

Evoluindo o sistema

Renan de Assis

Ana Dulce

Quem é o Renan?

- Bacharel em Física



Quem é o Renan?

- Bacharel em Física
- Trabalho com engenharia de software no Serasa



Quem é o Renan?

- Bacharel em Física
- Trabalho com engenharia de software no Serasa
- Gosto de vôlei, jogos de tabuleiro e tenho uma tatuagem do desenho Avatar
- Ajudo na organização de eventos Python Brasil



Quem é a Ana?

- Desenvolvedora Python desde 2018 - atualmente engenheira de software no KaBuM!;
- Cofundadora do PyLadies São Carlos;
- Organizadora de eventos da comunidade Python no tempo livre;
- Atual Conselheira e ex-presidente da Associação Python Brasil;
- Python Fellow Member;

Motivação

- Diversas oficinas e cursos de "crie sua primeira API"
- O sistema ficou pronto, mas e agora?



Motivação

- Existem muitos outros recursos "ao redor" da API para melhorar nosso sistema
 - Qualidade
 - Confiabilidade
 - Observabilidade
 - Facilidade de manutenção



Alinhamentos iniciais

- Contexto aplicações web
- Não levem como verdade absoluta
- O que essa oficina **não** vai abordar?
 - Como **criar** uma API com nenhum framework de Python
 - Funcionamento detalhado de uma API



O que essa oficina vai te mostrar?

- 1. configuração do ambiente
- 2. documentação
- 3. logs
- 4. testes
- 5. formatadores
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks



BORA TIMEE

Em que passo estamos?

- 1. configuração do ambiente
- 2. documentação
- 3. logs
- 4. testes
- 5. formatadores
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

Precisamos de um projeto para evoluir...

<https://github.com/renan-asantos/tutorial-evoluindo-sistema>

The screenshot shows a GitHub repository page for 'tutorial-evoluindo-sistema'. The repository is public and has 3 branches and 0 tags. The main branch is selected. There are four commits from 'renan-asantos' with the message 'commit inicial'. The first commit is 1675e4c, 8 hours ago, with 1 commit. The other three commits are 8 hours ago. The repository has 0 stars and 1 watching. The 'About' section notes 'No description, website, or topics provided.' The page includes standard GitHub navigation and search features.

tutorial-evoluindo-sistema Public

Pin Unwatch 1 Fork 0 Star 0

main 3 Branches 0 Tags Go to file Add file Code About

renan-asantos commit inicial 1675e4c · 8 hours ago 1 Commit

migrations commit inicial 8 hours ago

slides commit inicial 8 hours ago

src commit inicial 8 hours ago

No description, website, or topics provided.

