

MÓDULO 05

Deploy de projetos

mentorama.

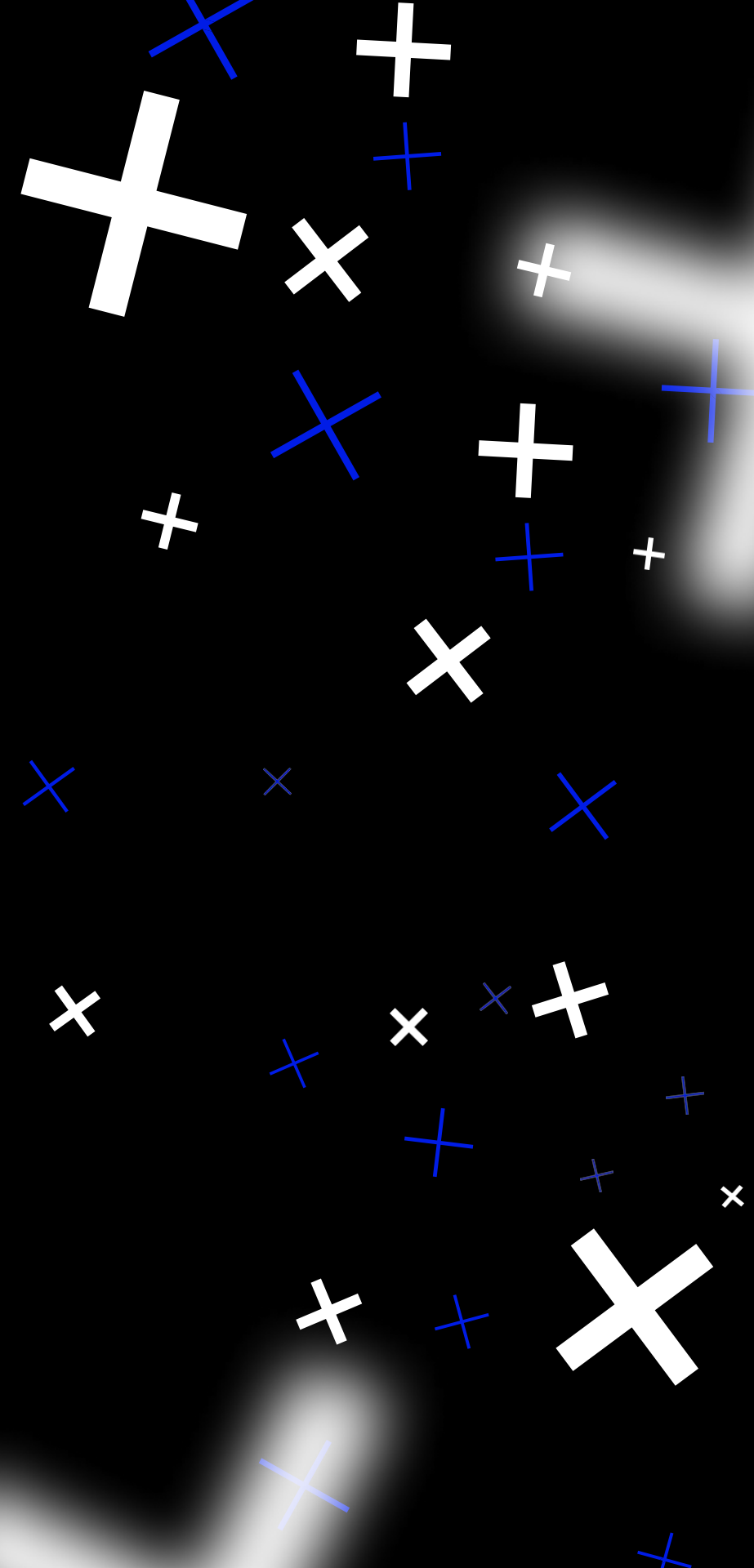
mentorama.

Python professional

Apresentação

mentorama.

mentorama.



O que veremos?

- ◆ Ferramentas de versionamento de código
- ◆ Arquivos README.md
- ◆ O que são os PEPs?
- ◆ Utilizando Linter no código
- ◆ GitFlow na prática
- ◆ Ambiente Linux
- ◆ Por que usar Docker?
- ◆ CI / CD – métodos de integração contínua, implantação ou entrega

mentorama.

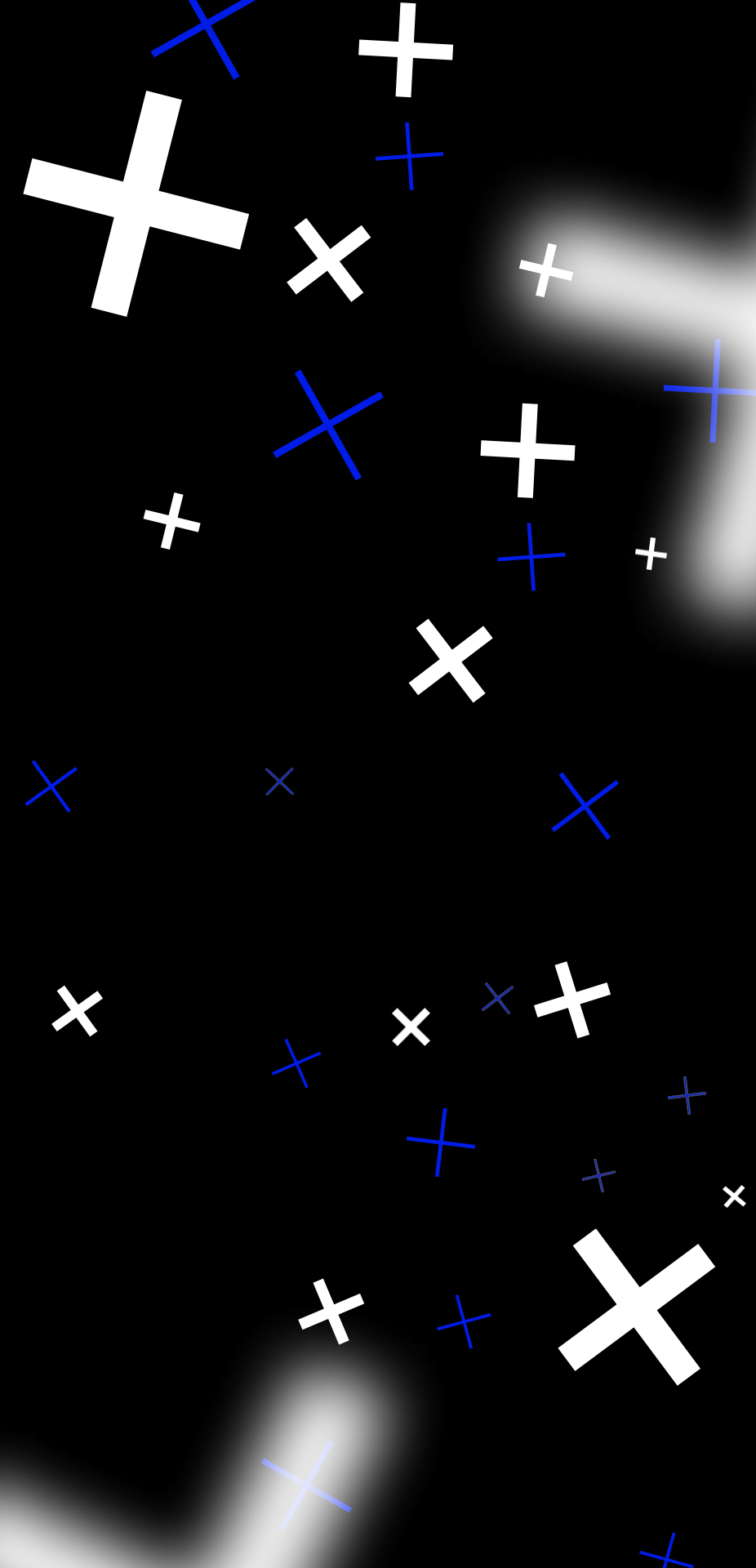
mentorama.

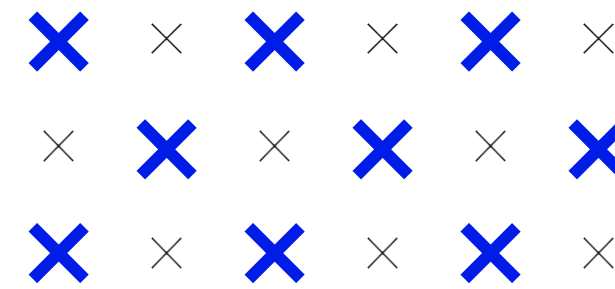
Python professional

Ferramentas de versionamento de código

mentorama.

