Tutorial: Como Instalar e Executar uma Aplicação To-Do com Django e React

Bem-vindo(a) a este tutorial detalhado! Aqui, você aprenderá a configurar, instalar e executar uma aplicação completa de lista de tarefas (to-do list) que utiliza Django como backend e React com Vite como frontend. Este guia foi elaborado para ser acessível a usuários com diferentes níveis de experiência técnica, fornecendo instruções claras e comandos que podem ser copiados e colados diretamente no seu terminal.

Ao final deste tutorial, você terá uma compreensão sólida de como esses dois frameworks trabalham juntos e como colocar sua aplicação em funcionamento.

1. Requisitos de Sistema

Para garantir uma instalação e execução suaves da sua aplicação, é fundamental que seu ambiente de desenvolvimento atenda aos seguintes requisitos. Certifique-se de ter as versões especificadas ou mais recentes para evitar problemas de compatibilidade.

1.1. Versões Necessárias de Software

- **Python:** Versão 3.9 ou superior. Recomendamos a versão 3.10 ou 3.11 para melhor compatibilidade com as bibliotecas mais recentes do Django.
- **Node.js:** Versão 16.x ou superior. A versão 18.x (LTS) é altamente recomendada para o desenvolvimento com React e Vite.
- **npm (Node Package Manager):** Geralmente vem junto com a instalação do Node.js. Certifique-se de ter a versão 8.x ou superior.
- **Git:** Para clonar o repositório da aplicação. Se você ainda não o tem, pode baixá-lo em https://git-scm.com/downloads.

1.2. Dependências que Precisam Ser Instaladas Previamente

Antes de começar, verifique se você tem os seguintes itens instalados em seu sistema operacional:

- **Um editor de código:** Recomendamos o Visual Studio Code (VS Code) devido ao seu excelente suporte para Python, JavaScript e ferramentas de desenvolvimento web. Você pode baixá-lo em https://code.visualstudio.com/download.
- **Um terminal:** Você usará o terminal para executar todos os comandos. No Windows, pode ser o PowerShell, o Git Bash ou o terminal integrado do VS Code. No macOS e Linux, o terminal padrão é suficiente.

Para verificar as versões do Python e Node.js instaladas, abra seu terminal e execute os seguintes comandos:

```
 python3 --version
node --version
npm --version
```

Se as versões não atenderem aos requisitos, por favor, atualize-as antes de prosseguir. Para Python, você pode usar ferramentas como `pyenv` ou `conda`. Para Node.js, `nvm` é uma ótima opção para gerenciar múltiplas versões.

2. Configuração do Ambiente de Desenvolvimento

Com os requisitos de sistema atendidos, o próximo passo é configurar seu ambiente de desenvolvimento. Isso envolve obter o código-fonte da aplicação e preparar os ambientes Python e Node.js para o backend e frontend, respectivamente.

2.1. Como Clonar ou Baixar o Código-Fonte

Assumimos que o código-fonte da sua aplicação de lista de tarefas está hospedado em um repositório Git (por exemplo, GitHub, GitLab, Bitbucket). Se você tem o código em um arquivo ZIP, descompacte-o em uma pasta de sua preferência e pule o comando `git clone`.

1. **Abra seu terminal** e navegue até o diretório onde você deseja armazenar o projeto. Por exemplo:

```bash
cd ~/Documentos/Projetos
```

2. **Clone o repositório** da sua aplicação. Substitua `[URL_DO_SEU_REPOSITORIO]` pela URL real do seu repositório Git:

   ```bash
   git clone [URL_DO_SEU_REPOSITORIO]
   ```

3. **Navegue para o diretório do projeto clonado**:

   ```bash
   cd nome-do-seu-projeto # Substitua pelo nome da pasta criada pelo git clone
   ```

   Dentro desta pasta, você deve encontrar duas subpastas principais: uma para o backend (Django) e outra para o frontend (React). Vamos chamá-las de `backend` e `frontend` para este tutorial.

2.2. Como Configurar o Ambiente Virtual Python para o Backend

É uma boa prática isolar as dependências do seu projeto Python usando um ambiente virtual. Isso evita conflitos entre diferentes projetos Python em sua máquina.