Readme Activity 0 stars 1 watching

Configuração do ambiente

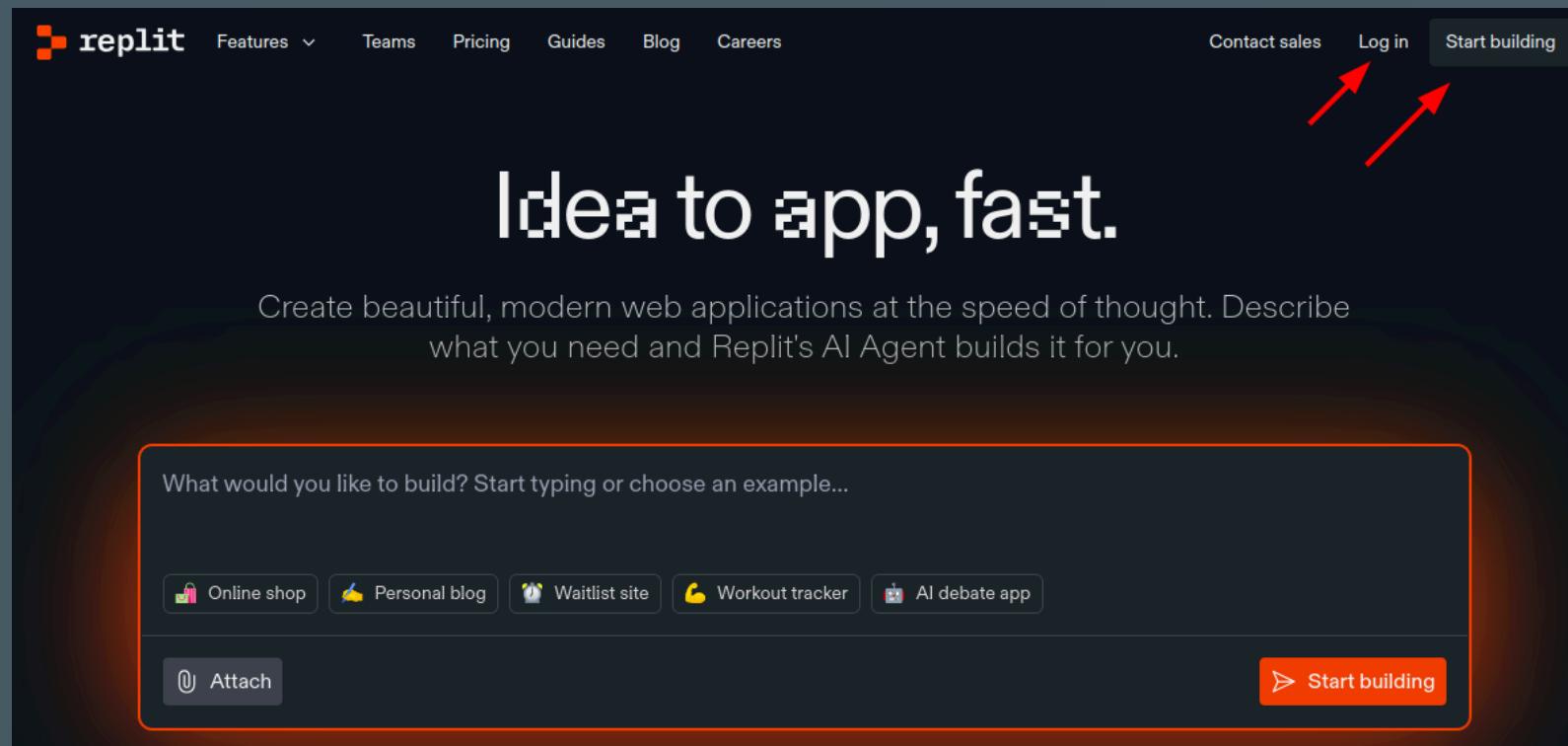
- Ambiente Unix-like
- Clonar projeto
- Versão do python
- Gerenciador de dependências
- Ambiente virtual ativo
- Rodar o projeto

Vamos usar o repl.it.com



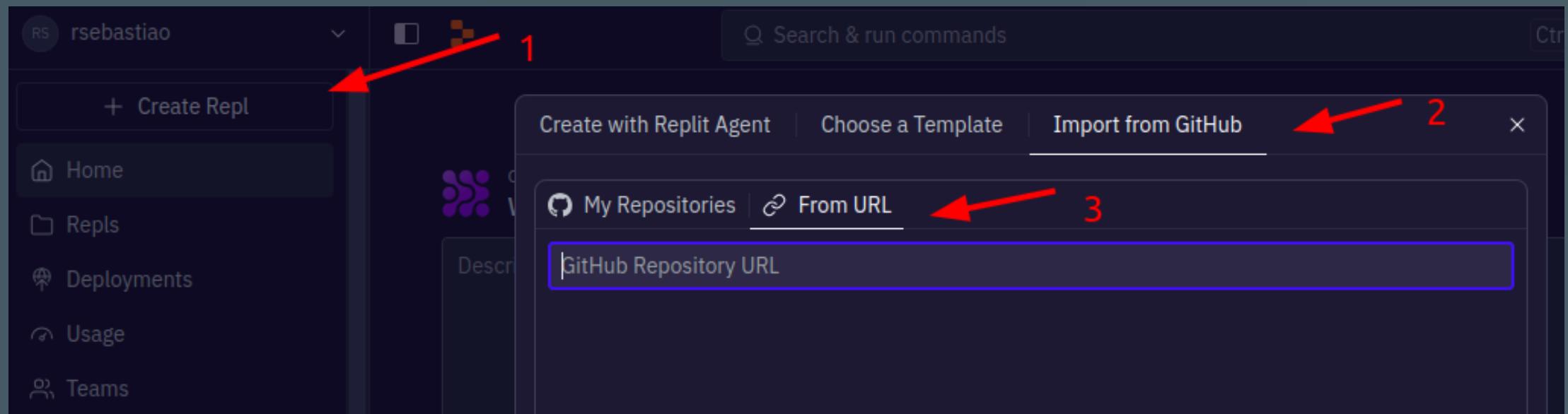
Configuração replit

- Acessar <https://replit.com>
- Fazer o login (ou criar uma nova conta no "Start Building")



