**🧭 Fase 1: Planejamento e Definição de Funcionalidades**

**Objetivo:** Definir claramente o que o programa fará.

📌 Estude:

* Conceitos de desempenho em sistemas operacionais (uso de CPU, RAM, disco, etc.)
* Comandos e ferramentas do Windows/Linux para diagnóstico e manutenção

✅ Defina funcionalidades como:

* Limpeza de arquivos temporários
* Otimização de inicialização (programas que iniciam com o sistema)
* Verificação de integridade de disco
* Monitoramento de temperatura e uso de hardware
* Recomendações baseadas no estado do sistema
* Interface gráfica amigável

**🧠 Fase 2: Escolha de Linguagens e Ferramentas**

**Objetivo:** Decidir como o projeto será construído.

📌 Estude:

* **Python** (recomendado): Fácil de usar e ótimo para automação
* Bibliotecas Python úteis:
  + psutil – para estatísticas de CPU, RAM, disco
  + subprocess – para executar comandos do sistema
  + tkinter, PyQt ou customtkinter – para criar interfaces gráficas
  + platform, os, shutil – para interações com o sistema operacional
  + pywin32 (para Windows) – controle avançado no Windows

**🔧 Fase 3: Desenvolvimento de Funcionalidades Núcleo**

**Objetivo:** Criar as funções básicas do otimizador.

✅ Desenvolva:

* 🧹 Limpeza de arquivos temporários
* 🚫 Desativação de programas de inicialização desnecessários
* 📊 Monitoramento em tempo real do hardware
* 🛠 Verificação de atualizações do sistema
* ⚙️ Ajustes em configurações do sistema para performance

📌 Estude:

* Como usar agendadores de tarefas no Windows (schtasks)
* Como ler e escrever no Registro do Windows (com cuidado!)
* Permissões de administrador

**🖥 Fase 4: Interface Gráfica**

**Objetivo:** Tornar o projeto acessível a qualquer usuário.

✅ Faça:

* Tela principal com status do sistema
* Botões de ações rápidas (limpar, otimizar, analisar, etc.)
* Modo escuro/claro
* Mensagens de recomendação automáticas

📌 Estude:

* tkinter, customtkinter ou PyQt para a interface
* Design UI/UX simples e funcional
* Como empacotar o programa (com pyinstaller)

**📦 Fase 5: Distribuição e Repositório GitHub**

**Objetivo:** Permitir que outras pessoas usem o projeto.

✅ Faça:

* Gerador de .exe com pyinstaller ou cx\_Freeze
* Repositório GitHub bem documentado
* README com instruções de uso, imagens e funcionalidades
* Sistema de atualizações (verificação de novas versões)

📌 Estude:

* Markdown para documentação
* Como criar releases no GitHub
* Como fazer update checker no app

**🚀 Fase 6: Recursos Avançados (Extras)**

**Objetivo:** Deixar o projeto mais poderoso.

✅ Considere adicionar:

* Benchmark do sistema (teste de desempenho)
* Notificações automáticas sobre problemas
* Integração com antivírus ou segurança do sistema
* Relatórios em PDF/HTML com diagnóstico completo
* Uso de IA (simples) para sugerir melhorias com base nos dados do PC

📌 Estude:

* Geração de relatórios (reportlab, pdfkit)
* Criação de logs do sistema
* Machine learning básico com scikit-learn (opcional)

**🧪 Fase 7: Testes e Otimizações**

**Objetivo:** Garantir a estabilidade e segurança do programa.

📌 Estude:

* Testes automatizados em Python (unittest, pytest)
* Testes em diferentes versões do Windows (10, 11)
* Segurança: evitar apagar arquivos importantes acidentalmente

**✅ Resultado Final**

* Um programa instalável com interface gráfica
* Otimiza o sistema com um clique
* Dá informações detalhadas e confiáveis sobre o hardware e software
* Publicado no GitHub com documentação clara

FixMyPC/

├── README.md

├── main.py

├── ui/

├── modules/

├── assets/

└── requirements.txt

**📄 Arquivos e Pastas com Explicação**

**✅ README.md**

Documentação principal do projeto.

* Descrição geral
* Funcionalidades
* Como instalar e executar
* Imagens ou GIFs da interface
* Créditos e licença

**✅ main.py**

Arquivo principal que inicia a aplicação.

* Importa os módulos necessários
* Carrega a interface
* Executa os diagnósticos

**📁 ui/**

Interface gráfica do usuário.

* interface.py ou main\_window.py: layout principal
* styles.py (opcional): estilo visual (cores, fontes)
* Se usar customtkinter, fica tudo aqui

**📁 modules/**

Lógica e funcionalidades principais.

* cleanup.py: limpeza de arquivos temporários
* startup.py: gerenciamento de programas de inicialização
* hardware.py: coleta de dados de CPU, RAM, disco, etc.
* optimizer.py: sugestões e ações de melhoria
* systeminfo.py: informações gerais do sistema
* utils.py: funções auxiliares

**📁 assets/**

Recursos visuais.

* Ícones
* Imagens
* Sons (se quiser alertas sonoros)
* Arquivos de configuração .json, .ini, etc.

**📁 setup/**

Arquivos para empacotar e distribuir o app.

* build\_spec.py (PyInstaller ou cx\_Freeze)
* installer\_config.nsi (se for criar instalador com NSIS)
* Logo do app e ícone do executável
* version.txt

**📄 requirements.txt**

Lista de dependências do projeto.  
Exemplo:

txt

CopiarEditar

psutil

customtkinter

pywin32