МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

СЕРГИЕНКО ЛЕВ ЭДУАРДОВИЧ

Отчет по
ЛАБОРАТОРНАЯ РАБОТА 3
ПО ДИСЦИПЛИНЕ
«Непрерывное интегрирование и сборка программного обеспечения»

Мониторинг в распределенных системах

Преподаватель

Давидовская М.И.
Филиппов М.А.

2025

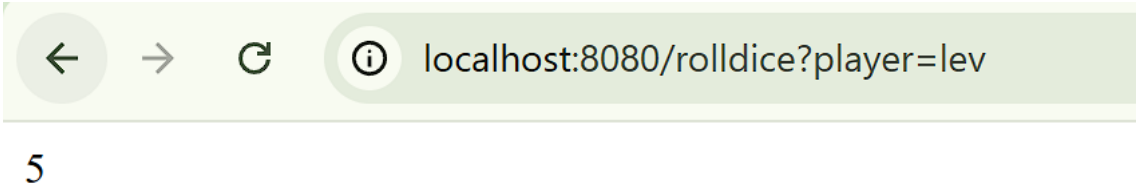
1. Цель работы.

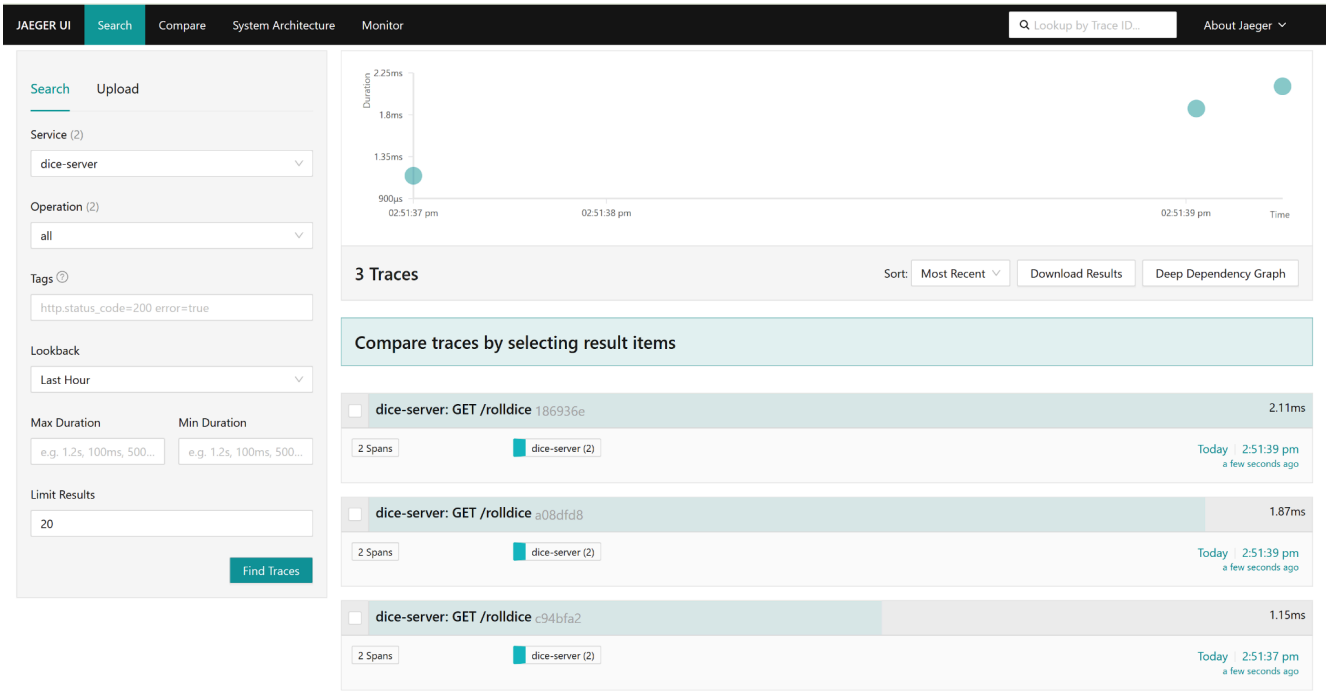
Изучение средств мониторинга для сбора и обработки телеметрии приложений и настройка системы, в которой данные метрик, журналов и трассировок собираются для микросервисного приложения на языке Python. Для демонстрации этапов интеграции OpenTelemetry в Python мы будем использовать простое приложение, реализованное с помощью программного средства разработки Flask. Первая версия данного приложения основывается на официальной документации.