Configuração replit

- "Create repl" >> "Import from Github" >> "From URL"

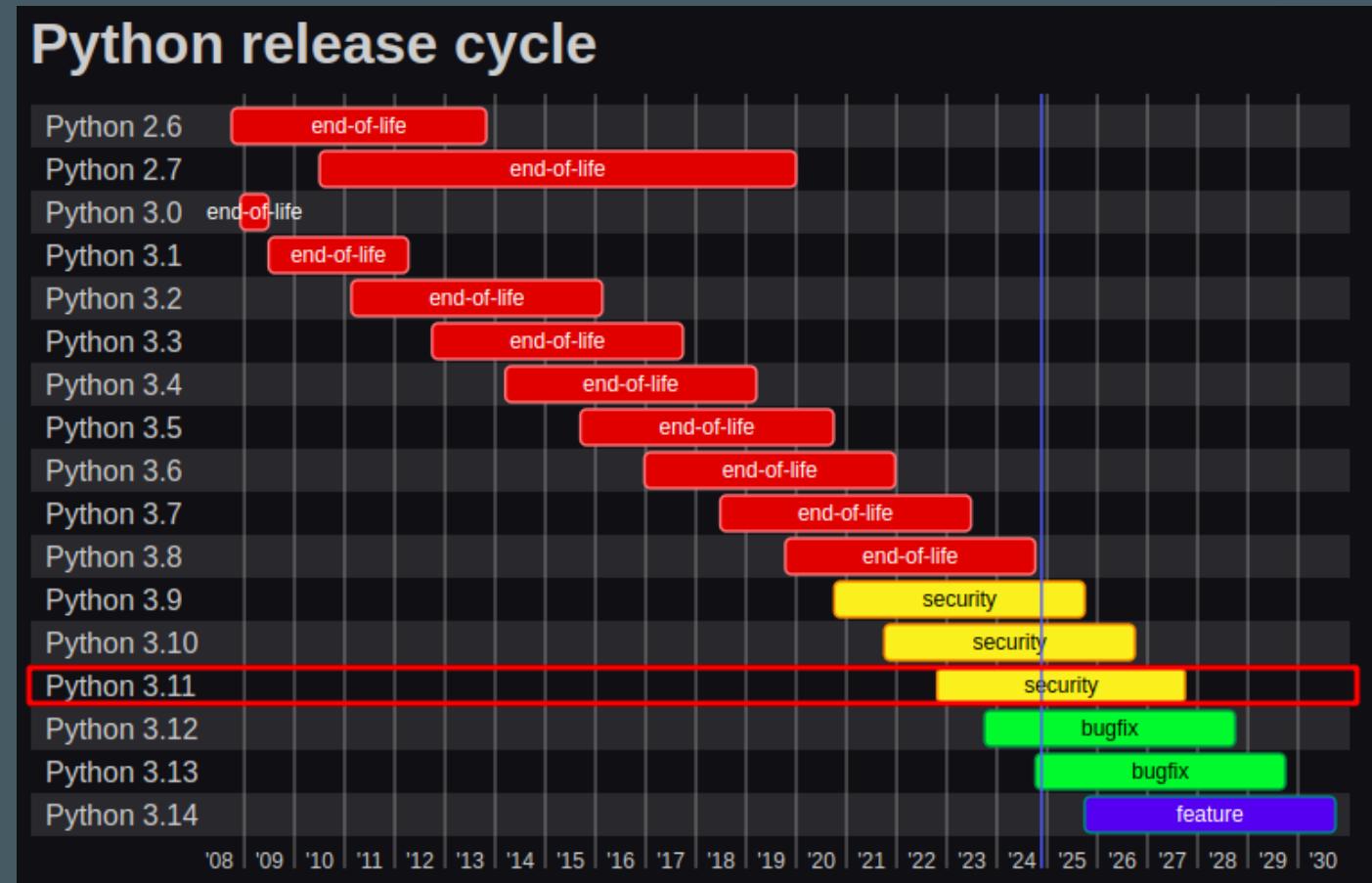


(lembrar de alterar arquivo .replit)

