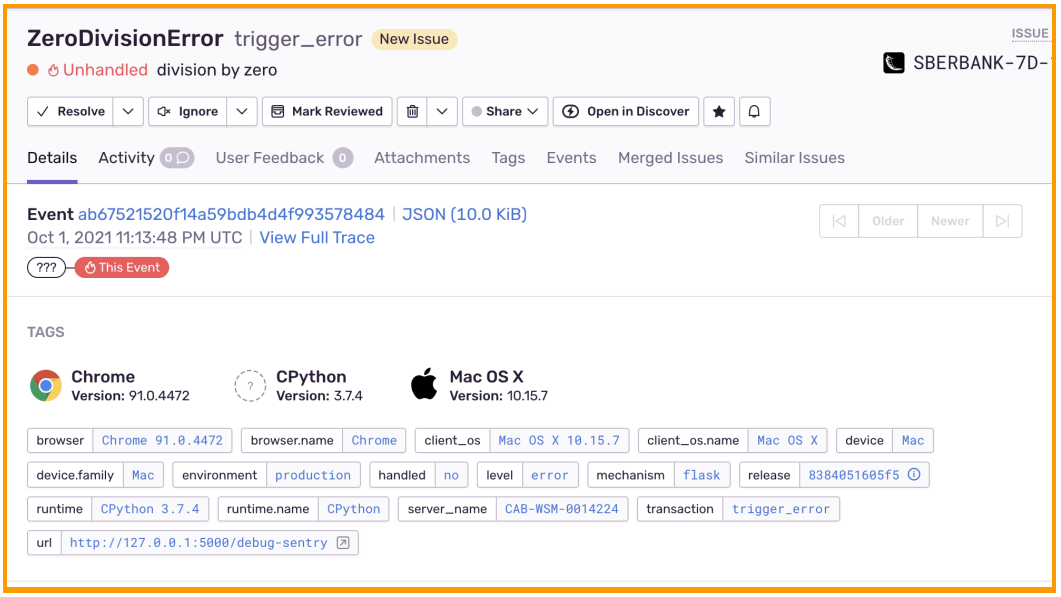
# Урок 3. Sentry — инструмент мониторинга ошибок

Sentry — это система оперативного мониторинга ошибок. Пользовательский интерфейс представляет собой dashboard со списком ошибок и возможностью выполнять над ними различные действия.

Зайдем на сайт Sentry, в раздел Python: https://docs.sentry.io/platforms/python/. Как и написано в документации, для начала необходимо создать учётную запись и sentry-проект.

```
from flask import Flask, request
import sentry_sdk
from sentry_sdk.integrations.flask import FlaskIntegration
import logging


sentry_sdk.init(
    dsn="https://yourtoken@o1022719.ingest.sentry.io/5999691",
    integrations=[FlaskIntegration()],

    # Set traces_sample_rate to 1.0 to capture 100%
    # of transactions for performance monitoring.
    # We recommend adjusting this value in production.
    traces_sample_rate=1.0
)

app = Flask(__name__)


@app.route('/debug-sentry')
def trigger_error():
    division_by_zero = 1 / 0


@app.route('/test_type')
def test_type():
    user_id = request.args.get('user_id')
    user_id = float(user_id)


@app.route('/test')
def ll():
    raise IndexError
```

```
@app.route('/test_logging')
def test_logging():
    logging.error("error to log")


if __name__ == '__main__':
    app.run()
```