1. **Navegue até o diretório do backend**:

   ```bash
   cd backend
   ```

2. **Crie um ambiente virtual** chamado `venv` (ou qualquer outro nome de sua preferência):

   ```bash
   python3 -m venv venv
   ```

3. **Ative o ambiente virtual**:

   - **No macOS/Linux:**

     ```bash
     source venv/bin/activate
     ```

   - **No Windows (PowerShell):**

     ```bash
     .\venv\Scripts\Activate.ps1
     ```

   - **No Windows (CMD):**

     ```bash
     .\venv\Scripts\activate.bat
     ```

Você saberá que o ambiente virtual está ativo quando vir `(venv)` no início da linha de comando do seu terminal.

2.3. Como Configurar o Ambiente Node.js para o Frontend

O frontend React com Vite também precisa de suas próprias dependências Node.js. Não é necessário um ambiente virtual separado como no Python, pois o `npm` gerencia as dependências localmente no diretório do projeto.

1. **Navegue até o diretório do frontend** (certifique-se de sair do diretório `backend` primeiro, se estiver lá):

   ```
   bash cd .. # Para voltar para a pasta raiz do projeto cd frontend
   ```

2. **Instale as dependências do Node.js** usando `npm`. Certifique-se de que você está no diretório `frontend`:

   ```
   bash npm install
   ```

   Este comando lerá o arquivo `package.json` e instalará todas as bibliotecas necessárias na pasta `node_modules`.

Com essas etapas, seu ambiente de desenvolvimento estará configurado e pronto para a instalação das dependências específicas de cada parte da aplicação.

3. Instalação do Backend (Django)

Agora que o ambiente virtual Python está ativo, podemos prosseguir com a instalação das dependências do Django e a configuração do banco de dados.

3.1. Passos Detalhados para Instalar as Dependências do Python

1. **Certifique-se de que seu ambiente virtual Python está ativo.** Se não estiver, navegue até o diretório `backend` e ative-o conforme as instruções na Seção 2.2.

2. **Instale as dependências do Python.** Seu projeto Django deve ter um arquivo `requirements.txt` listando todas as bibliotecas necessárias. Execute o seguinte comando no diretório `backend`:

```
bash pip install -r requirements.txt
```

Este comando instalará o Django, o Django REST Framework e quaisquer outras bibliotecas Python que sua aplicação utilize.

3.2. Como Configurar o Banco de Dados

Por padrão, o Django utiliza SQLite para desenvolvimento, que é um banco de dados baseado em arquivo e não requer configuração adicional de servidor. Se sua aplicação usa PostgreSQL, MySQL ou outro banco de dados, você precisará garantir que o servidor de banco de dados esteja em execução e que as credenciais estejam configuradas corretamente no arquivo `settings.py` do seu projeto Django.

Para este tutorial, assumiremos o uso de SQLite, que é a configuração padrão e mais simples para começar.

3.3. Como Executar as Migrações

As migrações do Django são a forma como o framework gerencia as alterações no esquema do seu banco de dados. Elas criam as tabelas necessárias para seus modelos.

1. **Execute as migrações** no diretório `backend` (com o ambiente virtual ativo):

   ```
   bash python manage.py migrate
   ```

   Este comando aplicará todas as migrações pendentes, criando as tabelas do banco de dados e as tabelas padrão do Django (como usuários e sessões).

2. **Opcional: Crie um superusuário** para acessar o painel administrativo do Django (se sua aplicação o utilizar):

   ```
   bash python manage.py createsuperuser
   ```

   Siga as instruções no terminal para criar um nome de usuário, endereço de e-mail e senha.

3.4. Como Iniciar o Servidor Django

Com as dependências instaladas e o banco de dados configurado, você pode iniciar o servidor de desenvolvimento do Django.

1. **Inicie o servidor** no diretório `backend` (com o ambiente virtual ativo):

   `bash python manage.py runserver`

   Você verá uma mensagem no terminal indicando que o servidor está rodando, geralmente em `http://127.0.0.1:8000/`.