Ambiente Unix-like

- Sistema operacional que se comporta similarmente ao system Unix (lançado em 1971)
- Exemplos:
 - Linux e suas distribuições (Ubuntu, Debian, Fedora, etc)
 - MacOS (Darwin)
 - Android (baseado em Linux)
 - iOS (baseado no Darwin)

Versão do python



Versão do python - Pyenv

 **pyenv** Public

[Sponsor](#) [Watch 387](#) [Fork 3.1k](#) [Star 39.5k](#)

[master](#) [2 Branches](#) [187 Tags](#) [Go to file](#) [Add file](#) [Code](#)

Commit	Message	Date
 noelleleigh	README: Fix Markdown in "Notes about python releases" (#3112)	61c0f25 · yesterday
	CI: replace set-output with GITHUB_OUTPUT (#3079)	last month
	Merge remote-tracking branch 'rbenv/master' into rbenv-1.0	8 years ago
	Fix fish subcommand completion	8 years ago
	2.4.19	4 days ago
	Describe --no-rehash option in the manpage (#2832)	last year
	Add CPython 3.14.0a2 (#3110)	4 days ago
	Add 3.13.0b3t and exclude it from pyenv latest (#3001)	5 months ago
	Merge remote-tracking branch 'rbenv/master' into rbenv-20...	6 years ago

About

Simple Python version management

[python](#) [shell](#)

[Readme](#) [MIT license](#) [Activity](#) [Custom properties](#)

[39.5k stars](#) [387 watching](#) [3.1k forks](#)

[Report repository](#)

Releases 112

Gerenciador de dependências

- Um sistema pode ser composto de código nosso e também de outras pessoas
- Geralmente precisamos instalar bibliotecas (dependências) para fazer um projeto maior
- Precisamos de uma ferramenta para auxiliar nessa tarefa
- Exemplos: Pip, pipenv, poetry, uv



Poetry

 **poetry** Public

[Watch 190](#) [Fork 2.3k](#) [Star 31.8k](#)

[main](#) [10 Branches](#) [135 Tags](#) [Add file](#) [Code](#)

Commit	Message	Time
 radoering env: fix check whether path is relative to system site packages (#9861)	c70cbf4 · 4 days ago	3,393 Commits
 .github	drop support for Python 3.8 (#9692)	last month
 assets	docs: update README gif	2 years ago
 docs	make allow-prereleases a tri-state setting to really forbid ...	last week
 src/poetry	env: fix check whether path is relative to system site packa...	4 days ago
 tests	env: fix check whether path is relative to system site packa...	4 days ago
 .cirrus.yml	drop support for Python 3.8 (#9692)	last month
 .gitattributes	tests/repo: generate metadata file fixtures	8 months ago
 .gitignore	remove unwanted generated files from fixtures (#9103)	9 months ago

About

Python packaging and dependency management made easy

[python-poetry.org](#)

python package-manager dependency-manager packaging poetry

[Readme](#) [MIT license](#) [Code of conduct](#) [Cite this repository](#) [Activity](#) [Custom properties](#) [31.8k stars](#) [190 watching](#)

Instalar dependências

- □ ×

```
# Cria venv e instala as dependências  
poetry install
```

```
# Instala pacote  
poetry add <lib>
```

```
# Instala pacote para desenvolvimento  
poetry add --group dev <lib>
```

Rodando o projeto

- Configurar o banco

```
alembic upgrade head  
alembic revision --autogenerate -m "mensagem de criação"
```

- Rodar o projeto

```
fastapi dev src/app.py
```

AVISAR SOBRE LIMITE REPLIT

Rodando o projeto

- Acessar /docs, ver o swagger e chamar um dos endpoints com sucesso (cadastrar um filme e um genero e retornar eles)

The screenshot shows the FastAPI Swagger UI interface. At the top, it displays "FastAPI 0.1.0 OAS 3.1" and a link to "/openapi.json". Below this, the "genres" endpoint is listed with the following operations:

- GET /genre/** Read Genres (blue button)
- POST /genre/** Create Genre (green button, highlighted)
- GET /genre/pages/** Read Genres By Page (blue button)
- GET /genre/{id}/** Read Genres (blue button)

Each operation has a dropdown arrow to its right, indicating more details can be viewed.

