**Visão Geral da Nova Arquitetura**

Vamos usar uma combinação poderosa:

1. **Backend (API):** Python com o framework **FastAPI**. Ele é extremamente rápido, moderno e tem documentação interativa automática (Swagger UI), o que é perfeito para desenvolver e testar sua API.
2. **Banco de Dados:** **MySQL**. Um sistema de gerenciamento de banco de dados relacional (SGBDR) robusto e amplamente utilizado.
3. **Ambiente e Orquestração:** **Docker** e **Docker Compose**. O Docker vai criar contêineres isolados para sua aplicação FastAPI e para o banco de dados MySQL. O Docker Compose vai orquestrar esses contêineres, fazendo com que "conversem" entre si e subam com um único comando.
4. **Versionamento:** **Git** e **GitHub**. Para controlar as versões do seu código e criar um portfólio.

**Passo a Passo Detalhado: Preparando o Ambiente**

Siga estas etapas na ordem correta para configurar todo o seu ambiente de desenvolvimento.

**Passo 0: Instalação das Ferramentas Essenciais (Pré-requisitos)**

Antes de começar a programar, você precisa ter estas ferramentas instaladas na sua máquina.

1. **Git:**
   * **O que é?** Sistema de controle de versão para rastrear as alterações no seu código.
   * **Como baixar?** Acesse [git-scm.com](https://git-scm.com/downloads) e baixe o instalador para o seu sistema operacional (Windows, macOS ou Linux). Durante a instalação, pode manter as opções padrão.
2. **Docker Desktop:**
   * **O que é?** A ferramenta que gerencia seus contêineres, imagens e o Docker Compose.
   * **Como baixar?** Acesse [docker.com/products/docker-desktop/](https://www.docker.com/products/docker-desktop/). Baixe a versão para seu sistema (Windows ou macOS). Para Linux, você precisará instalar o Docker Engine e o Docker Compose separadamente (siga o guia oficial do site).
   * **Importante (Windows):** O Docker Desktop pode exigir que o WSL 2 (Windows Subsystem for Linux 2) esteja ativado. O próprio instalador do Docker geralmente guia você por esse processo. Após instalar, **inicie o Docker Desktop** e espere ele indicar que está rodando.
3. **Visual Studio Code (VS Code):**
   * **O que é?** Um editor de código leve e poderoso. É a nossa recomendação, mas você pode usar outro de sua preferência.
   * **Como baixar?** Acesse [code.visualstudio.com](https://code.visualstudio.com/) e instale.
   * **Extensões Recomendadas (dentro do VS Code):**
     + Python (da Microsoft): Essencial para desenvolvimento Python.
     + Docker (da Microsoft): Ajuda a gerenciar contêineres e arquivos Docker diretamente do editor.

**Passo 1: Estrutura do Projeto e Configuração do Git/GitHub**

1. **Crie a Pasta do Projeto:** No seu computador, crie uma pasta principal para o projeto. Ex: sistema-despesas.
2. **Abra no VS Code:** Abra o VS Code e vá em File > Open Folder... e selecione a pasta sistema-despesas.
3. **Inicie o Git:** Abra um terminal dentro do VS Code (Terminal > New Terminal). Digite o comando:

git init

Isso cria um repositório Git local na sua pasta.

1. **Crie o Repositório no GitHub:**

* Acesse seu perfil no [github.com](https://github.com).
* Clique em New para criar um novo repositório.
* Dê um nome (ex: sistema-controle-despesas), marque-o como Public e **não** adicione README, .gitignore ou licença ainda. Clique em Create repository.
* O GitHub lhe dará os comandos para conectar seu repositório local. Copie e cole no seu terminal do VS Code:

Bash

git remote add origin https://github.com/SEU\_USUARIO/sistema-controle-despesas.git

git branch -M main

**Passo 2: "Dockerizando" o Ambiente com Docker Compose**

Agora, vamos dizer ao Docker como construir e executar nosso sistema. Faremos isso com dois arquivos principais.

1. **Crie o Dockerfile:** Este arquivo é a "receita" para construir a imagem da sua aplicação Python/FastAPI.
   * Na raiz da pasta sistema-despesas, crie um arquivo chamado Dockerfile (sem extensão).
   * Cole o seguinte conteúdo nele:

Dockerfile

1. # 1. Usar uma imagem oficial do Python como base
2. FROM python:3.11-slim
3. # 2. Definir o diretório de trabalho dentro do contêiner
4. WORKDIR /app
5. # 3. Copiar o arquivo de dependências para o contêiner
6. # Copiamos primeiro para aproveitar o cache do Docker. Se as dependências não mudam,
7. # o Docker não precisa reinstalar tudo a cada build.
8. COPY requirements.txt .
9. # 4. Instalar as dependências
10. RUN pip install --no-cache-dir -r requirements.txt
11. # 5. Copiar todo o resto do código do projeto para o diretório de trabalho
12. COPY . .
13. # 6. Expor a porta 8000, que é a porta padrão do FastAPI
14. EXPOSE 8000
15. # 7. Comando para iniciar a aplicação quando o contêiner for executado
16. # uvicorn: servidor que roda o FastAPI
17. # main:app: arquivo main.py, objeto app
18. # --host 0.0.0.0: permite que a aplicação seja acessível de fora do contêiner
19. # --port 8000: porta que o servidor vai ouvir
20. # --reload: reinicia o servidor automaticamente quando o código é alterado (ótimo para desenvolvimento)
21. CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000", "--reload"]