   ``` Watching for file changes with StatReloader Performing system checks...

   System check identified no issues (0 silenced).

   You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions. Run 'python manage.py migrate' to apply them. September 15, 2025 - 14:30:00 Django version 4.2.5, using settings 'your_project_name.settings' Starting development server at http://127.0.0.1:8000/ Quit the server with CONTROL-C. ```

   Mantenha este terminal aberto, pois o servidor Django precisa estar em execução para que o frontend possa se comunicar com ele. Se você precisar parar o servidor, pressione `Ctrl+C` no terminal.

# 4. Instalação do Frontend (React)

Com o backend configurado e potencialmente em execução, vamos agora configurar e iniciar o frontend React.

## 4.1. Passos Detalhados para Instalar as Dependências do Node.js

1. **Certifique-se de que você está no diretório `frontend`** do seu projeto. Se você ainda estiver no diretório `backend`, use `cd ../frontend` para navegar.

2. **Instale as dependências do Node.js** novamente, caso não tenha feito na seção de configuração do ambiente, ou para garantir que todas as dependências estão atualizadas:

   `bash npm install`

   Este comando lerá o arquivo `package.json` e instalará todas as bibliotecas e pacotes JavaScript necessários para o seu projeto React/Vite.
   ```

4.2. Como Configurar as Variáveis de Ambiente (se necessário)

Sua aplicação React pode precisar de variáveis de ambiente para se conectar ao backend ou para outras configurações específicas. Com Vite, as variáveis de ambiente são geralmente carregadas de arquivos `.env` na raiz do diretório `frontend`.

1. **Verifique se existe um arquivo `.env.example`** no diretório `frontend`. Se sim, copie-o para `.env`:

   `bash cp .env.example .env`

2. **Edite o arquivo `.env`** com as configurações corretas. A variável mais comum é a URL do backend. Por exemplo:

   `dotenv VITE_API_URL=http://127.0.0.1:8000/api/`

   Certifique-se de que a URL corresponde ao endereço onde seu servidor Django está rodando (geralmente `http://127.0.0.1:8000/`). O `/api/` no final é um exemplo e pode variar dependendo de como sua API Django está configurada.

4.3. Como Iniciar o Servidor de Desenvolvimento React

Com todas as dependências instaladas e as variáveis de ambiente configuradas, você pode iniciar o servidor de desenvolvimento do frontend.

1. **Inicie o servidor de desenvolvimento** no diretório `frontend`:

   `bash npm run dev`

   Este comando iniciará o servidor de desenvolvimento do Vite, que compilará seu código React e o servirá em um endereço local, geralmente `http://localhost:5173/` ou similar. Você verá uma saída no terminal indicando o endereço:

   ``` VITE v5.0.12 ready in 321 ms

   ➜ Local: http://localhost:5173/ ➜ Network: use --host to expose ➜ press h + enter to show help ```

   Abra seu navegador e acesse o endereço `http://localhost:5173/` (ou o endereço indicado no seu terminal). Você deverá ver a interface da sua aplicação
   ```

de lista de tarefas. Mantenha este terminal aberto, pois o servidor Vite precisa estar em execução para que o frontend funcione. Para parar o servidor, pressione `Ctrl+C`.

5. Verificação da Instalação

Após instalar e iniciar o backend e o frontend, é crucial verificar se ambos estão funcionando corretamente e se a comunicação entre eles está estabelecida.

5.1. Como Verificar se o Backend Está Funcionando Corretamente

1. **Acesse a URL do backend no navegador:** Com o servidor Django em execução (no terminal onde você executou `python manage.py runserver`), abra seu navegador e tente acessar `http://127.0.0.1:8000/`. Dependendo da configuração do seu projeto Django, você pode ver a página de boas-vindas do Django, uma página de erro 404 (se não houver uma rota configurada para a raiz) ou a interface do Django Admin (se você a configurou).

2. **Teste um endpoint da API:** Se sua aplicação possui uma API REST, tente acessar um endpoint específico. Por exemplo, se você tem um endpoint para listar tarefas em `http://127.0.0.1:8000/api/tasks/`, tente acessá-lo. Você pode usar o navegador ou ferramentas como `curl` ou Postman para isso.

   `bash curl http://127.0.0.1:8000/api/tasks/`

   Você deve receber uma resposta JSON (provavelmente uma lista vazia ou com algumas tarefas de exemplo, se houver).

3. **Verifique os logs do terminal:** Observe o terminal onde o servidor Django está rodando. Requisições bem-sucedidas geralmente aparecem com um código de status `200 OK`. Erros (como `404 NOT FOUND` ou `500 INTERNAL SERVER ERROR`) indicarão problemas.