2. Вариант задания.

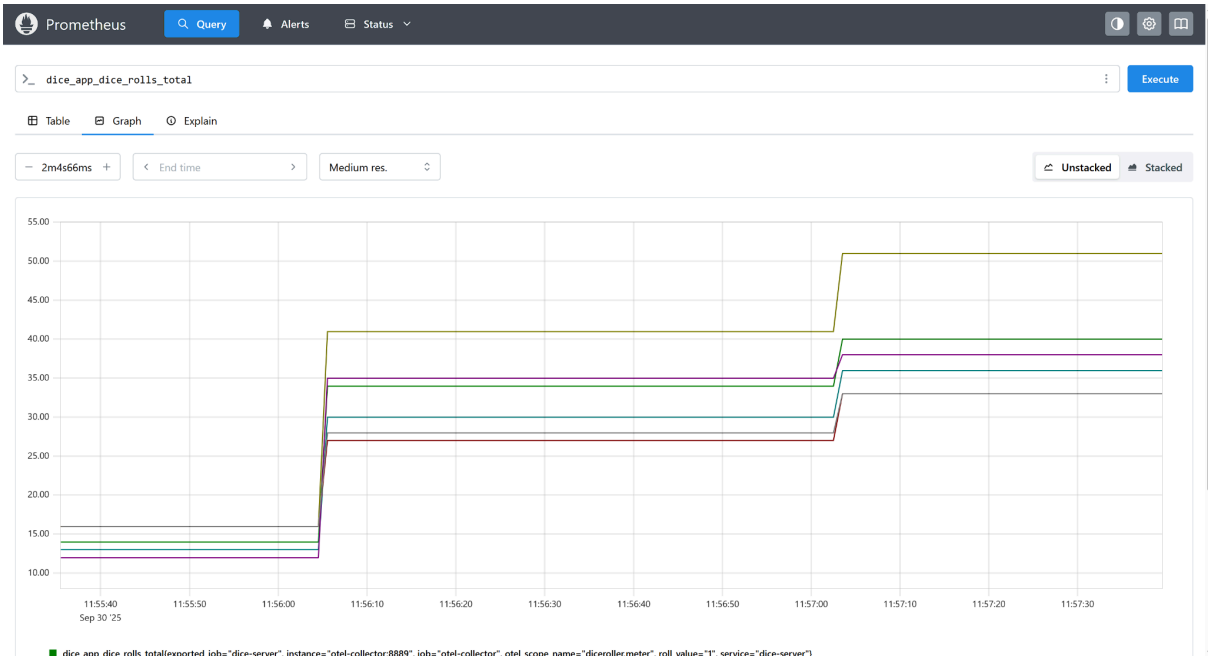
3. Код приложений, конфигурационных файлов.

Задание 1. Подключение инструментов и библиотек OpenTelemetry и настройка трассировки приложения

