 'axios'
const baseURL = import.meta.env.VITE_API_BASE || 'http://localhost:8000'
export const api = axios.create({ baseURL: baseURL + '/api' })

```

### frontend/src/components/Navbar.tsx

```

import { Link, NavLink } from 'react-router-dom'

export default function Navbar(){
  const link = 'px-3 py-2 rounded hover:bg-neutral-800'
  return (
    <header className="sticky top-0 bg-neutral-950/80 backdrop-blur border-b
border-neutral-800">
      <div className="max-w-5xl mx-auto px-4 h-14 flex items-center justify-
between">
        <Link to="/" className="font-semibold">{`<TC Damião />`}</Link>
        <nav className="flex gap-1">
          <NavLink to="/projects" className={link}>Projetos</NavLink>
          <NavLink to="/research" className={link}>Pesquisa</NavLink>
          <NavLink to="/about" className={link}>Sobre</NavLink>
          <NavLink to="/contact" className={link}>Contato</NavLink>
        </nav>
      </div>
    </header>
  )
}

```

### frontend/src/components/Footer.tsx

```

export default function Footer(){
  return (
    <footer className="border-t border-neutral-800 py-6 text-sm text-
neutral-400">
      <div className="max-w-5xl mx-auto px-4 flex justify-between">
        <span>© {new Date().getFullYear()} TC Damião</span>

```

```

        <a href="/" className="hover:text-neutral-200">Voltar ao topo</a>
      </div>
    </footer>
  )
}

```

### frontend/src/components/ProjectCard.tsx

```

import { Link } from 'react-router-dom'

export default function ProjectCard({ p }: { p: any }){
  return (
    <Link to={`/${p.slug}`} className="block border border-
neutral-800 rounded-2xl p-4 hover:bg-neutral-900 transition">
      <div className="text-xs uppercase tracking-wider text-
neutral-400">{p.category}</div>
      <h3 className="text-lg font-semibold mt-1">{p.title}</h3>
      <p className="text-neutral-300 mt-2 line-clamp-2">{p.short_desc}</p>
      <div className="mt-3 text-sm text-neutral-400">{p.tech_stack}</div>
    </Link>
  )
}

```

### frontend/src/components/TagPill.tsx

```

export default function TagPill({ name }: { name: string }){
  return <span className="text-xs px-2 py-1 rounded-full border border-
neutral-700">{name}</span>
}

```

### frontend/src/pages/Home.tsx

```

import { useEffect, useState } from 'react'
import { api } from '../lib/api'
import ProjectCard from '../components/ProjectCard'

export default function Home(){
  const [featured, setFeatured] = useState<any[]>([])
  useEffect(()=>{ api.get('/projects/?ordering=-created_at').then(r=>
setFeatured(r.data.results.slice(0,6))) },[])
  return (
    <section>
      <h1 className="text-3xl font-bold">Portfólio</h1>
      <p className="mt-2 text-neutral-300 max-w-2xl">Projetos de software,
hardware, pesquisa e competições. Minimalista, direto ao ponto.</p>
      <div className="grid md:grid-cols-2 gap-4 mt-6">
        {featured.map(p => <ProjectCard key={p.id} p={p}/>)}
      </div>
    </section>
  )
}

```

```

    </section>
  )
}

```

## frontend/src/pages/Projects.tsx

```

import { useEffect, useState } from 'react'
import { api } from '../lib/api'
import ProjectCard from '../components/ProjectCard'

export default function Projects(){
  const [items,setItems] = useState<any[]>([])
  const [q,setQ] = useState('')
  const [cat,setCat] = useState('')
  useEffect(()=>{ api.get('/projects/').then(r=> setItems(r.data.results)) },
  [])
  const filtered = items.filter(p => (
    (!q ||
    (p.title+p.short_desc+p.tech_stack).toLowerCase().includes(q.toLowerCase()))
    && (!cat || p.category===cat)
  ))
  return (
    <section>
      <h2 className="text-2xl font-semibold">Projetos</h2>
      <div className="mt-4 flex gap-2">
        <input value={q} onChange={e=>setQ(e.target.value)}
placeholder="Buscar" className="bg-neutral-900 border border-neutral-800
rounded px-3 py-2 w-full"/>
        <select value={cat} onChange={e=>setCat(e.target.value)}
className="bg-neutral-900 border border-neutral-800 rounded px-3">
          <option value="">Todas</option>
          <option value="software">Software</option>
          <option value="hardware">Hardware</option>
          <option value="data">Dados/IA</option>
          <option value="research">Pesquisa</option>
          <option value="competition">Competição</option>
        </select>
      </div>
      <div className="grid md:grid-cols-2 gap-4 mt-4">
        {filtered.map(p => <ProjectCard key={p.id} p={p}/> ) }
      </div>
    </section>
  )
}

```

## frontend/src/pages/ProjectDetail.tsx