Em que passo estamos?

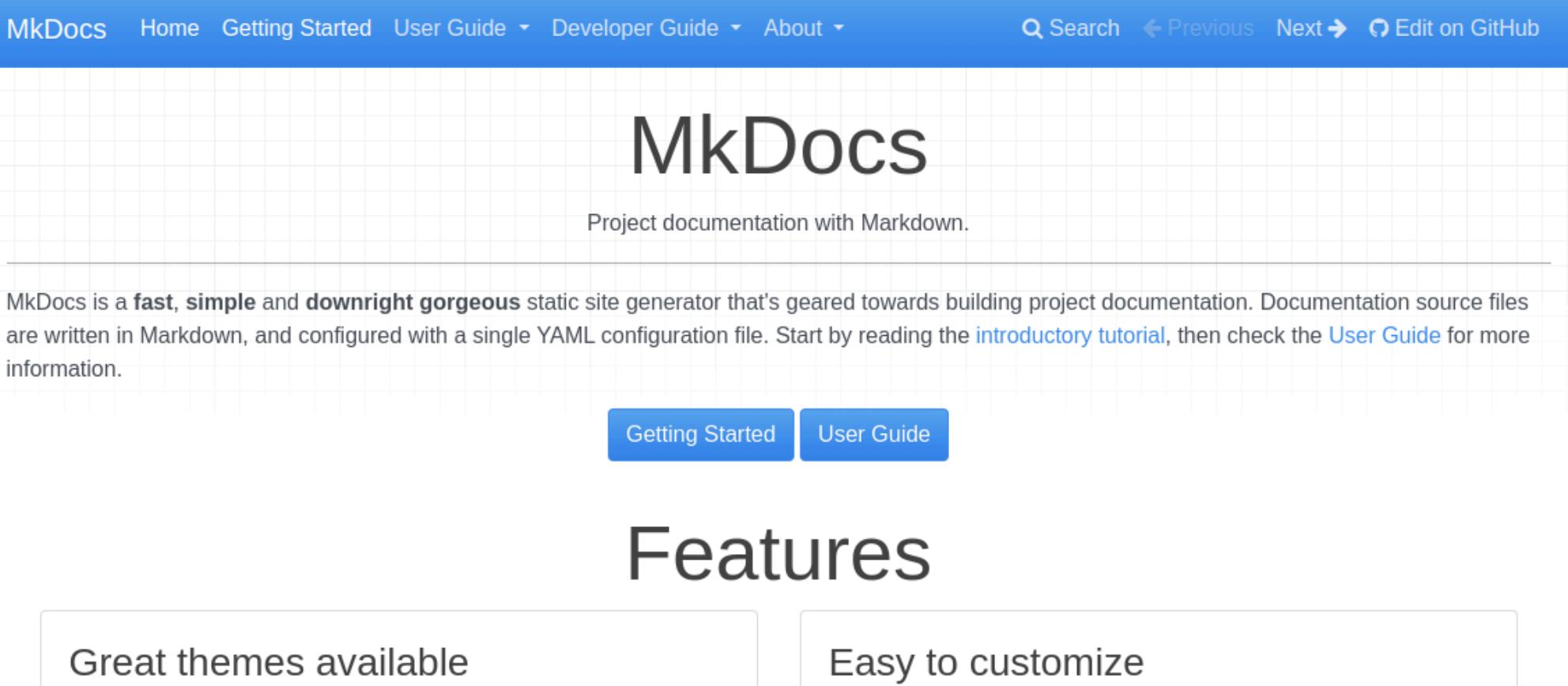
- 1. configuração do ambiente
- 2. documentação
- 3. logs
- 4. testes
- 5. formatadores
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

Documentação

- Quando não sabemos algo de algum projeto ou biblioteca vamos na documentação dela
 - fastapi, django, requests...
- Criar documentações é uma forma de repassar conhecimento
- A boa formatação da documentação ajuda nesse repasse



Documentação



A screenshot of the MkDocs documentation website. The header is blue with white text. It includes navigation links: MkDocs, Home, Getting Started, User Guide (with a dropdown arrow), Developer Guide (with a dropdown arrow), and About (with a dropdown arrow). To the right of these are search, previous, next, and GitHub edit links. The main content area has a grid background. It features a large title "MkDocs" and a subtitle "Project documentation with Markdown." Below this is a descriptive paragraph about the project. At the bottom, there are two blue buttons labeled "Getting Started" and "User Guide". The page also lists "Features" like "Great themes available" and "Easy to customize".