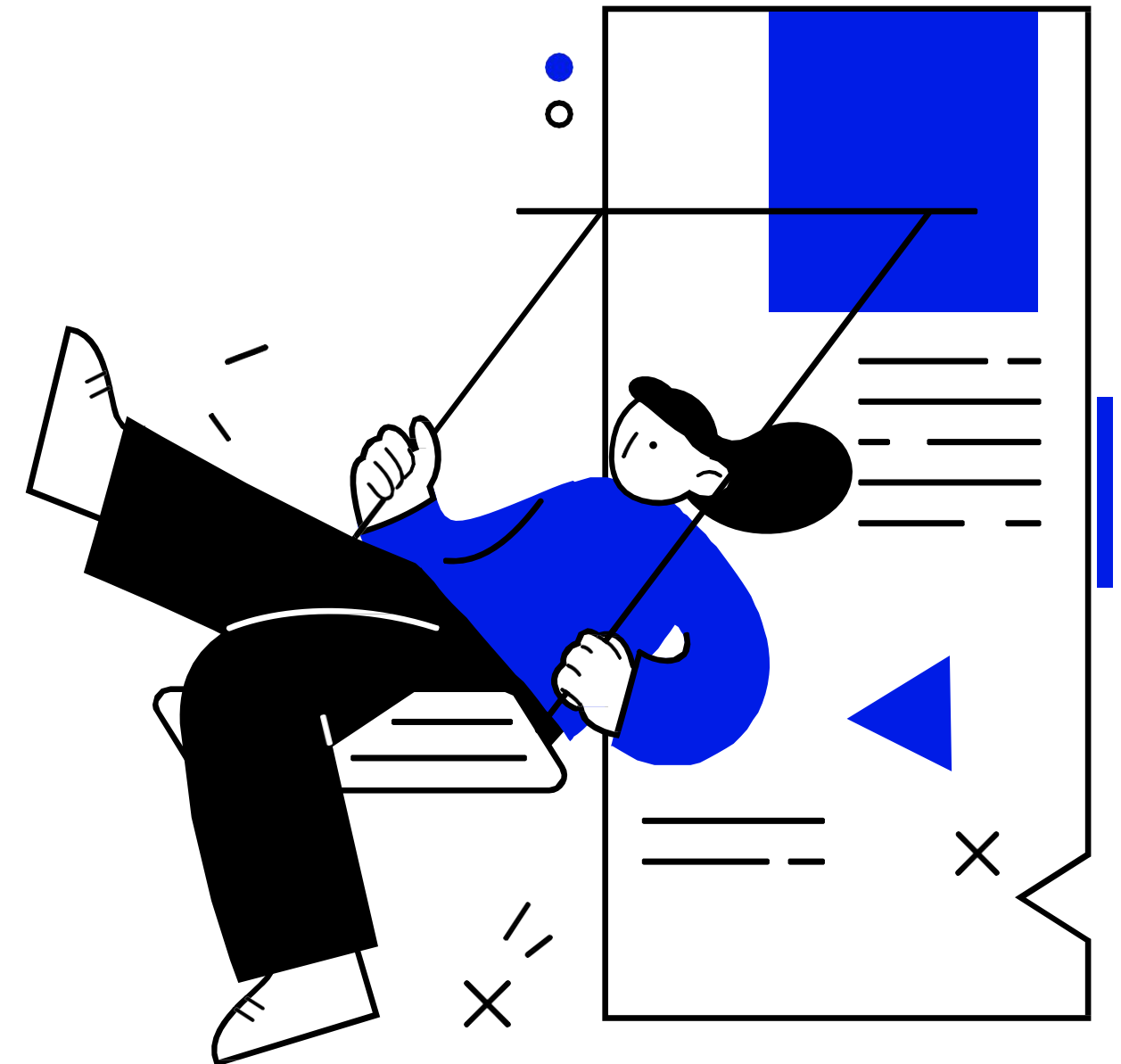
mentorama.





- ◆ Facilitar a publicação do código
- ◆ Manter a rastreabilidade das alterações realizadas
- ◆ Facilitar a colaboração de diversos integrantes no mesmo projeto
- ◆ O git é a ferramenta de versionamento de código mais popular, porém existem muitas outras
- ◆ O git é um versionador de código open source

mentorama.



mentorama.



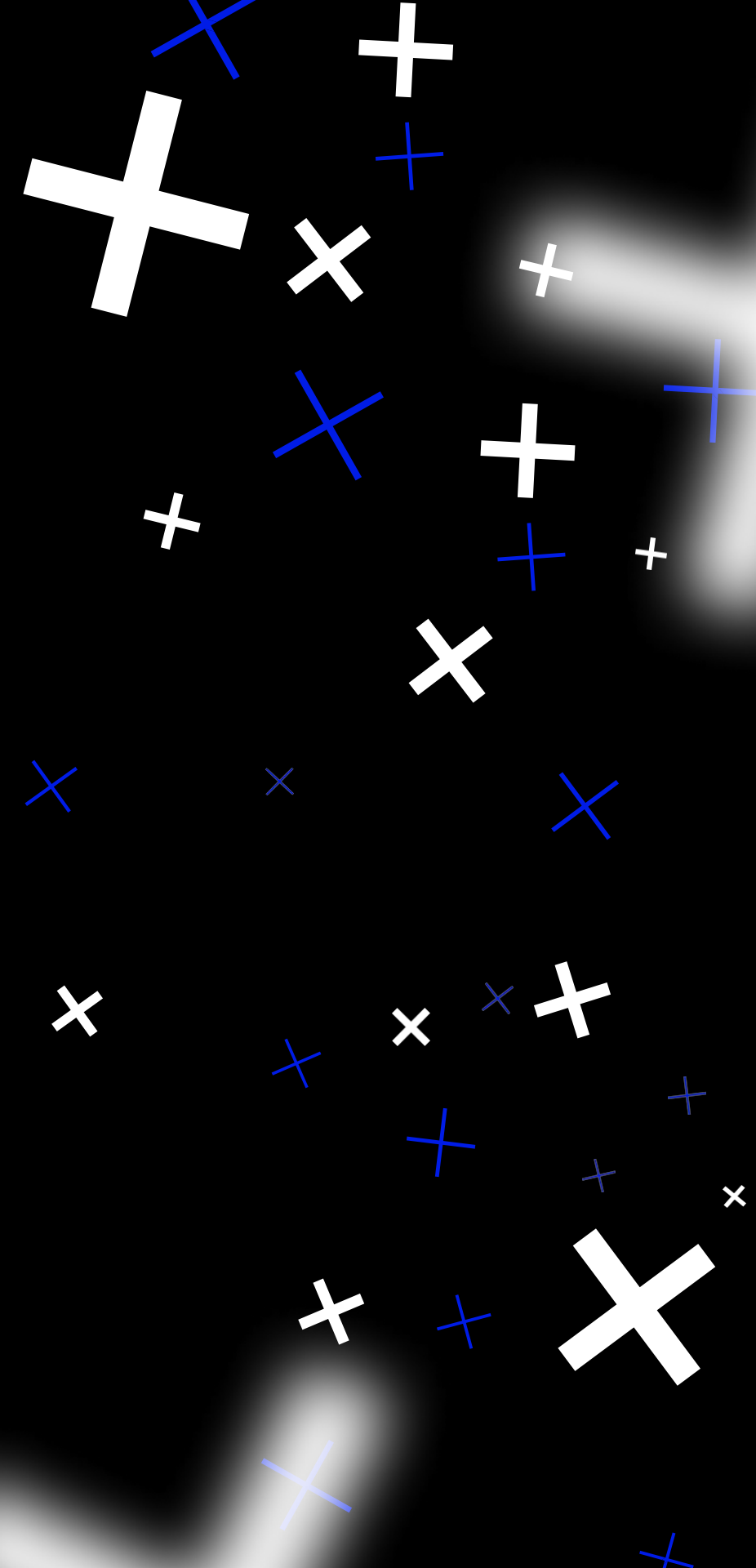
Mãos a obra: Publicando um projeto com o Git

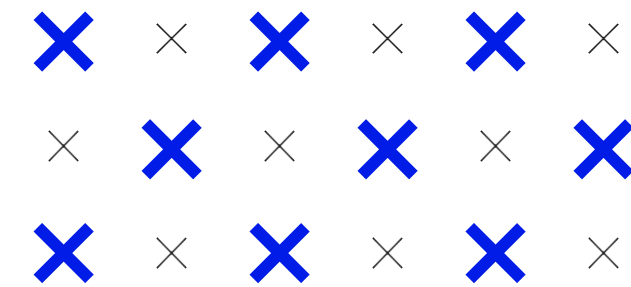
Python professional

Arquivos README.md

mentorama.

