**Guia — Projeto Django de Análise Fundamentalista (Starter)**

Este guia foi preparado para o Silvio Luiz Puzzovio. Contém um passo a passo para rodar o starter em Windows/macOS/Linux, explicações didáticas das Class-Based Views (ListView, DetailView, CreateView, UpdateView, DeleteView) e o desenho do modelo de dados com ERD.

# 1) O que vem no pacote

• Estrutura Django completa (settings/urls/asgi/wsgi).  
• App core com models, views, forms, urls e admin.  
• Templates com Bootstrap.  
• README com instruções.  
• Diagrama ER (Mermaid) em docs/er\_diagram.md.

# 2) Como rodar no seu PC

## 2.1 Pré-requisitos

• Python 3.10+ instalado (verifique com: python --version).  
• Pip atualizado: python -m pip install --upgrade pip.

## 2.2 Passo a passo — Windows (PowerShell)

|  |
| --- |
| # 1) Descompacte o zip cd .\django-fundamentalista-starter  # 2) Crie e ative o ambiente virtual python -m venv .venv .\.venv\Scripts\Activate.ps1  # 3) Instale dependências e inicialize o banco pip install -r requirements.txt python manage.py migrate  # 4) Crie um usuário admin (para acessar /admin) python manage.py createsuperuser  # 5) Rode o servidor python manage.py runserver |

## 2.3 Passo a passo — macOS / Linux (bash/zsh)

|  |
| --- |
| # 1) Descompacte o zip cd django-fundamentalista-starter  # 2) Crie e ative o ambiente virtual python -m venv .venv source .venv/bin/activate  # 3) Instale dependências e inicialize o banco pip install -r requirements.txt python manage.py migrate  # 4) Crie um usuário admin (para acessar /admin) python manage.py createsuperuser  # 5) Rode o servidor python manage.py runserver |

Abra: http://127.0.0.1:8000 — As operações de criação/edição/remoção exigem login; já configurei LOGIN\_URL = /admin/login/.

## 2.4 Estrutura de pastas

|  |
| --- |
| django-fundamentalista-starter/ ├─ manage.py ├─ requirements.txt ├─ fundamentus/ │ ├─ settings.py │ ├─ urls.py │ ├─ asgi.py │ └─ wsgi.py ├─ core/ │ ├─ models.py │ ├─ views.py │ ├─ urls.py │ ├─ forms.py │ ├─ admin.py │ └─ migrations/ ├─ templates/ │ ├─ base.html │ └─ core/ │ ├─ company\_list.html │ ├─ company\_detail.html │ ├─ company\_form.html │ └─ company\_confirm\_delete.html └─ docs/  └─ er\_diagram.md |

# 3) CBVs — explicação e exemplos

O código está extensivamente comentado em core/views.py. Abaixo, trechos didáticos:

## 3.1 ListView — lista com filtro e paginação

|  |
| --- |
| from django.views.generic import ListView from .models import Company  class CompanyListView(ListView):  model = Company  template\_name = 'core/company\_list.html'  context\_object\_name = 'companies'  paginate\_by = 10   def get\_queryset(self):  qs = super().get\_queryset().select\_related('exchange', 'sector', 'industry', 'currency')  q = self.request.GET.get('q')  if q:  qs = qs.filter(name\_\_icontains=q) | qs.filter(ticker\_\_icontains=q)  return qs.order\_by('name')   def get\_context\_data(self, \*\*kwargs):  ctx = super().get\_context\_data(\*\*kwargs)  ctx['q'] = self.request.GET.get('q', '')  return ctx |

## 3.2 DetailView — detalhamento de um registro

|  |
| --- |
| from django.views.generic import DetailView from .models import Company  class CompanyDetailView(DetailView):  model = Company  template\_name = 'core/company\_detail.html'  context\_object\_name = 'company'   def get\_context\_data(self, \*\*kwargs):  ctx = super().get\_context\_data(\*\*kwargs)  company = self.object  ctx['latest\_facts'] = getattr(company, 'facts', None)  ctx['indicator\_values'] = getattr(company, 'indicator\_values', None)  return ctx |

## 3.3 CreateView — criação com ModelForm

|  |
| --- |
| from django.views.generic import CreateView from django.urls import reverse\_lazy from .forms import CompanyForm from .models import Company  class CompanyCreateView(CreateView):  model = Company  form\_class = CompanyForm  template\_name = 'core/company\_form.html'  success\_url = reverse\_lazy('core:company\_list')   def form\_valid(self, form):  obj = form.save(commit=False)  if obj.ticker:  obj.ticker = obj.ticker.upper().strip()  obj.save()  return super().form\_valid(form) |

## 3.4 UpdateView — edição reaproveitando o formulário

|  |
| --- |
| from django.views.generic import UpdateView from .forms import CompanyForm from .models import Company  class CompanyUpdateView(UpdateView):  model = Company  form\_class = CompanyForm  template\_name = 'core/company\_form.html'  success\_url = reverse\_lazy('core:company\_list') |

## 3.5 DeleteView — confirmação e exclusão

|  |
| --- |
| from django.views.generic import DeleteView from .models import Company  class CompanyDeleteView(DeleteView):  model = Company  template\_name = 'core/company\_confirm\_delete.html'  success\_url = reverse\_lazy('core:company\_list') |

# 4) Modelo de dados

Entidades: Currency, Exchange, Sector/Industry, Company, DataSource, Filing, FinancialStatement, Account, Fact, IndicatorDefinition, IndicatorValue, Price, CorporateAction, FXRate, SourceMapping, ImportJob.

Pontos-chave: plano de contas normalizado (Account), fatos atômicos por período (Fact), indicadores derivados por fórmula (IndicatorDefinition → IndicatorValue), séries de preços diárias (Price) e rastreabilidade de importação (ImportJob).

## 4.1 ERD (Mermaid)

|  |
| --- |
| erDiagram  Currency ||--o{ Company : moeda  Exchange ||--o{ Company : lista  Sector ||--o{ Industry : tem  Industry ||--o{ Company : classifica  Company ||--o{ Filing : entrega  DataSource ||--o{ Filing : origina  Filing ||--o{ FinancialStatement : compoe  FinancialStatement ||--o{ Fact : contem  Account ||--o{ Fact : referencia  Company ||--o{ IndicatorValue : calcula  IndicatorDefinition ||--o{ IndicatorValue : define  Company ||--o{ Price : possui  Company ||--o{ CorporateAction : evento  Currency ||--o{ FXRate : base  Currency ||--o{ FXRate : quote  DataSource ||--o{ SourceMapping : mapeia |

# 5) Próximos passos recomendados

1) Importadores CVM/SEC/APIs via management commands.  
2) Serviço de cálculo e cache de indicadores (signals pós-importação).  
3) Regras de permissão e perfis de acesso para CRUD.  
4) Trocar SQLite por PostgreSQL em produção.  
5) Testes automatizados para fórmulas e integridade de dados.

# 6) Observações

• Locale padrão: pt-br / Timezone: America/Sao\_Paulo.  
• LOGIN\_URL = /admin/login/ para facilitar testes de CRUD autenticado.  
• Templates baseados em Bootstrap 5.3.3.