Основы дебаггинга и профилирования

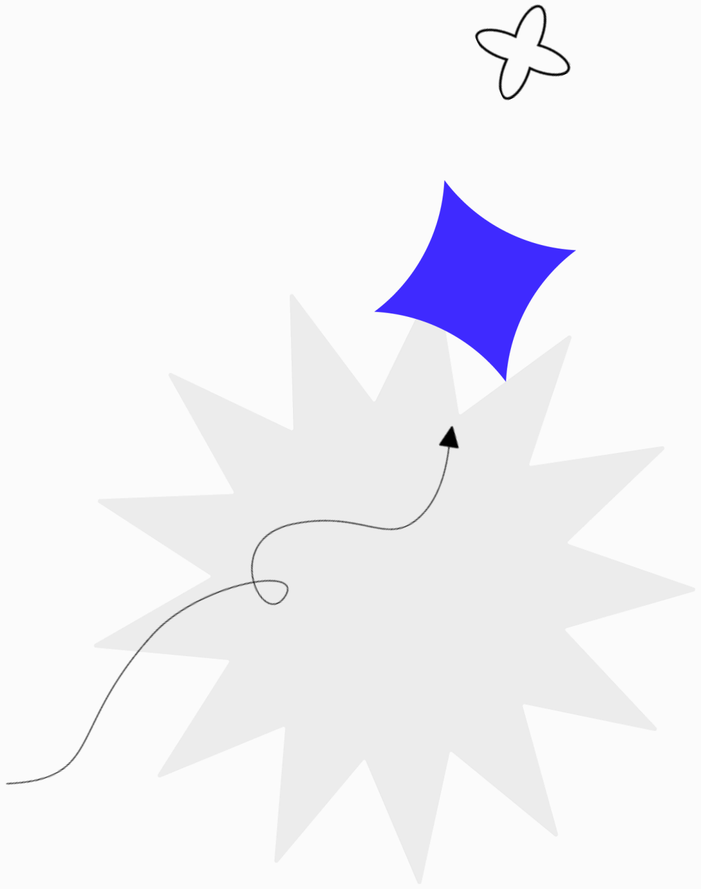
Никита Нестеренко
Руководитель проекта и главный инженер
по разработке в СБЕР

Skillbox

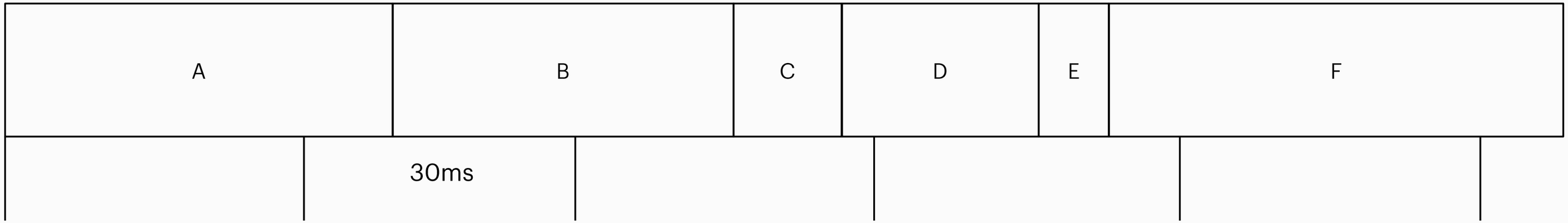
Профилирование

Метрики выполнения кода

- Время выполнения функций
- Количество вызовов функций
- Дерево вызовов функций
- Загрузка CPU и потребление памяти
- Обращения к другим ресурсам сервера



Статистические профайлеры



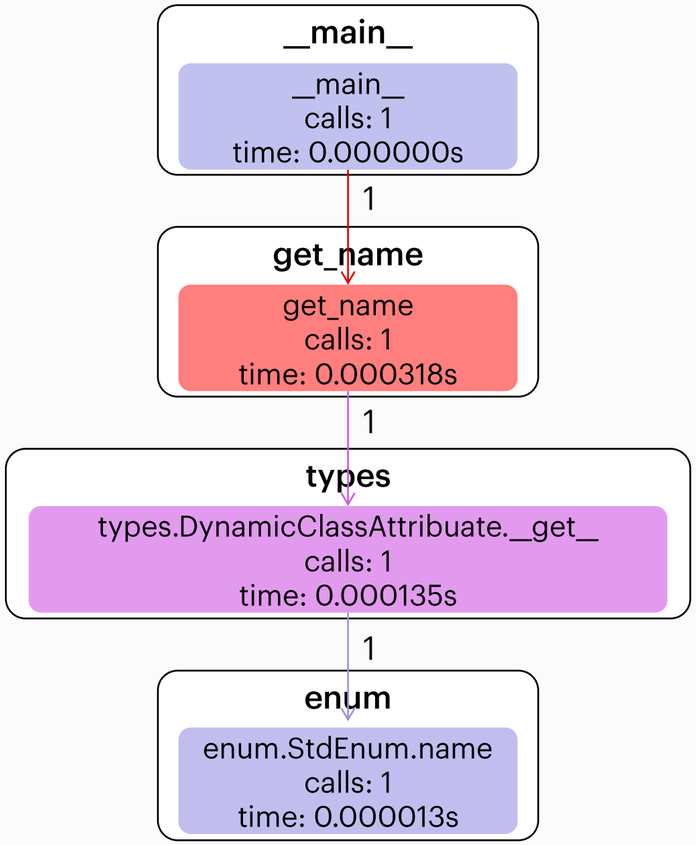
Достоинства профайлеров

- Минимальное влияние на программу
- Простые и универсальные в использовании
- Определение тяжёлых и часто используемых функций

Недостатки профайлеров

- Точность анализа зависит от количества измерений и временного интервала
- Результаты анализа содержат неполную информацию
- Для сбора точной статистики может потребоваться длительное время
- Малое количество инструментов для анализа

Событийные профайлеры



Достоинства профайлеров

- Подробная информация о работе программы
- Не требуется написание кода для анализа
- Обширное количество инструментов профилирования

Недостатки профайлеров

- Дополнительные накладные расходы

Ручное профилирование

```python
import time

def profiler(func):
    def wrapper(*args, **kwargs):
        before = time.time()
        f = func(*args, **kwargs)
        after = time.time()
        print(after - before)

    return wrapper

@profiler
def hello_guys():
    print("Hello guys")

if __name__ == '__main__':
    hello_guys()
```