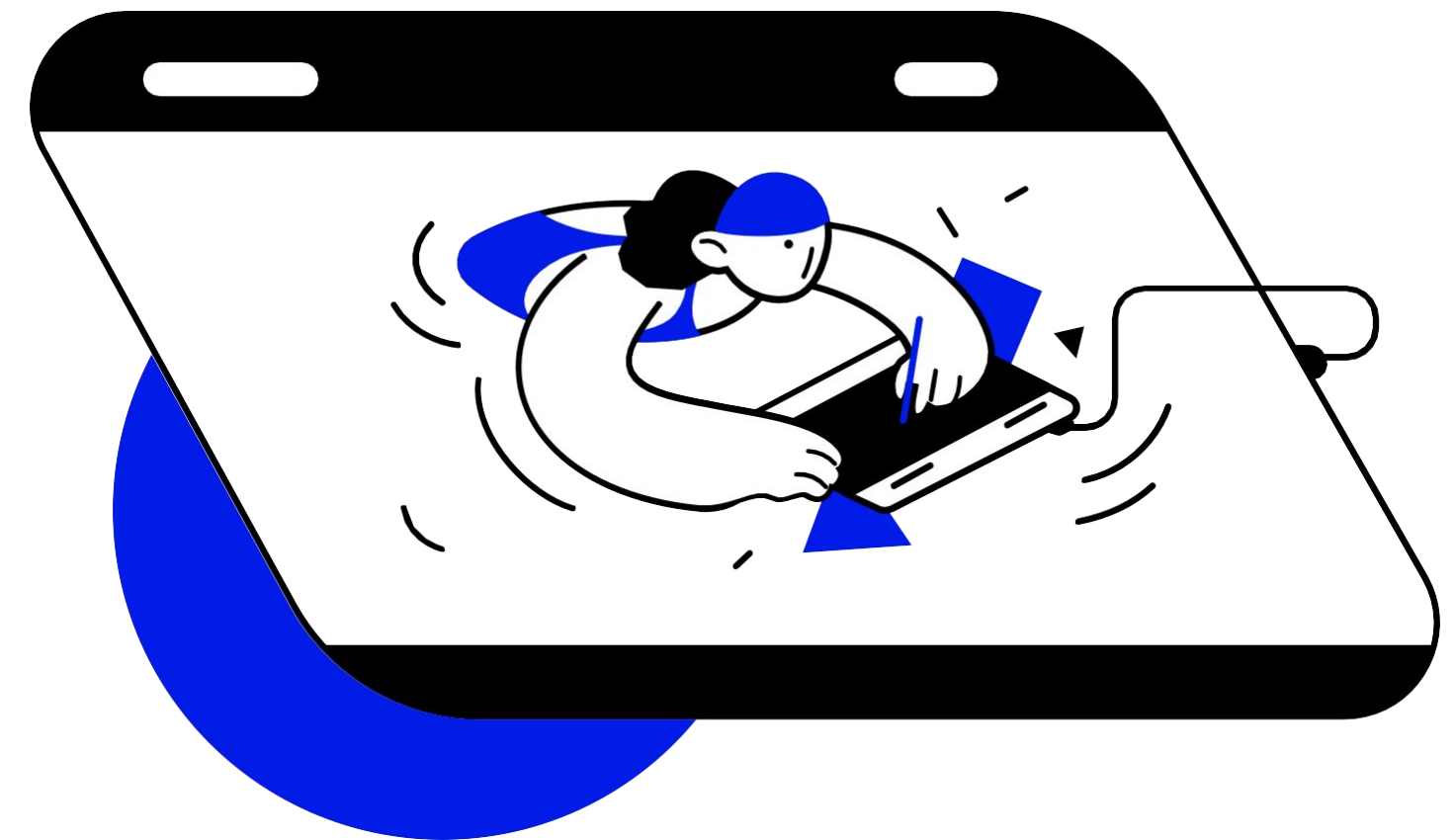
mentorama.





São arquivos utilizados para realizar uma apresentação do projeto e passar instruções básicas sobre a sua implementação e utilização

- ◆ É importante que o documento seja facilmente compreendido e compacto
- ◆ Não omita informações, mas também evite se alongar nas explicações
- ◆ É possível utilizar markdowns para melhorar a legibilidade do arquivo



mentorama.

mentorama.



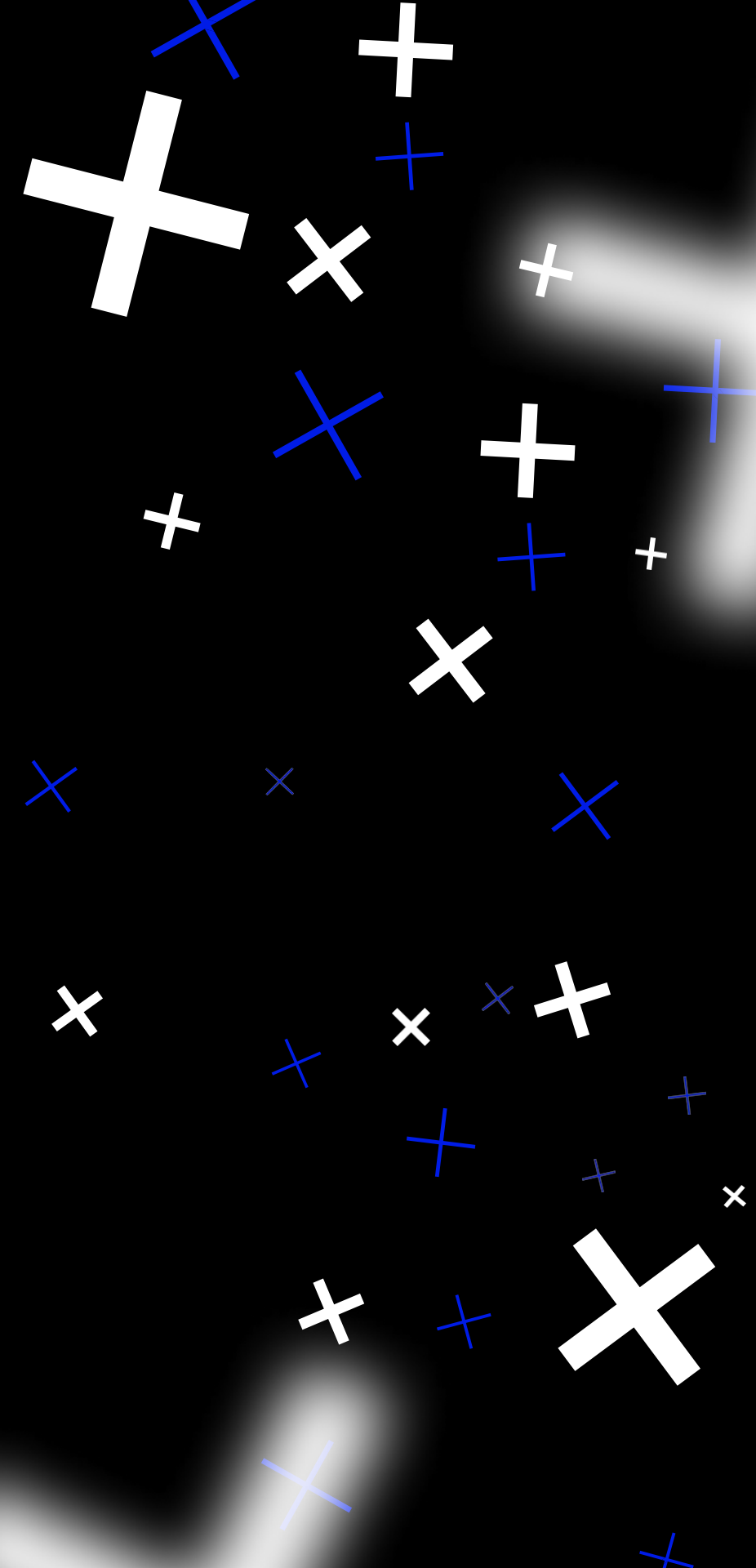
Mãos a obra: Construindo uma arquivo README.md básico

Python professional

O que são os PEPs?

mentorama.

mentorama.



