Fine-Grained Analysis of Software Ecosystems as Networks

LOGS: A arte de fazer o software contar sua história

Estudo de caso do projeto FASTEN

Ingrid Sena

Python Brasil 15
Brasil, 27 Out 2019

Ingrid Sena

- Open Source Compliance Software Engineer
- 29 anos
- Berlim, Alemanha

LOGS: A arte de fazer o software contar sua história

Quem somos

- O que é FASTEN
  - https://www.fasten-project.eu/
  - TUDelf, AUEB, UNIMI, OW2, Endocode, SIG, XWIKI
- Quem é a ENDOCODE
- O que é QUARTERMASTER
  - OSS
  - Licenças e Copyrights

Motivação

- Iniciando no projeto
- Projeto muito grande
- Vários containers
- Nova linguagem

Definições

- O que é *print( )*?
  - é uma função que imprime uma mensagem específica em uma saída padrão.

- O que é debugger (depurador)?
  - é uma ferramenta usada para analisar e testar manualmente o seu sistema.

Print() vs PDB (Debugger)

código fonte

```python
def ePrimo(num):
    if num % 2 == 0:
        return num == 2
    iter_div = 3
    while iter_div * iter_div <= num:
        if num % iter_div == 0:
            return False
        iter_div += 2
    return True


print(f"## é primo? {ePrimo(31)}\n")

```

Print() vs PDB (Debugger)

com *prints*

```python
def ePrimo(num):
    print(f"Chamando ePrimo com o argumento num: {num}")

    if num % 2 == 0:
        result = num % 2
        print(f"O resto da divisão de {num} por 2 (módulo) é: {result}")
        return num == 2

    iter_div = 3
    print(f"Iterando com iter_div: {iter_div}")
    while iter_div * iter_div <= num:
        print("Enquanto iter_div * iter_div <= num, continue...")
        if num % iter_div == 0:
            print("Se num % iter_div for igual a zero, então retorne falso")
            return False
        print("Incrementando iter_div + 2")
        iter_div += 2
        print(f"Agora o valor de iter_div é: {iter_div}")

    return True


print(f"## é primo?: {ePrimo(31)}\n")

```

Print() vs PDB (Debugger)

com PDB

```python
def ePrimo(num):
    breakpoint()
    if num % 2 == 0:
        return num == 2
    iter_div = 3
    while iter_div * iter_div <= num:
        if num % iter_div == 0:
            return False
        iter_div += 2
    return True


print(f"## é primo? {ePrimo(31)}\n")

```

Definições

- O que é logger?
  - "...talvez a abstração mais simples de armazenamento. E um arquivo *append-only*, sequência de registros ordenada pelo tempo." - Jay Kreps
  - "Te ajudara a desenvolver um melhor entendimento do fluxo do seu programa e descobrir cenários que você provavelmente nunca previu." - Real Python

ERROS ≠ FALHAS

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Ok, mas... E OS TESTES?

Ok, mas... E OS TESTES?

Ok, mas... E OS TESTES?