MkDocs Home Getting Started User Guide ▾ Developer Guide ▾ About ▾

Search ← Previous Next → Edit on GitHub

MkDocs

Project documentation with Markdown.

MkDocs is a **fast, simple and downright gorgeous** static site generator that's geared towards building project documentation. Documentation source files are written in Markdown, and configured with a single YAML configuration file. Start by reading the [introductory tutorial](#), then check the [User Guide](#) for more information.

[Getting Started](#) [User Guide](#)

Features

Great themes available

Easy to customize

Documentação

- □ ×

```
# instalação mkdocs  
poetry add --group dev mkdocs
```

```
# inicia nossa documentação  
mkdocs new .
```

```
# sobe a documentação  
mkdocs serve
```

Documentação

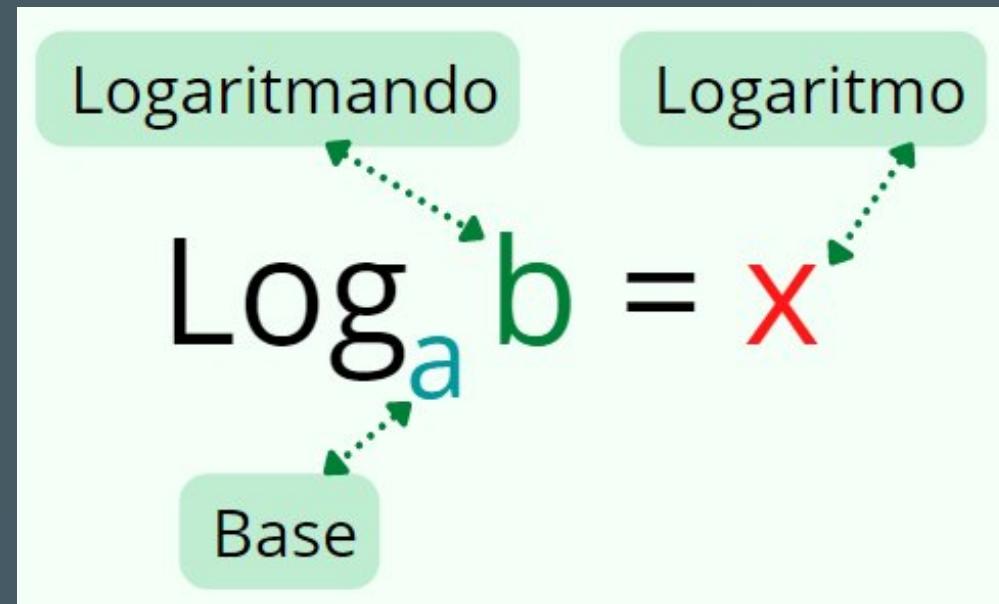
- Entendimento do sistema
- Criação de uma documentação simples

Em que passo estamos?

- 1. ~~configuração do ambiente~~
- 2. ~~documentação~~
- 3. logs
- 4. testes
- 5. formatadores
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

Logs

- O que são? Onde vivem? Como se alimentam?
- Não tem nada a ver com logaritmo



Logs

“ “ Expressão utilizada para descrever o processo de **registro** de **eventos** relevantes em um sistema computacional.¹ ” ”

- Registro: "escrever" ou "marcar" **algo** em **algum lugar**
- Eventos que aconteceram no **passado** e podem ser **observados**
- Exemplos de eventos relevantes: **login**, **logoff**, algum erro no sistema, sucesso em uma requisição externa
- Local do registro pode ser a saída do terminal, um arquivo, um servidor de email

1. https://pt.wikipedia.org/wiki/Log_de_dados

Logs

- □ ×

```
import logging

logging.warning("Cuidado! Pouco espaço em disco")
logging.error("Deu algo errado em")
logging.critical("DELETARAM O BANCO")

>>> WARNING:root:Cuidado! Pouco espaço em disco
>>> ERROR:root:Deu algo errado em
>>> CRITICAL:root:DELETARAM O BANCO
```

Logs: em que contribuem?