Python Enhancement Proposals

- ◆ PEP é um documento de design que fornece informações para a comunidade, descreve uma nova característica, processo ou ambiente do Python
- ◆ PEPs possuem categorias (tipo) por temas abordados
- ◆ Qualquer pessoa pode enviar uma PEP
- ◆ O PEP-8, criado em 2001, é o mais conhecido para Style Guide para o Python

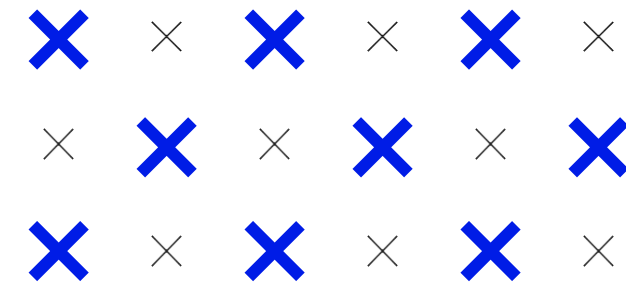
<https://www.python.org/dev/peps/pep-0001/>

mentorama.

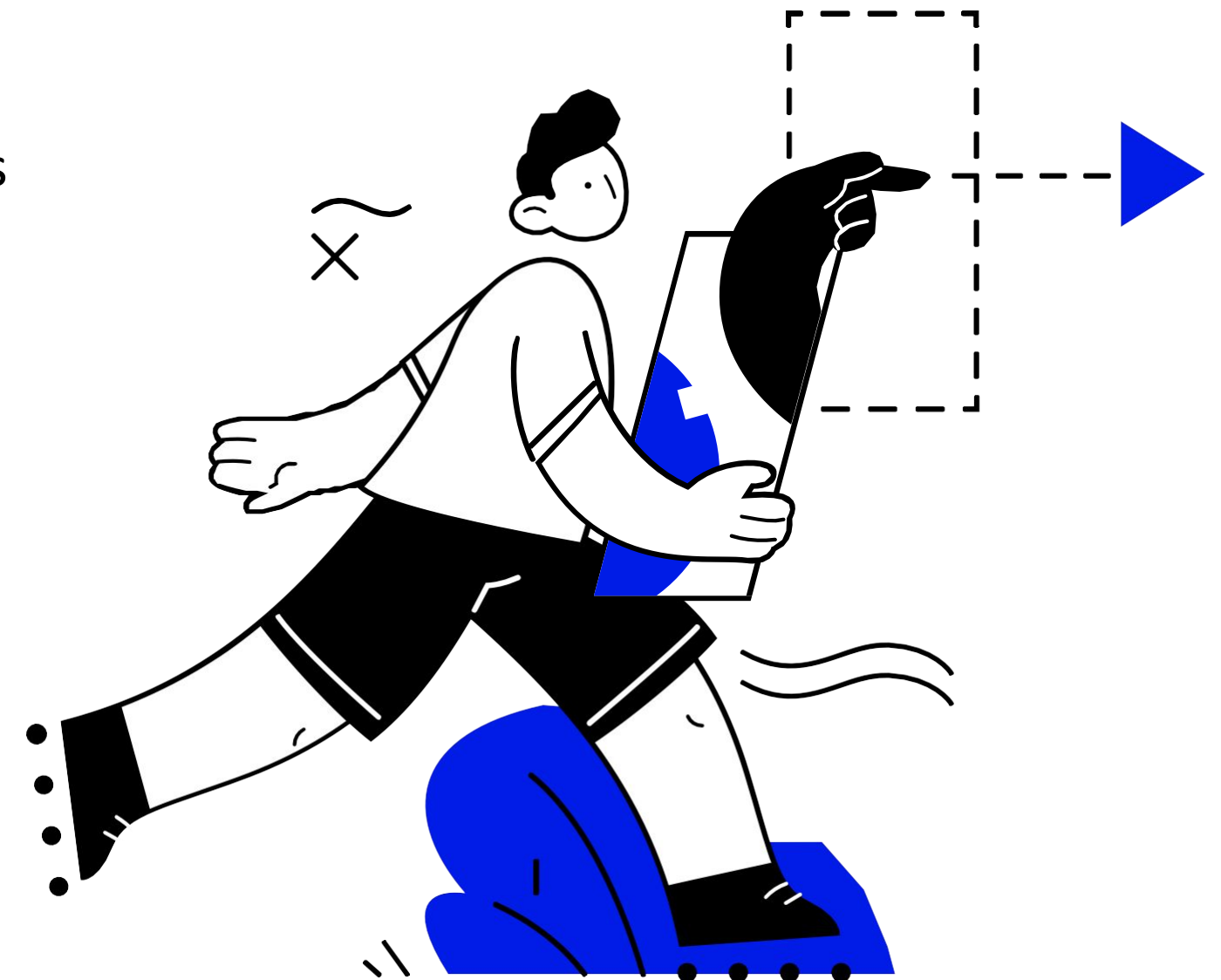
mentorama.



PEP-8



- ◆ A indentação é feita preferencialmente com quatro espaços
- ◆ Cada linha deve ter no máximo 72 caracteres
- ◆ Nomes de funções deve ser em letras minúsculas com as palavras separadas por underline
- ◆ Não faça a comparação com booleanos
- ◆ No PEP-257 estão as convenções de DocStrings
- ◆ Existem bibliotecas para verificar o código automaticamente



mentorama.

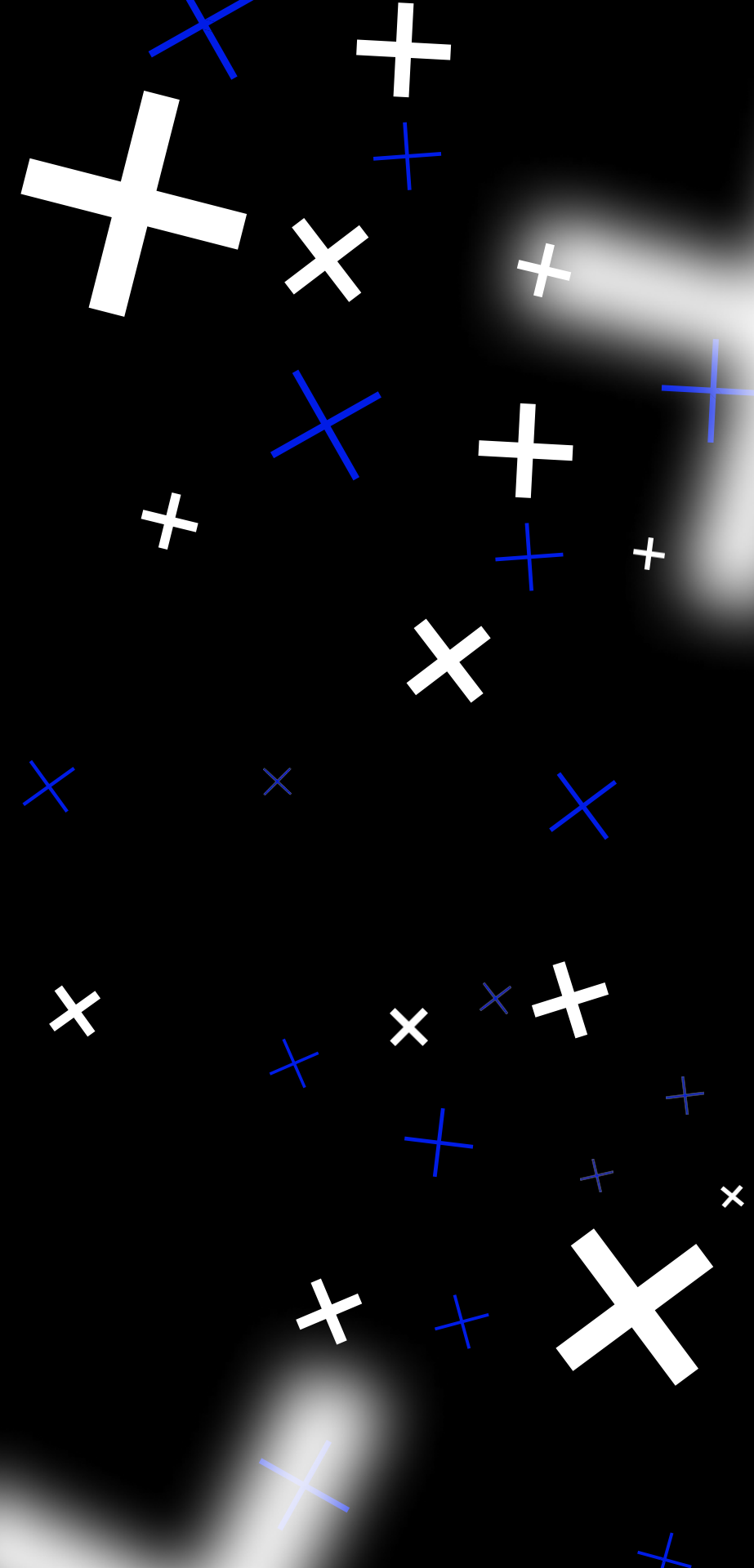
mentorama.