5.2. Como Verificar se o Frontend Está Funcionando Corretamente

1. **Acesse a URL do frontend no navegador:** Com o servidor Vite em execução (no terminal onde você executou `npm run dev`), abra seu navegador e acesse

`http://localhost:5173/` (ou o endereço indicado). Você deve ver a interface gráfica da sua aplicação de lista de tarefas.

2. **Interaja com a interface:** Tente adicionar uma nova tarefa, marcar uma como concluída ou excluí-la. Observe se a interface responde conforme o esperado.

3. **Verifique o console do navegador:** Abra as ferramentas de desenvolvedor do seu navegador (geralmente `F12` ou `Ctrl+Shift+I`) e vá para a aba

Console`. Verifique se há erros JavaScript ou mensagens de aviso. Isso pode indicar problemas no código do frontend.

5.3. Como Testar a Integração entre Frontend e Backend

Este é o passo mais importante para garantir que sua aplicação está funcionando como um todo:

1. **Com ambos os servidores (Django e Vite) em execução**, acesse o frontend em `http://localhost:5173/`.

2. **Tente realizar uma operação que envolva comunicação com o backend.** Por exemplo, adicione uma nova tarefa à lista. Se a tarefa aparecer na lista e persistir após recarregar a página (ou se você puder vê-la acessando o endpoint da API do backend diretamente), a integração está funcionando.

3. **Observe os logs de ambos os terminais:**

    - No terminal do Django, você deve ver as requisições HTTP (POST, GET, PUT, DELETE) que o frontend está fazendo para a API.
    - No console do navegador (ferramentas de desenvolvedor), você pode ver as requisições de rede na aba `Network` e quaisquer erros de comunicação (como erros CORS ou falha na requisição).

Se todas essas verificações forem bem-sucedidas, parabéns! Sua aplicação Django e React está instalada e funcionando corretamente.

6. Solução de Problemas Comuns

Mesmo seguindo todos os passos, é comum encontrar alguns problemas. Aqui estão alguns dos mais frequentes e como resolvê-los.

6.1. Erros Comuns Durante a Instalação e Como Resolvê-los

- **`Command 'python3' not found`** ou **`Command 'node' not found`:** Isso significa que Python ou Node.js não estão instalados ou não estão no PATH do seu sistema. Revise a Seção 1.1 e certifique-se de que ambos estão instalados corretamente e acessíveis via terminal.
- **`pip is not recognized as an internal or external command`** **(Windows):** Isso geralmente indica que o Python não foi adicionado ao PATH durante a instalação. Você pode precisar reinstalar o Python e marcar a opção "Add Python to PATH" ou adicionar manualmente.
- **`ModuleNotFoundError: No module named 'django'`:** Isso ocorre se você não ativou o ambiente virtual Python antes de instalar as dependências ou se as dependências não foram instaladas corretamente. Certifique-se de que `(venv)` aparece no seu prompt de comando e execute `pip install -r requirements.txt` novamente.
- **`npm ERR! code ERESOLVE`** ou **`npm ERR! EACCES: permission denied`:** Erros de permissão ao instalar pacotes npm. Tente executar `npm install` com `sudo` (em Linux/macOS) ou como administrador (em Windows), ou corrija as permissões da pasta `node_modules` e do cache npm. Para `ERESOLVE`, pode ser um conflito de dependências; tente `npm install --force` (com cautela) ou `npm install --legacy-peer-deps`.
- **Erro `500 Internal Server Error` no Django:** Verifique o terminal onde o servidor Django está rodando. Ele geralmente fornecerá um traceback detalhado que aponta para a causa do erro, como um problema de configuração, erro de código ou banco de dados.

6.2. Problemas de Conexão entre Frontend e Backend

- **Erro CORS (Cross-Origin Resource Sharing):** Este é um dos problemas mais comuns ao integrar frontend e backend em domínios/portas diferentes. O navegador bloqueia requisições do frontend para o backend por segurança. No Django, você precisará instalar e configurar a biblioteca `django-cors-headers`.

