Geração de Projetos - Modelo

Você pode usar um gerador de projetos para começar, por já incluir configurações iniciais, segurança, banco de dados e os primeiros *endpoints* API já feitos para você.

Um gerador de projetos sempre terá uma pré-configuração que você pode atualizar e adaptar para suas próprias necessidades, mas pode ser um bom ponto de partida para seu projeto.

Full Stack FastAPI PostgreSQL

GitHub: https://github.com/tiangolo/full-stack-fastapi-postgresql

Full Stack FastAPI PostgreSQL - Recursos

- Integração completa **Docker**.
- Modo de implantação Docker Swarm.
- Integração e otimização **Docker Compose** para desenvolvimento local.
- **Pronto para Produção** com servidor *web* usando Uvicorn e Gunicorn.
- *Backend* **FastAPI** Python:
    - **Rápido**: Alta performance, no nível de **NodeJS** e **Go** (graças ao Starlette e Pydantic).
    - **Intuitivo**: Ótimo suporte de editor. *Auto-Complete* em todo lugar. Menos tempo *debugando*.
    - **Fácil**: Projetado para ser fácil de usar e aprender. Menos tempo lendo documentações.
    - **Curto**: Minimize duplicação de código. Múltiplos recursos para cada declaração de parâmetro.
    - **Robusto**: Tenha código pronto para produção. Com documentação interativa automática.
    - **Baseado em Padrões**: Baseado em (e completamente compatível com) padrões abertos para APIs: OpenAPI e JSON Schema.
    - **Muitos outros recursos** incluindo validação automática, serialização, documentação interativa, autenticação com *tokens* OAuth2 JWT etc.
- **Senha segura** *hashing* por padrão.
- Autenticação **Token JWT**.
- Modelos **SQLAlchemy** (independente de extensões Flask, para que eles possam ser usados com *workers* Celery diretamente).
- Modelos básicos para usuários (modifique e remova conforme suas necessidades).
- Migrações **Alembic**.
- **CORS** (*Cross Origin Resource Sharing* - Compartilhamento de Recursos Entre Origens).
- *Worker* **Celery** que pode importar e usar modelos e códigos do resto do *backend* seletivamente.
- Testes *backend REST* baseados no **Pytest**, integrados com Docker, então você pode testar a interação completa da API, independente do banco de dados. Como roda no Docker, ele pode construir um novo repositório de dados do zero toda vez (assim você pode usar ElasticSearch, MongoDB, CouchDB, ou o que quiser, e apenas testar que a API esteja funcionando).
- Fácil integração com Python através dos **Kernels Jupyter** para desenvolvimento remoto ou no Docker com extensões como Atom Hydrogen ou Visual Studio Code Jupyter.
- *Frontend* **Vue**:
    - Gerado com Vue CLI.
    - Controle de **Autenticação JWT**.
    - Visualização de *login*.
    - Após o *login*, visualização do painel de controle principal.
    - Painel de controle principal com criação e edição de usuário.
    - Edição do próprio usuário.
    - **Vuex**.
    - **Vue-router**.

    - **Vuetify** para belos componentes *material design*.
    - **TypeScript**.
    - Servidor Docker baseado em **Nginx** (configurado para rodar "lindamente" com Vue-router).
    - Construção multi-estágio Docker, então você não precisa salvar ou *commitar* código compilado.
    - Testes *frontend* rodados na hora da construção (pode ser desabilitado também).
    - Feito tão modular quanto possível, então ele funciona fora da caixa, mas você pode gerar novamente com Vue CLI ou criar conforme você queira, e reutilizar o que quiser.
- **PGAdmin** para banco de dados PostgreSQL, você pode modificar para usar PHPMyAdmin e MySQL facilmente.
- **Flower** para monitoração de tarefas Celery.
- Balanceamento de carga entre *frontend* e *backend* com **Traefik**, então você pode ter ambos sob o mesmo domínio, separados por rota, mas servidos por diferentes containers.
- Integração Traefik, incluindo geração automática de certificados **HTTPS** Let's Encrypt.
- GitLab **CI** (integração contínua), incluindo testes *frontend* e *backend*.

Full Stack FastAPI Couchbase

GitHub: https://github.com/tiangolo/full-stack-fastapi-couchbase

⚠ **WARNING** ⚠

Se você está iniciando um novo projeto do zero, verifique as alternativas aqui.

Por exemplo, o gerador de projetos Full Stack FastAPI PostgreSQL pode ser uma alternativa melhor, como ele é ativamente mantido e utilizado. E ele inclui todos os novos recursos e melhorias.

Você ainda é livre para utilizar o gerador baseado em Couchbase se quiser, ele provavelmente ainda funciona bem, e você já tem um projeto gerado com ele que roda bem também (e você provavelmente já atualizou ele para encaixar nas suas necessidades).

Você pode ler mais sobre nas documentaçãoes do repositório.

Full Stack FastAPI MongoDB

...pode demorar, dependendo do meu tempo disponível e outros fatores. 😅 🎉

Modelos de Aprendizado de Máquina com spaCy e FastAPI

GitHub: https://github.com/microsoft/cookiecutter-spacy-fastapi

Modelos de Aprendizado de Máquina com spaCy e FastAPI - Recursos

- Integração com modelo NER **spaCy**.
- Formato de requisição **Busca Cognitiva Azure** acoplado.
- Servidor Python *web* **Pronto para Produção** usando Uvicorn e Gunicorn.
- Implantação **Azure DevOps** Kubernetes (AKS) CI/CD acoplada.
- **Multilingual** facilmente escolhido como uma das linguagens spaCy acopladas durante a configuração do projeto.
- **Facilmente extensível** para outros modelos de *frameworks* (Pytorch, Tensorflow), não apenas spaCy.