Python professional

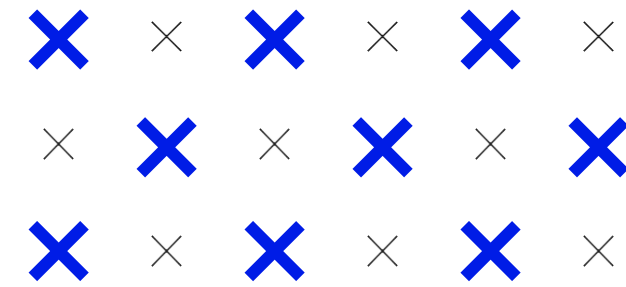
Utilizando Linter no código

mentorama.

mentorama.



Vantagens em utilizar um linter



- ◆ Facilita a padronização do código dentro de uma equipe
- ◆ Cada linter aplica alguns padrões específicos
- ◆ Existem diversas bibliotecas para aplicações de linter
- ◆ Os linters podem realizar alterações de forma automatizada





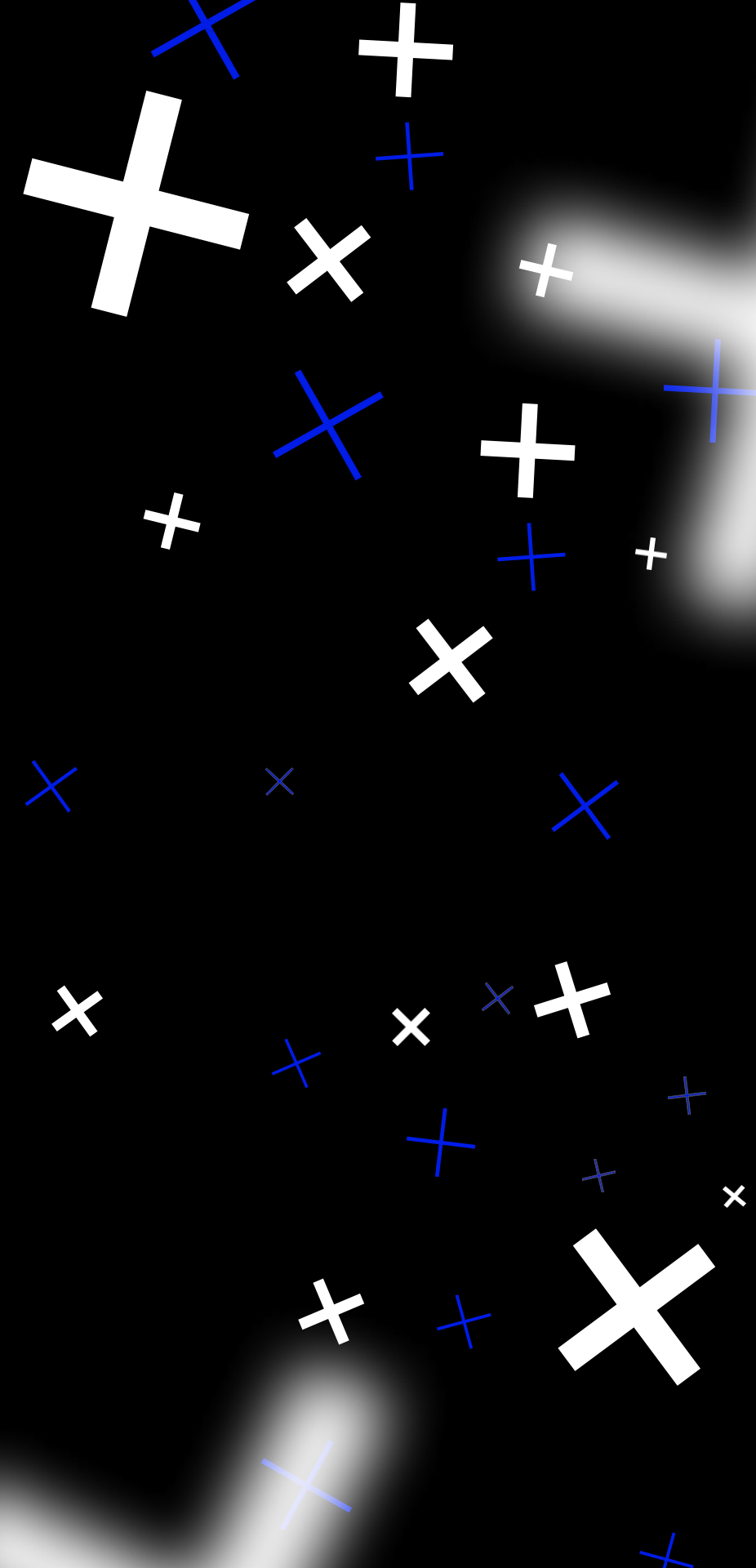
Mãos a obra: Utilizando linters

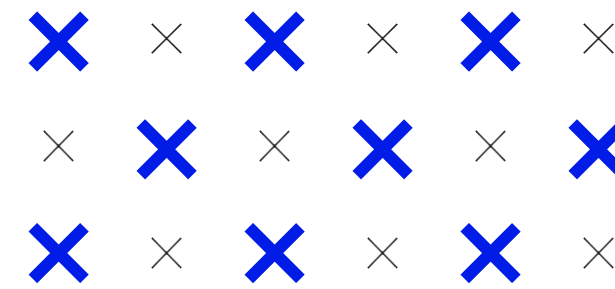
Python professional

GitFlow na prática

mentorama.

mentorama.





- ◆ É um design de fluxo de trabalho Git publicado por Vincent Driessen
- ◆ Utilização de uma branch master e uma branch develop
- ◆ A cada nova versão da aplicação que é concluída, é realizado o merge entre as branches master e develop
- ◆ Para cada nova feature a ser implementada, uma nova branch é criada e realizado o merge com a develop
- ◆ Para realizar o merge entre duas branches é aberto um pull request e submetido à revisão de outros membros da equipe



mentorama.

mentorama.