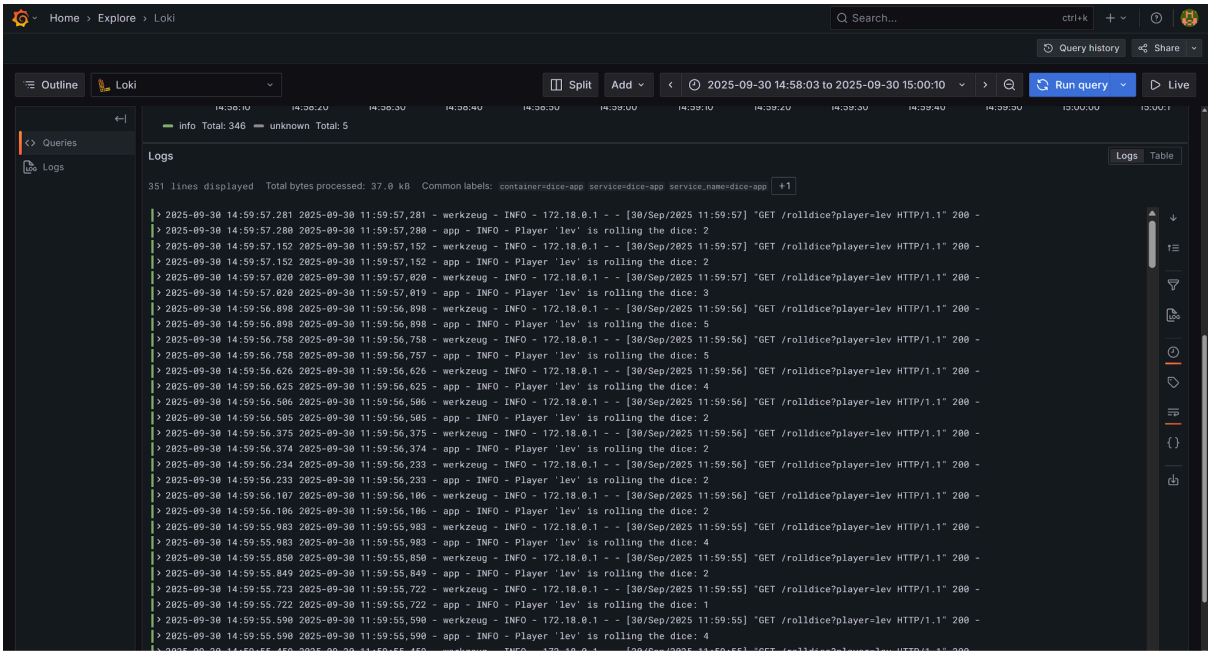


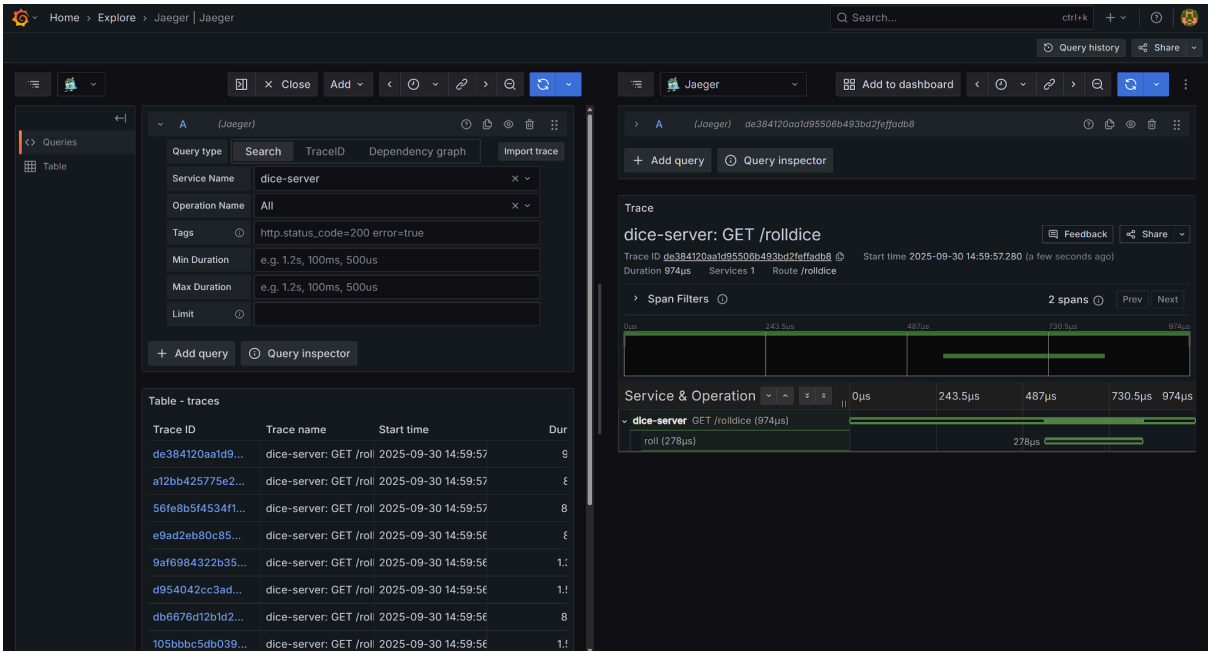


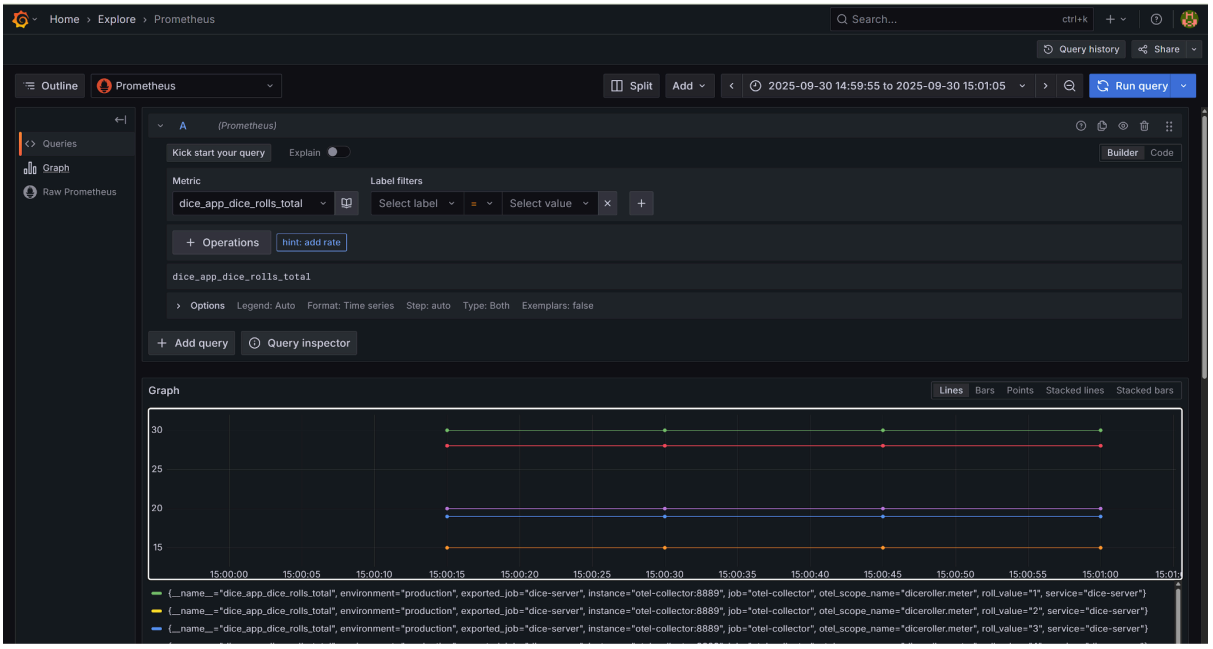
Задание 2. Настройка сбора метрик приложения



Задание 3. Анализ журналов, визуализация и исследование телеметрии







4. Ответы на контрольные вопросы.

1. **Что такое OpenTelemetry? Атрибуты, события, контекст, журналы, трассировки, показатели**

OpenTelemetry - CNCF-проект: набор спецификаций, SDK, API и инструментов для генерации, сбора, обработки и экспорта телеметрии (трассировок, метрик, логов).

Атрибуты (attributes) - ключ/значение, сопровождают спаны/метрики/логи.