- Monitoramento da aplicação
 - Criando gráficos do comportamento do sistema
 - Entender o nível de criticidade do evento ocorrido
 - Auxílio para criar alertas



Logs



Loguru

Python logging made (stupidly) simple

Logs

- Adicionar logs...

Em que passo estamos?

- 1. ~~configuração do ambiente~~
- 2. ~~documentação~~
- 3. logs
- 4. testes
- 5. formatadores
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

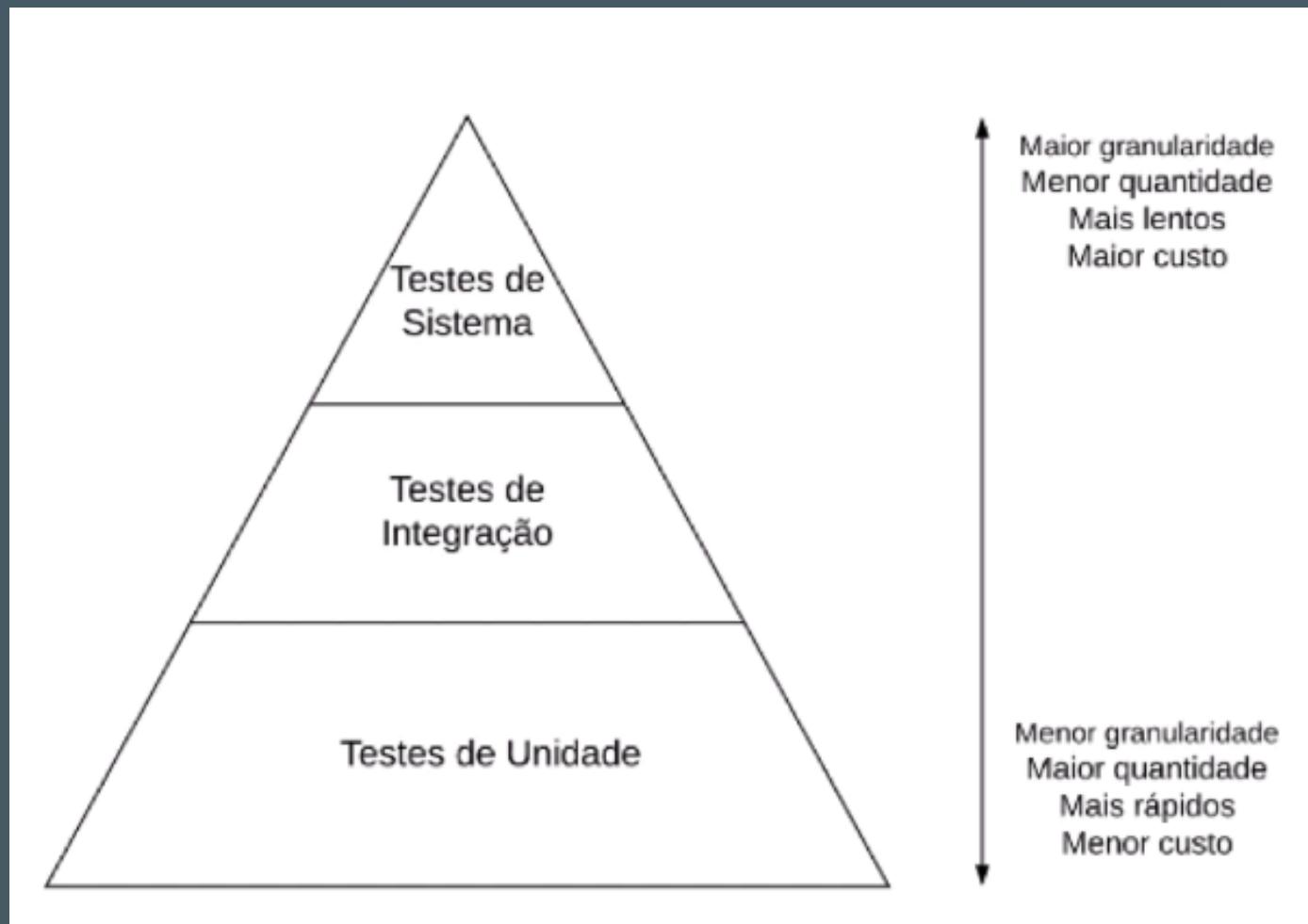
Testes

- Software é uma construção complexa
- Logo, está sujeita a erros
- Assim, foram criados testes para minimizar que erros cheguem ao usuário

Testes: em que contribuem?

- Previnem que bugs cheguem ao usuário (redução de custos)
- Auxiliam no entendimento do sistema
- Garantem o funcionamento do código depois de alterado (testes regressivos)

Tipos de testes



Testes

- Como fazer um bom teste?

Em que passo estamos?

- 1. ~~configuração do ambiente~~
- 2. ~~documentação~~
- 3. logs
- 4. testes
- 5. formatadores automáticos
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

"Estilos" de código

- Pessoas programam com "estilos" diferentes
- O código vai funcionar de um jeito ou de outro, mas possuem diferenças estéticas
 - Ordem de imports
 - " ou '?
 - Quebra de linhas diferentes
 - Tamanhos de linha sem padrão
 - ...

Contents

- Introduction
- A Foolish Consistency is the Hobgoblin of Little Minds
- Code Lay-out
 - Indentation
 - Tabs or Spaces?
 - Maximum Line Length
 - Should a Line Break Before or After a Binary Operator?
 - Blank Lines
 - Source File Encoding
 - Imports
 - Module Level Dunder Names
- String Quotes
- Whitespace in Expressions and Statements
 - Pet Peeves
 - Other Recommendations
- When to Use Trailing Commas
- Comments
 - Block Comments
 - Inline Comments
 - Documentation Strings

PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Alyssa Coghlan <ncoghlan at gmail.com>

Status: Active

Type: Process

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

► Table of Contents

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing [style guidelines for the C code in the C implementation of Python](#).