**2. Crie o docker-compose.yml:** Este arquivo orquestra (gerencia) múltiplos contêineres: nossa aplicação (app) e nosso banco de dados (db).

* Na raiz do projeto, crie um arquivo chamado docker-compose.yml.
* Cole o seguinte conteúdo:

YAML

version: '3.8'

services:

# Serviço da nossa aplicação FastAPI

app:

build: . # Constrói a imagem usando o Dockerfile na pasta atual

container\_name: fastapi\_app # Nome do contêiner

ports:

- "8000:8000" # Mapeia a porta 8000 do contêiner para a porta 8000 da sua máquina

volumes:

- .:/app # Mapeia a pasta atual do projeto para a pasta /app dentro do contêiner

# Isso permite que alterações no código local reflitam no contêiner instantaneamente (hot-reload)

environment:

# Variáveis de ambiente que a aplicação usará para conectar ao banco

- DATABASE\_URL=mysql+pymysql://user:password@db:3306/expenses\_db

depends\_on:

- db # Diz ao Docker para iniciar o serviço 'db' antes de iniciar o 'app'

# Serviço do nosso banco de dados MySQL

db:

image: mysql:8.0 # Usa uma imagem oficial do MySQL

container\_name: mysql\_db # Nome do contêiner

ports:

- "3307:3306" # Mapeia a porta 3306 do contêiner para a 3307 da sua máquina

# Usamos 3307 para não conflitar com uma possível instalação local do MySQL

environment:

# Variáveis de ambiente para configurar o MySQL na primeira inicialização

MYSQL\_ROOT\_PASSWORD: rootpassword # Senha do usuário root

MYSQL\_DATABASE: expenses\_db # Cria um banco de dados com este nome

MYSQL\_USER: user # Cria um usuário

MYSQL\_PASSWORD: password # Define a senha para este usuário

volumes:

- mysql\_data:/var/lib/mysql # Persiste os dados do banco em um volume Docker

# Mesmo que o contêiner seja removido, os dados não serão perdidos

volumes:

mysql\_data: # Define o volume nomeado para persistência dos dados

**Crie o requirements.txt:** Este arquivo lista as bibliotecas Python que nosso projeto precisa.

* Na raiz do projeto, crie o arquivo requirements.txt.
* Adicione as dependências iniciais:

Plaintext

fastapi

uvicorn[standard]

sqlalchemy

pymysql

**Explicação:**

* + fastapi: O framework web.
  + uvicorn: O servidor que executará a aplicação.
  + sqlalchemy: O ORM (Object Relational Mapper) que usaremos para conversar com o MySQL de forma "pythonica", sem escrever SQL puro.
  + pymysql: O "driver" que permite ao SQLAlchemy se conectar com o MySQL.

1. **Crie o .gitignore:** Este arquivo diz ao Git quais arquivos e pastas ignorar. É uma boa prática essencial.

* Na raiz do projeto, crie o arquivo .gitignore.
* Adicione o seguinte conteúdo:

# Python

\_\_pycache\_\_/

\*.pyc

.env

venv/

\*.egg-info/

# Docker

docker-compose.override.yml

# IDEs

.vscode/

.idea/

**Passo 3: Executando o Ambiente Pela Primeira Vez**

Com os arquivos de configuração prontos, vamos "subir" nosso ambiente.

1. **Abra o terminal** na raiz do projeto (onde está o docker-compose.yml).
2. **Execute o comando:**

docker-compose up –build

--build: Força o Docker a construir a imagem da sua aplicação (app) a partir do Dockerfile. Você só precisa usar --build na primeira vez ou quando alterar o Dockerfile ou requirements.txt.

up: Cria e inicia os contêineres definidos no docker-compose.yml.

1. **O que esperar:** O Docker vai baixar a imagem do Python, a do MySQL, instalar suas dependências e iniciar os dois serviços. Você verá muitos logs no seu terminal, tanto da aplicação quanto do banco.

**Se tudo deu certo, seu ambiente está PRONTO!** Você tem:

* Um contêiner rodando sua (ainda inexistente) aplicação FastAPI, acessível em http://localhost:8000.
* Um contêiner rodando o MySQL, com seus dados persistidos.

Para parar os contêineres, pressione Ctrl + C no terminal. Para iniciá-los novamente, basta usar docker-compose up.