LOGS e Testes

DESENVOLVIMENTO

TESTES

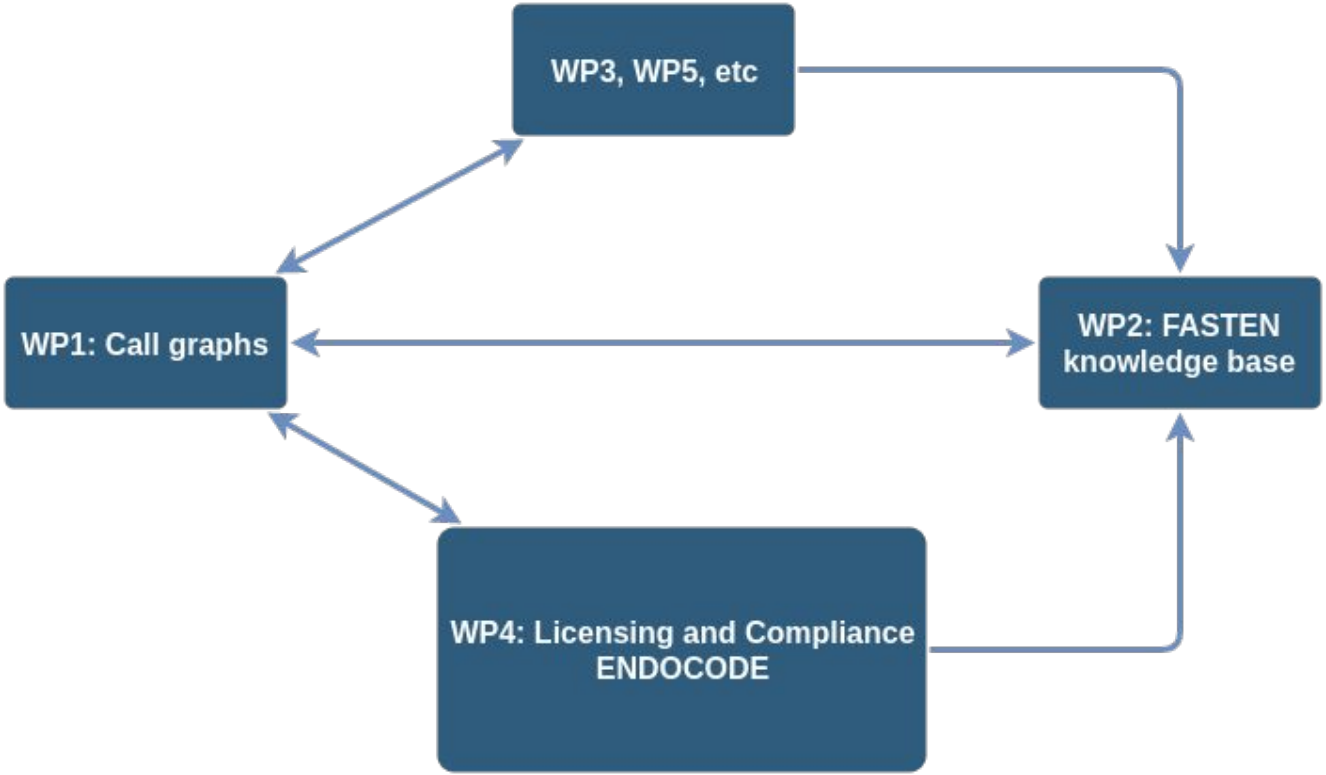
PRODUÇÃO

LOGS

Exemplo de tipos de Logs

- log de aplicação
- log de servidor
- logs para replicação de dados
- logs de cliques/atividades
- log de transações
- etc

QUARTERMASTER



Tá, mas e agora?

- Vazamento de dados
- Responsabilidade Social

- GDPR - *General Data Protection Regulation*
- CCPA - *California Consumer Privacy Act*
- LGPD - *Lei geral de Proteção de Dados*

BIBLIOGRAFIA

[EN] Blogpost: "The Log: What every software engineer should know about real-time data's unifying abstraction" (Jay Kreps - 2013)

[EN] Palestra: "Python Logging: A meditation on silent failures" (Jess Unrein - PyOhio 2016)

[EN] Blog Real Python: "Logging in Python" (Abhinav Ajitsaria - 2018)

[PTBR] Manifesto Os Doze Fatores "Logs: Trate logs como fluxos de eventos"

[PTBR] Site oficial do governo sobre a "Lei Geral de Proteção de Dados Pessoais"

OUTROS LINKS

[EN] Blogpost: "Logs were our lifeblood. Now they're our liability." (Vicki Boykis - 2019)

[EN] Palestra: "A guided tour of Python logging" (Curtis Maloney - PyCon AU 2018)

[EN] Blog Real Python: "Python Logging: A Stroll Through the Source Code" (Brad Solomon - 2019)

[EN] DataDog Blogpost: "How to collect, customize, and centralize Python logs" (Bunge and Chang - 2019)

[EN] Python 3.6 official documentation "Logging HOWTO"

github, gitlab: **@senaingrid**

twitter: **@senaingrid90**

The FASTEN project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825328.