This document and [PEP 257](#) (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

Formatadores automáticos

- É uma boa prática definir um padrão para o estilo de escrita de código
 - legibilidade
 - acessibilidade
- Garantimos esse estilo com formatadores automáticos

Ruff

— □ ×

```
# instalação ruff
poetry add --group dev ruff

# Preview do que vai ser formatado
ruff format --check --diff

# formata automaticamente
ruff check --select I --fix
ruff format
```

Análise estática

- Nosso código pode conter diversos problemas como:
 - Erros de sintaxe
 - Potenciais problemas
 - Nomes duplicados
 - Nomes ruins
 - Códigos complexos
 - Violações de convenções (PEP-8)

Análise estática

- Para mitigar essas questões existem os linters
 - Flake8: pep8
 - pylint: Padronização de convenções
 - bandit: Problemas de segurança
 - Radon: complexidade de código

Ruff

- □ ×

```
# verifica quais regras infringidas  
ruff check
```

```
# corrige regras possíveis  
ruff check --fix
```

- Cada um escolher uma regra para infligir

Em que passo estamos?

- 1. configuração do ambiente
- 2. documentação
- 3. logs
- 4. testes
- 5. formatadores automáticos
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

Automatizando comandos

- Rodamos muitos comandos ao longo da curso
- Poderíamos criar comandos mais simples e fáceis de lembrar...
- Algumas bibliotecas nos ajudam nisso:
 - <https://github.com/taskipy/taskipy>
 - <https://github.com/nat-n/poethopoet>
 - <https://github.com/pyinvoke/invoke>

Em que passo estamos?

- 1. configuração do ambiente
- 2. documentação
- 3. logs
- 4. testes
- 5. formatadores automáticos
- 6. análise estática
- 7. automação de comandos
- 8. automação com git hooks

Git

- Ferramenta de controle de versão

```
# Resumo rápido de git

git add .          # Prepara as modificações
git commit -m "mensagem" # Grava as modificações
git push          # Manda modificações pro github
```

Automação com git hooks

- O que são?
- No que contribuem para o nosso sistema?