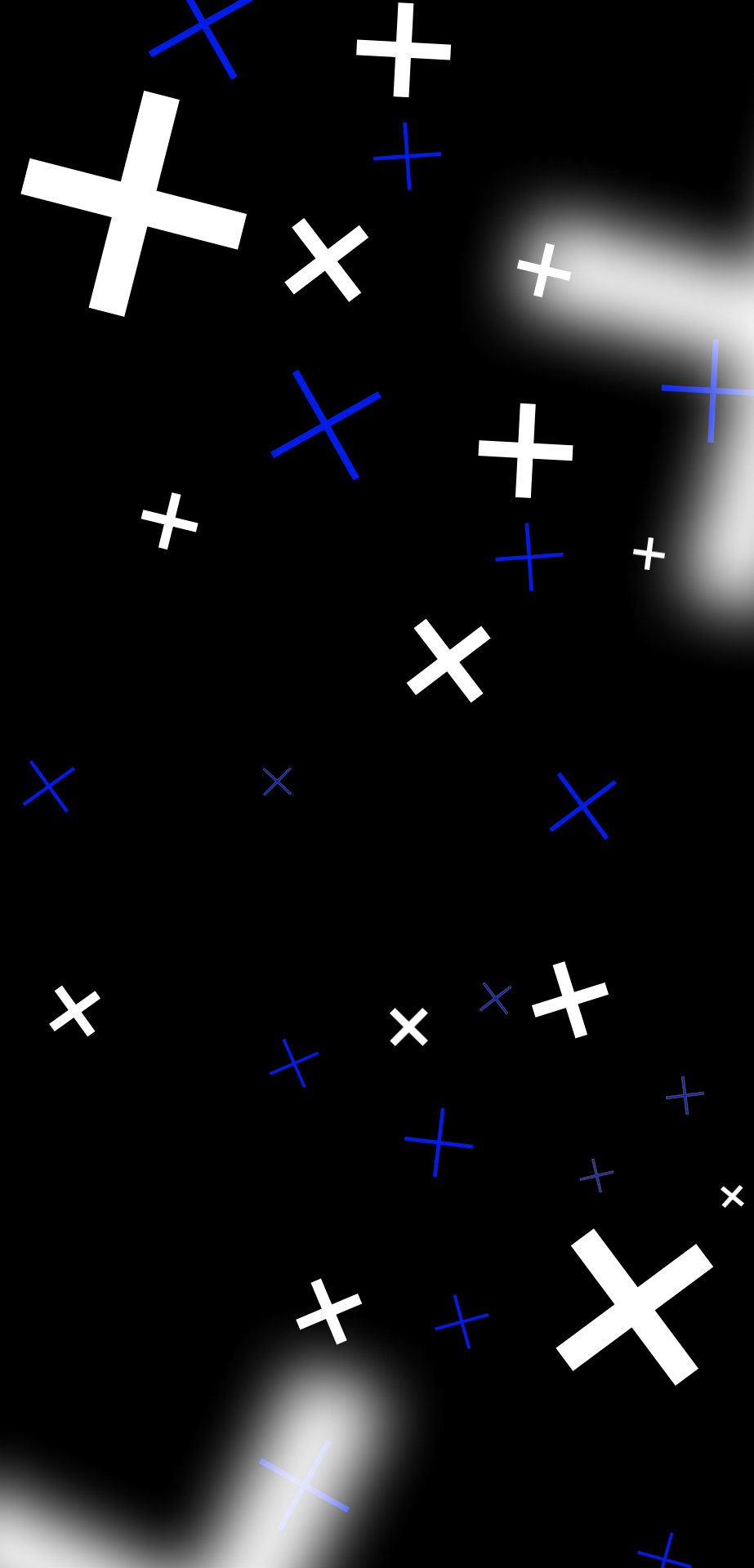
Mãos a obra: Seguindo o Git Flow em um projeto

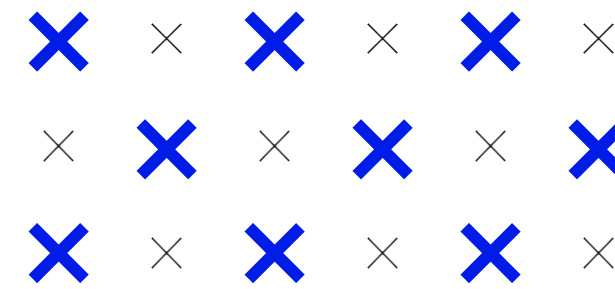
Python professional

Ambiente Linux

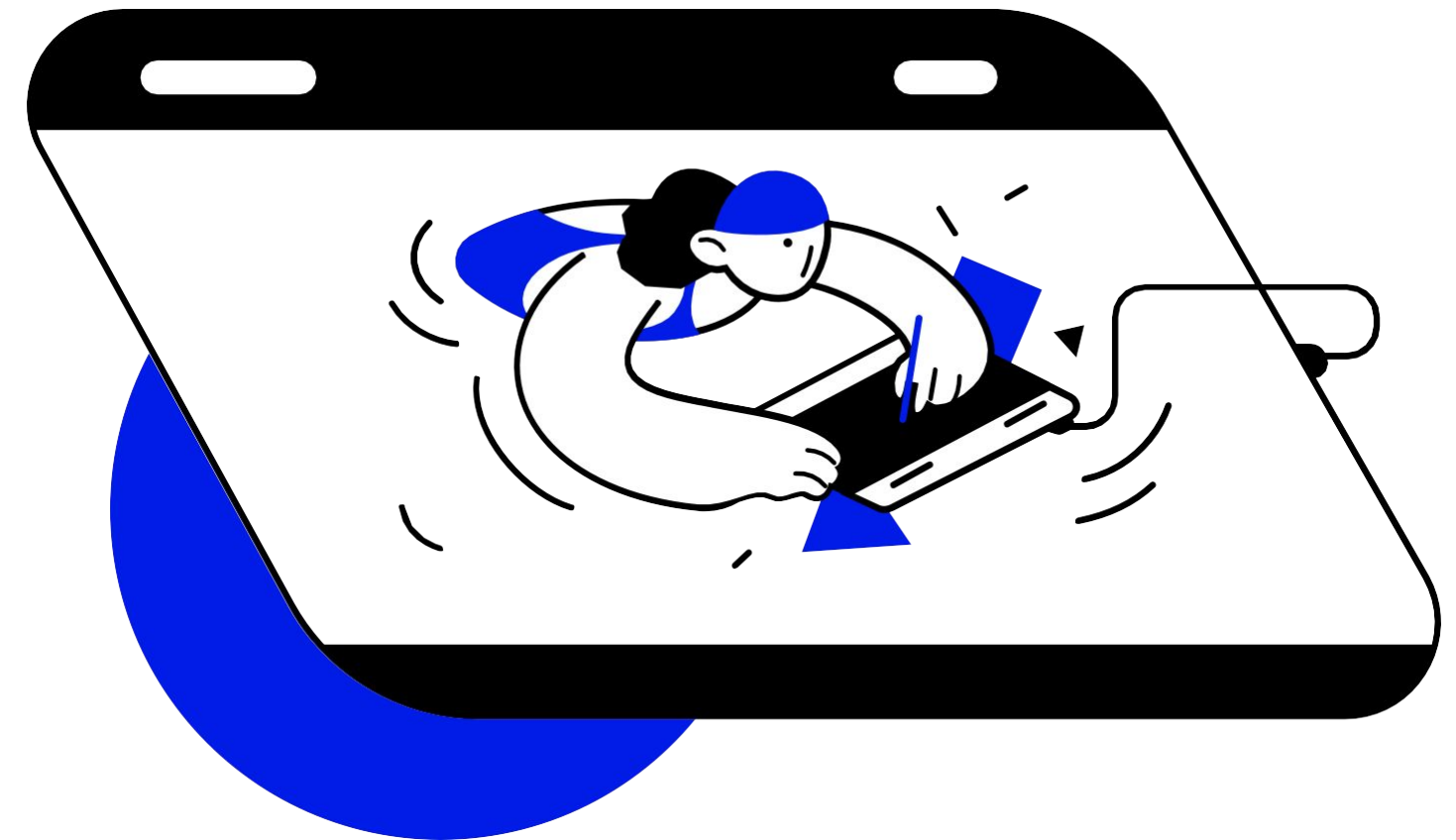
mentorama.

mentorama.