    1. **Instale `django-cors-headers`:**

        `bash pip install django-cors-headers`

    2. **Adicione ao `INSTALLED_APPS` no `settings.py` do Django:**

        ```python

# your_project_name/settings.py

INSTALLED_APPS = [ # ... 'corsheaders', # ... ] ```

    3. **Adicione `CorsMiddleware` ao `MIDDLEWARE` no `settings.py`:** Deve vir antes de qualquer outro middleware que possa gerar respostas.

        ```python

# your_project_name/settings.py

MIDDLEWARE = [ 'corsheaders.middleware.CorsMiddleware', # ... outros middlewares ] ```

    4. **Configure `CORS_ALLOWED_ORIGINS` ou `CORS_ALLOW_ALL_ORIGINS` no `settings.py`:**

        Para permitir requisições de um domínio específico (seu frontend):

        ```python
        ```

your_project_name/settings.py

CORS_ALLOWED_ORIGINS = ['http://localhost:5173', # A porta do seu frontend Vite # Adicione outros domínios se necessário] ```

Ou, para permitir todas as origens (menos seguro, mas útil para desenvolvimento rápido):

```python

# your_project_name/settings.py

CORS_ALLOW_ALL_ORIGINS = True ```

Após essas alterações, reinicie o servidor Django.

- **`Failed to fetch` ou `Network Error` no frontend:** Isso geralmente indica que o frontend não conseguiu se conectar ao backend. Verifique:
  - Se o servidor Django está realmente em execução.
  - Se a `VITE_API_URL` no seu arquivo `.env` do frontend está correta e aponta para o endereço e porta certos do backend.
  - Se não há um firewall bloqueando a comunicação entre as portas.
- **Erro `404 Not Found` para endpoints da API:** Verifique as URLs dos seus endpoints no frontend e as configurações de rotas (`urls.py`) no backend Django. Certifique-se de que as URLs correspondem exatamente.

## 7. Próximos Passos

Com sua aplicação funcionando, aqui estão algumas ideias para continuar seu desenvolvimento:
```

7.1. Como Personalizar a Aplicação

- **Modifique o código:** Explore os arquivos do projeto. No backend (Django), você pode adicionar novos modelos, views, serializers e URLs. No frontend (React), você pode alterar componentes, estilos e adicionar novas funcionalidades.
- **Estilo e UI/UX:** Personalize a aparência da sua aplicação. Experimente com bibliotecas de componentes (como Material UI, Ant Design) ou frameworks CSS (Tailwind CSS, Bootstrap).
- **Novas funcionalidades:** Adicione recursos como autenticação de usuário, filtros de tarefas, categorias, prazos, etc.

7.2. Como Implantar em Produção

Implantar uma aplicação em produção envolve mais etapas do que apenas executá-la localmente. Aqui estão os passos gerais:

1. **Configuração de Produção do Django:**
    - Defina `DEBUG = False` no `settings.py`.
    - Configure `ALLOWED_HOSTS` com os domínios da sua aplicação.
    - Configure um banco de dados de produção (PostgreSQL, MySQL).
    - Configure a coleta de arquivos estáticos (`STATIC_ROOT` , `STATIC_URL`).
    - Use um servidor de aplicação como Gunicorn ou uWSGI.
    - Use um servidor web como Nginx ou Apache para servir arquivos estáticos e atuar como proxy reverso para o Gunicorn/uWSGI.
2. **Build de Produção do React:**
    - No diretório `frontend` , execute `npm run build` . Isso criará uma versão otimizada e estática do seu frontend na pasta `dist` (ou similar).
    - Esses arquivos estáticos podem ser servidos pelo mesmo servidor web (Nginx/Apache) que serve o backend Django, ou por um serviço de hospedagem de frontend separado (como Netlify, Vercel).
3. **Escolha um Provedor de Hospedagem:**

- **Para o Backend (Django):** Heroku, DigitalOcean, AWS, Google Cloud, PythonAnywhere, Render.
- **Para o Frontend (React):** Netlify, Vercel, GitHub Pages, Firebase Hosting.

4. **Configuração de Variáveis de Ambiente:** Em produção, as variáveis de ambiente devem ser configuradas de forma segura no seu provedor de hospedagem, e não diretamente no código ou em arquivos `.env` versionados.

Este tutorial forneceu a base para você começar. Aprofunde-se na documentação do Django e do React para explorar todo o potencial dessas tecnologias e construir aplicações incríveis!