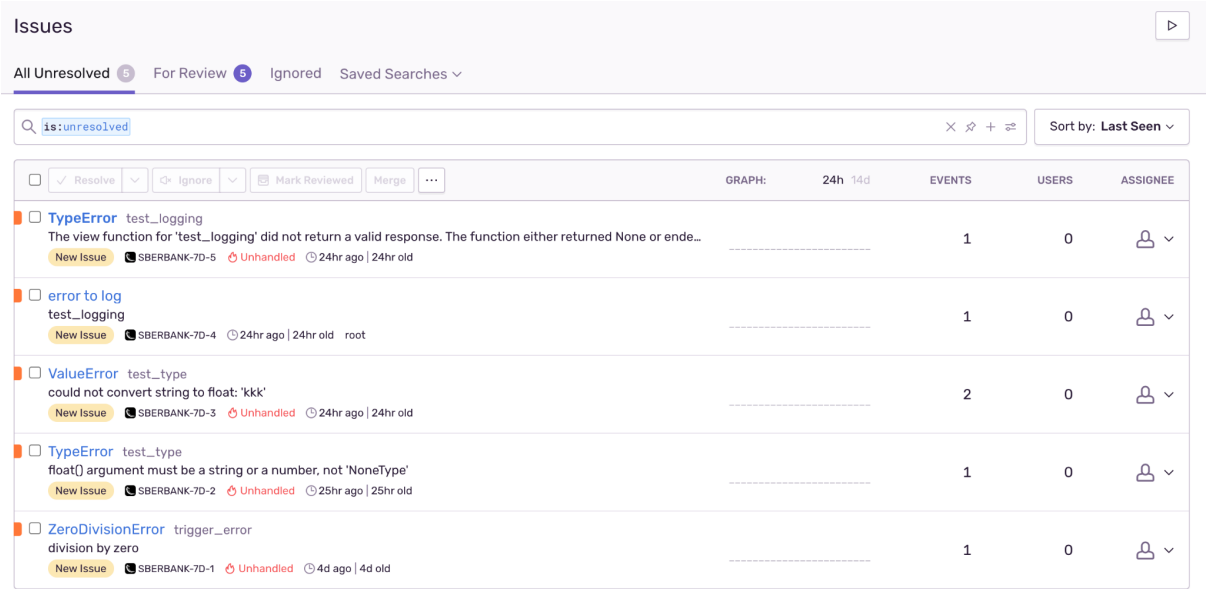



Одно из ключевых преимуществ относительно логов — это хранение состояния переменных в момент ошибки: мы можем сразу понять, какая ситуация вызвала исключение. Особенно это актуально, если человек ещё не сильно погружен в проект и не знает структуру кода. В случае лога мы бы увидели только traceback, пошли бы на локальную машину и там дебажили проект, пробуя подавать на вход разные данные.

Ещё один очевидный плюс — удобная фильтрация ошибок. Так, например, в случае повторения ошибки, Sentry добавит тикету ошибки в счётчик +1. Таким образом, мы не будем засорять интерфейс кучей одинаковых ошибок. Ошибки, которые фиксирует Sentry, можно отправлять на почту, в Telegram-бота или Slack. В случае повторных ошибок оповещение отправлено не будет, и, если вы считаете, что этот баг не такой важный, ошибка будет просто накручивать счётчик.

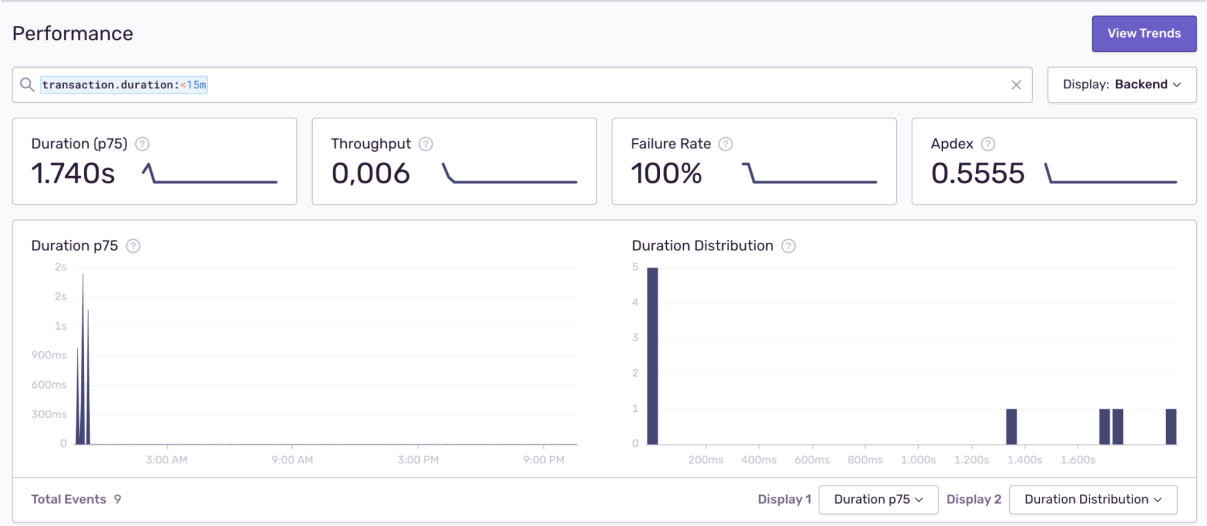
Здесь есть два решения: править все баги и не оставлять на потом или ставить autoresolve через какой-то промежуток времени, чтобы ошибки закрывались и не скапливались.

Sentry сразу же интегрируется с модулем logging, то есть при вызове лога в коде он также будет отражен в интерфейсе Sentry.




Что касается мониторинга производительности, в интерфейсе Sentry, во вкладке Performance, мы видим графики производительности наших роутов. С помощью мониторинга производительности Sentry отслеживает производительность приложений, измеряет такие показатели, как пропускная способность и задержка, а также отображает влияние ошибок в нескольких службах.

В интерфейс выводится четыре ключевых показателя: Duration (показывает длительность, быстрее которой выполняется определённая часть запросов), Throughput (пропускная способность, количество зарегистрированных транзакций в минуту), Failure Rate (интенсивность отказов, процент зарегистрированных транзакций, которые были зафиксированы в Sentry и имели неуспешный статус) и метрика Apdex (стандартная метрика, используемая для отслеживания и измерения удовлетворённости пользователей на основе времени отклика вашего приложения). Оценка Apdex показывает соотношение удовлетворительных, допустимых и неудовлетворённых запросов в конкретной транзакции или конечной точке. Чем выше метрика, тем удовлетвореннее наши пользователи.




Плюсы Sentry:

- легко интегрировать в проект,
- поддержка большого количества инструментов,
- realtime-фиксация ошибок приложения,
- мониторинг производительности запросов,
- система оповещений по email, Telegram, Slack,
- отображение traceback и состояния переменных.