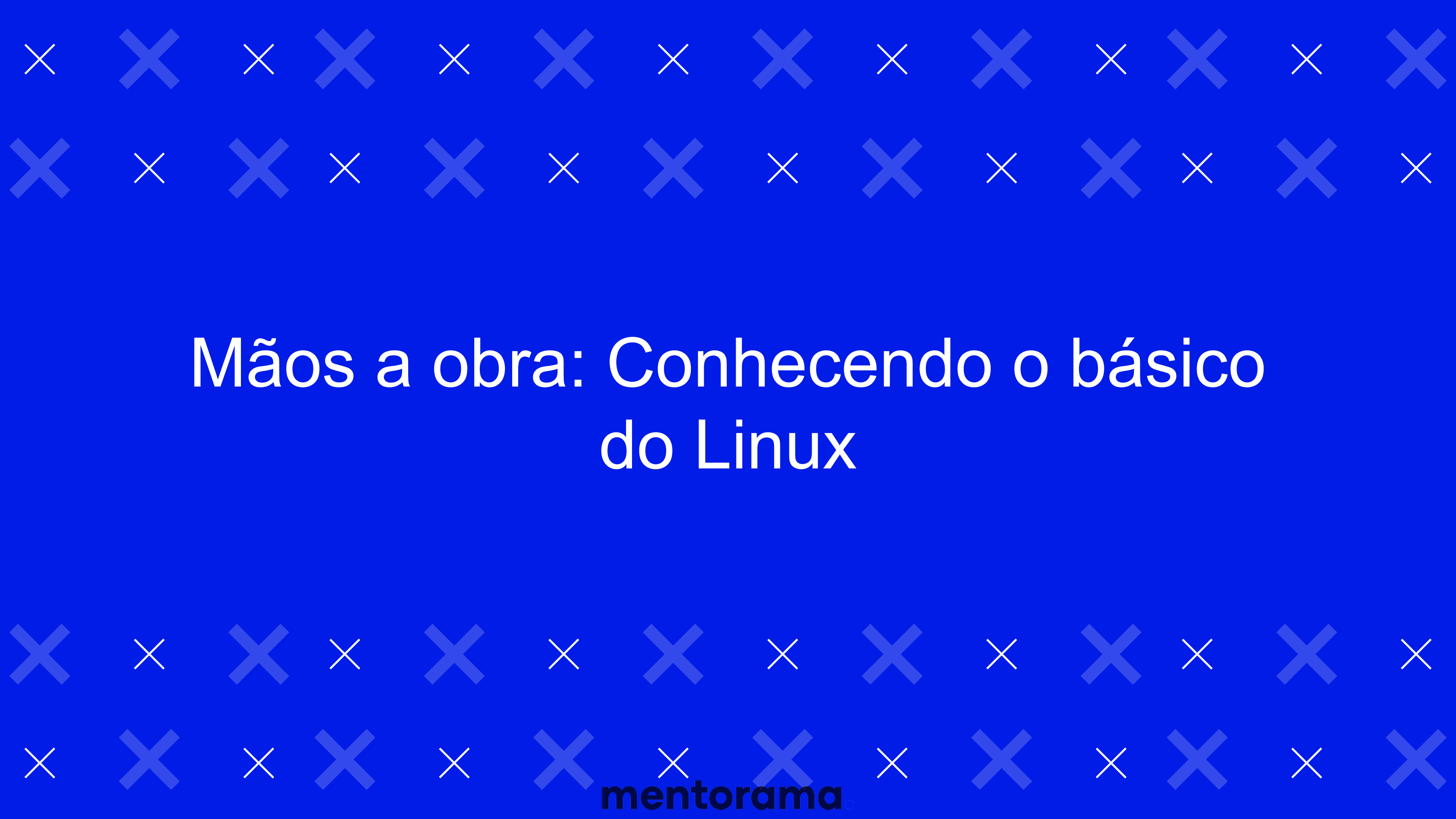


- ◆ A linguagem Python está presente para diversos Sistemas Operacionais
- ◆ Em servidores web, a prevalência é de utilização de Sistemas Operacionais Linux
- ◆ Existem diversas distribuições Linux diferentes e cada uma delas possui suas particularidades
- ◆ Não é necessário mudar o Sistema Operacional utilizado para desenvolver, mas é importante ter um conhecimento básico em Linux



mentorama.

mentorama.



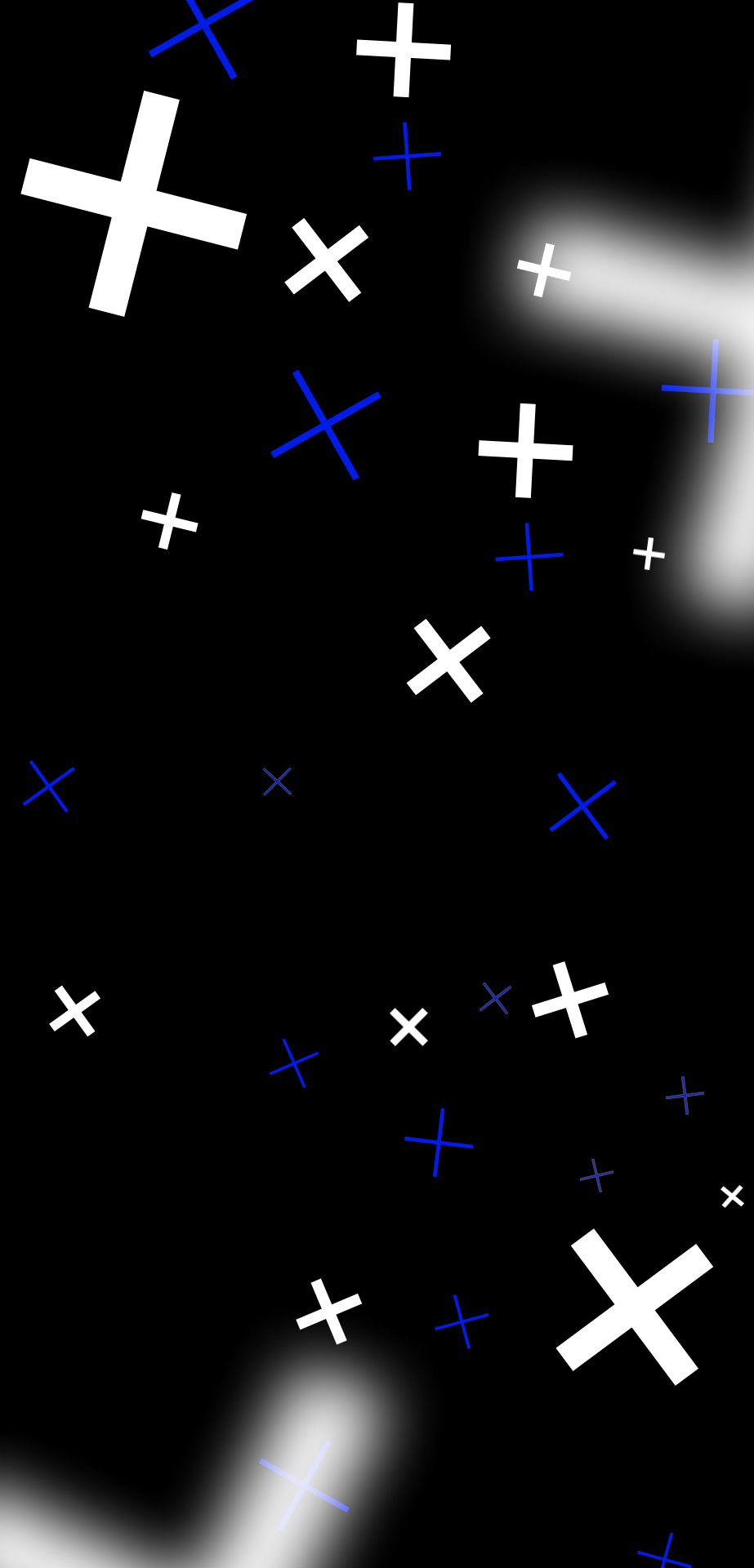
Mãos a obra: Conhecendo o básico do Linux

Python professional

Por que usar Docker?

mentorama.