```

import { useEffect, useState } from 'react'
import { useParams } from 'react-router-dom'

```

```
import { api } from '../lib/api'

export default function ProjectDetail(){
  const { slug } = useParams()
  const [p,setP] = useState<any>(null)
  useEffect(()=>{ api.get(`/projects/?search=${slug}`).then(r=>
    setP(r.data.results.find((x:any)=>x.slug===slug))) },[slug])
  if(!p) return <div>Carregando...</div>
  return (
    <article className="prose prose-invert max-w-none">
      <h1>{p.title}</h1>
      <p>{p.long_desc || p.short_desc}</p>
      {p.repo_url && <p><a href={p.repo_url}>Repositório</a></p>}
      {p.demo_url && <p><a href={p.demo_url}>Demo</a></p>}
    </article>
  )
}
```

## frontend/src/pages/Research.tsx


```
import { useEffect, useState } from 'react'
import { api } from '../lib/api'

export default function Research(){
  const [pubs,setPubs] = useState<any[]>([])
  useEffect(()=>{ api.get('/publications/').then(r=>
    setPubs(r.data.results)) },[])
  return (
    <section>
      <h2 className="text-2xl font-semibold">Publicações</h2>
      <ul className="mt-4 space-y-3">
        {pubs.map(p => (
          <li key={p.id} className="border border-neutral-800 rounded-xl
p-4">
            <div className="font-medium">{p.title}</div>
            <div className="text-sm text-neutral-400">{p.authors} •
{p.venue} {p.year}</div>
            {p.url && <a className="text-sm underline" href={p.url}>Link</a>}
          </li>
        ))}
      </ul>
    </section>
  )
}
```

## frontend/src/pages/About.tsx

```
export default function About(){
  return (
```


```

    <section className="space-y-3">
      <h2 className="text-2xl font-semibold">Sobre</h2>
      <p>Engenharia da Computação  IA aplicada (GNNs, MLPs, RAG), hardware embarcado, pesquisas e competições. Portfólio compacto e direto.</p>
    </section>
  )
}

```

### frontend/src/pages/Contact.tsx

```

export default function Contact(){
  return (
    <section>
      <h2 className="text-2xl font-semibold">Contato</h2>
      <p className="text-neutral-300">Prefere via e-mail? <a
className="underline"
href="mailto:talisoncaedoso@gmail.com">talisoncaedoso@gmail.com</a></p>
      <p className="text-neutral-300">ou LinkedIn/GitHub (links no rodapé 
ajuste conforme preferir).</p>
    </section>
  )
}

```

### frontend/src/index.css

```

@tailwind base;
@tailwind components;
@tailwind utilities;

/* optional: text wrapping for prose */
.prose p { white-space: pre-wrap; }

```

## Quickstart

```

# 1) Clone repo e crie .env
cp .env.example .env

# 2) Suba ambiente
docker compose up --build

# 3) Acesse
# Frontend: http://localhost:5173
# Backend: http://localhost:8000/api/projects/

# 4) Crie superuser p/ admin

```

```
docker compose exec backend python manage.py createsuperuser
# Painel: http://localhost:8000/admin
```

---

## Deploy (sugestão rápida)

- **Frontend (Netlify):** `npm run build` → publicar pasta `dist/`. Configure `VITE_API_BASE` apontando para seu backend público.
  - **Backend (Railway/Render/Fly.io):** defina `DATABASE_URL`, `ALLOWED_HOSTS`, `CORS_ALLOWED_ORIGINS` (inclua o domínio do Netlify) e sirva com `gunicorn` (adicione no Procfile se quiser).
- 

## Próximos incrementos (quando der tempo)

1. **Autenticação simples** (admin-only) para endpoint de contato.
2. **Uploads de imagem** (S3/Cloudinary) para `cover_url`.
3. **Sitemap + OpenGraph** para SEO.
4. **Página de "Linha do Tempo"** agregando projetos/achievements.
5. **Dark/Light toggle** (já está escuro por padrão).

Minimalista agora, expansível depois. Centraliza software, hardware, pesquisa e competições sem dor de cabeça.