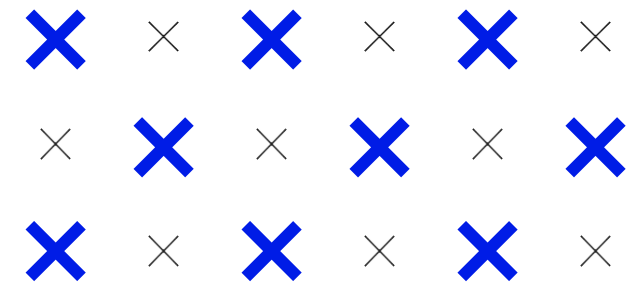
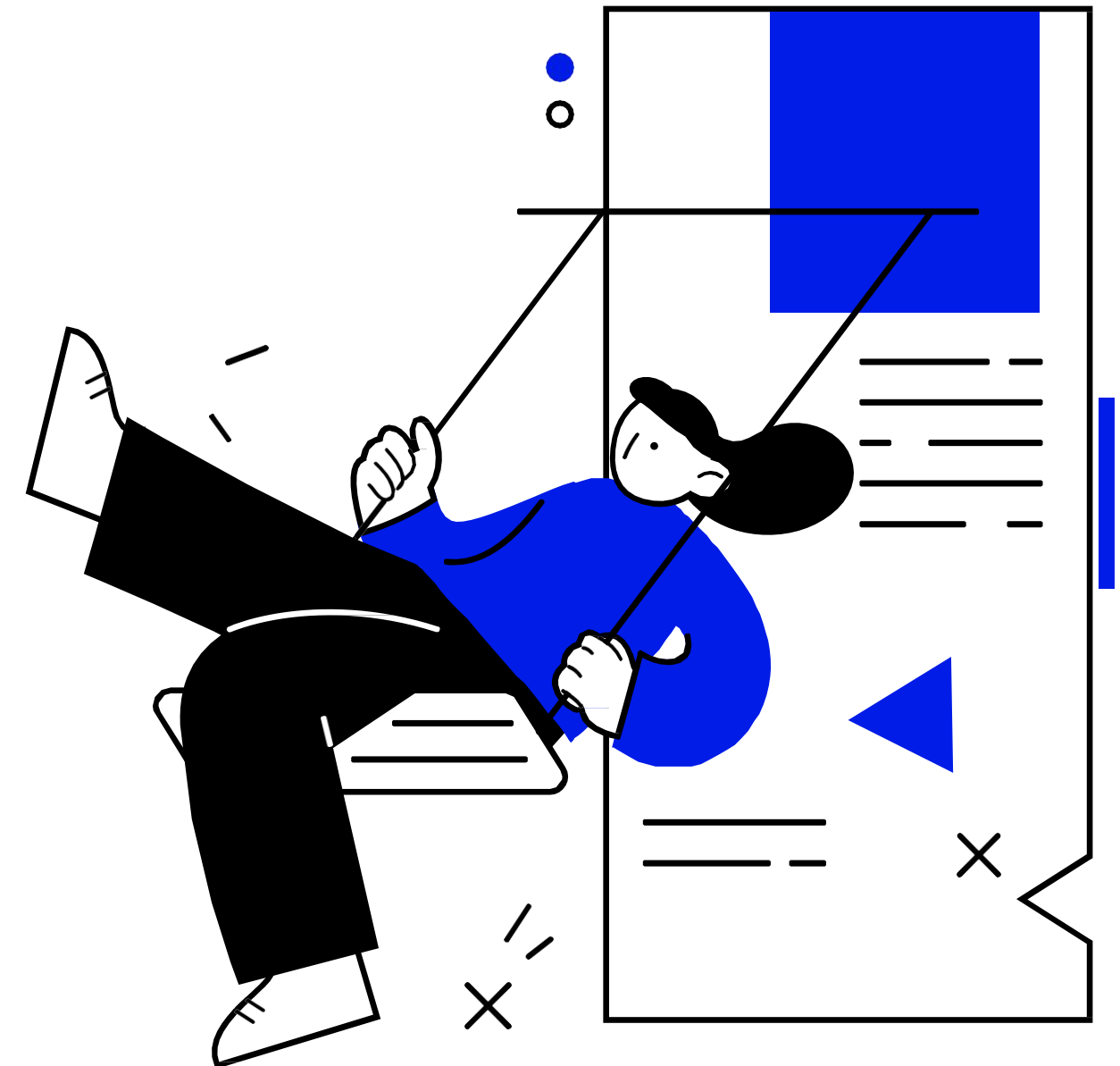
mentorama.



- ◆ Independência de Sistemas Operacionais
- ◆ Evita conflitos entre aplicações no mesmo servidor
- ◆ Facilita o deploy, a ativação e a atualização dos projetos
- ◆ Mais leve e escalável do que máquinas virtuais
- ◆ Facilita a implementação de arquitetura em microserviços

mentorama.

mentorama.





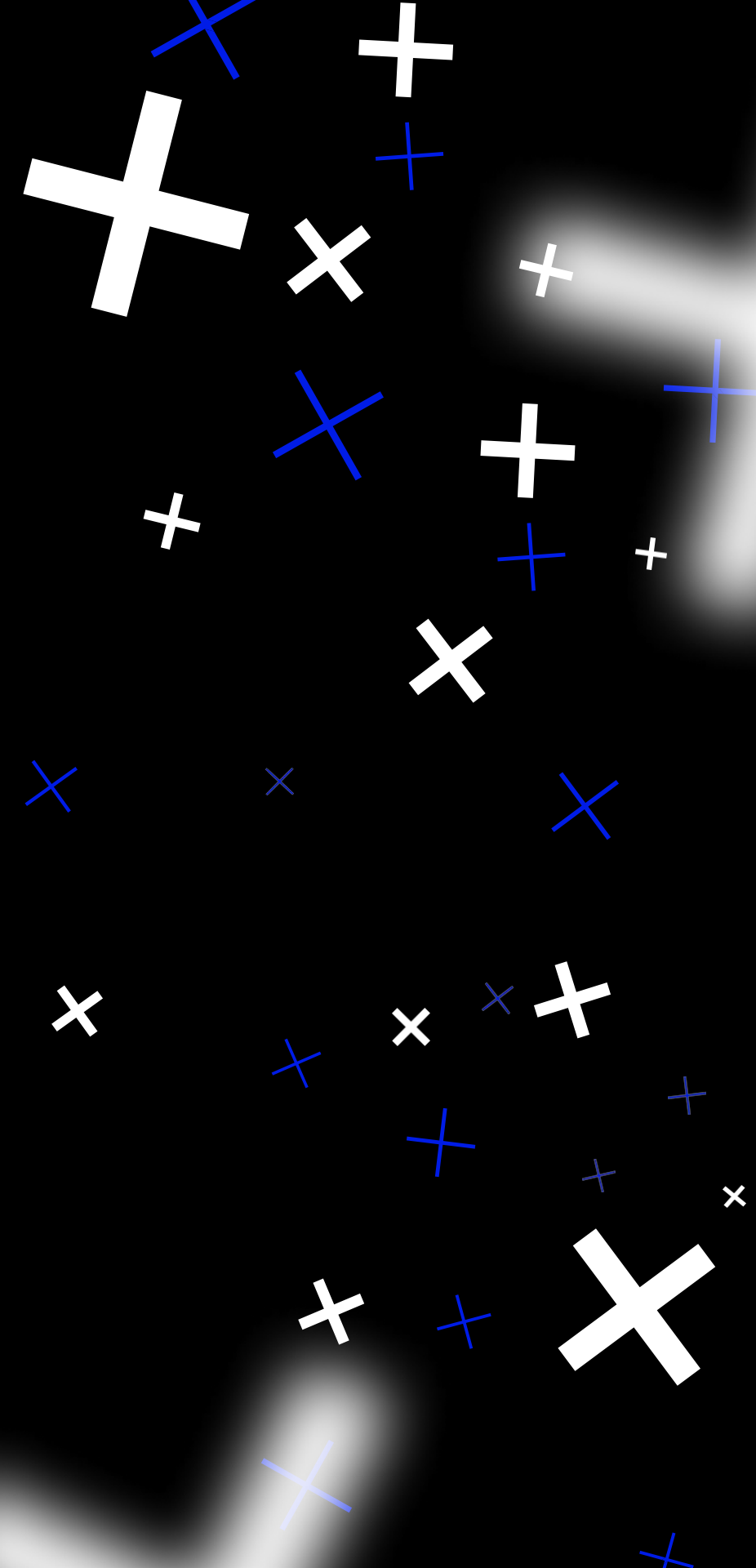
Mãos a obra: Aplicando Docker em nosso projeto

Python profissional

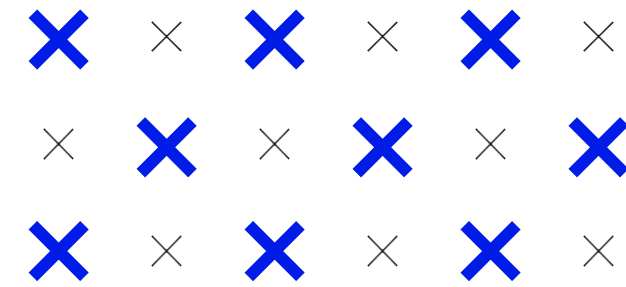
CI / CD – métodos de
integração contínua,
implantação ou entrega

mentorama.

mentorama.



Continuous Integration e Continuous Delivery são importantes para projetos



- ◆ É preciso ser dinâmico e entregar novas features em ciclos curtos e com qualidade
- ◆ Para realizar a entrega é necessário garantir que o código está testado e segue os padrões de lint definidos pela equipe
- ◆ A utilização de CI / CD pode evitar que falhas sejam incorporados ao projetos
- ◆ Algumas ferramenta podem ajudar na implementação do CI / CD no projeto



mentorama.

mentorama.

Tarefa

Crie um projeto em Python:

- ◆ Crie um projeto em uma plataforma de hospedagem de código (EX: Github)
- ◆ Disponibilize o código da tarefa do Módulo 1 utilizando o Gitflow:
 - ◇ Crie um arquivo README.md para o projeto
 - ◇ Crie uma branch developer develop
 - ◇ Execute os testes unitários e utilize uma ferramenta de linter no código da tarefa do módulo 1
 - ◇ Se os testes e o lint estiverem corretos, faça o primeiro commit do projeto na branch develop
- ◆ Adicione o Docker ao seu projeto seguindo o Gitflow