События (events) - временные записи внутри спана (например, исключение, checkpoint).

Контекст (context / context propagation) - идентификаторы трассы/спана и метаданные, которые передаются между сервисами для корреляции запросов.

Журналы (logs) - текстовые записи событий; в OpenTelemetry стремятся связывать логи с трассировками/метриками.

Трассировки (traces) - последовательность спанов, описывающих обработку одного запроса/транзакции.

Метрики (metrics) - агрегированные числовые показатели (счётчики, гейджи, гистограммы и т.д.).

2. **Что такое наблюдаемость? Надежность и показатели**

Наблюдаемость - способность понять внутреннее состояние системы по ее внешним сигналам (логи, метрики, трассы); помогает находить причину инцидентов, не только увидеть их.

Надежность (reliability) - свойство системы работать корректно в заданных условиях; наблюдаемость повышает надежность (быстрее обнаружение и восстановление).

Ключевые показатели: SLO/SLI/SLA (latency, error rate, availability), время восстановления (MTTR), частота инцидентов и загрузка ресурсов.

3. **Что такое трассировка? Что такое распределенная трассировка? Как собирать и какие проблемы в распределенной архитектуре?**

Трассировка - запись последовательности операций (спанов) при обработке запроса внутри/между процессов.

Распределенная трассировка - объединение спанов через микросервисы, контейнеры и процессы, чтобы проследить весь путь запроса. Контекст (trace id, span id) передаётся по сети (headers).

Как собирать: инструментировать код (ручной SDK) или использовать автоинструментацию, прокси/агенты или OpenTelemetry Collector; передавать контекст через HTTP/gRPC/месседж-заголовки.

Проблемы: потеря контекста (mis-propagation), различия в форматах/таймингах, overhead / нагрузка, высокое количество данных (sampling/retention), согласованность времён (clock skew), безопасность заголовков.

4. **Сигналы OpenTelemetry: трассировки, метрики, логи**

OpenTelemetry стандартизирует три основных сигнала: traces, metrics, logs. Каждый сигнал имеет свои модели данных (спаны/атрибуты,

метрика->семья типов, лог->строка + labels) и их связывают через resource и контекст.

5. **Инструменты OpenTelemetry: автоматические, ручные, библиотеки**

Автоинструментация - агенты/инструменты, которые вставляют трассировку/метрики без правок в приложение (например, instrumentation agents, auto-instrumentation libs).

Ручная (manual) инструментализация - использование SDK/API (создать спаны, выставить атрибуты, инкрементировать метрики).

Библиотеки/SDK - язык-специфичные пакеты (Java, Python, Go, JS и т.д.), плагины для фреймворков, интеграции с HTTP/DB/GRPC.

6. **Компоненты OpenTelemetry: спецификация, сборщики, библиотеки инструментальных средств, экспортеры, автоматические измерительные инструменты**

Спецификация - формальные описания сигналов, semantic conventions и протоколов.

Collector - vendor-agnostic агент/сервер для приёма, обработки (sampling, batching, enrichment) и экспорта телеметрии.

SDK/инструменты - language SDKs, auto-instrumentation, semantic conventions.

Экспортеры - плагины/модули, отправляющие данные в бекэнды (OTLP, Jaeger, Prometheus, commercial backends).

Автоматические измерительные инструменты - интеграции, экспортеры метрик, профайлеры, sidecars.

7. **Ресурс OpenTelemetry (resource / телеметрия)**

Resource - набор атрибутов, описывающих сущность, которая генерирует телеметрию (service.name, service.version, host.id). Ресурсы присоединяются ко всем сигналам для корректной маршрутизации и агрегации.

8. **Экспортеры OpenTelemetry**

Модули, переводящие и отправляющие телеметрию в конкретный бекэнд: OTLP (универсальный), Jaeger, Zipkin, Prometheus

(экспонирование метрик), коммерческие экспортеры (Datadog, New Relic и др.). Экспортеры могут быть встроены в SDK или выполняться в Collector.

9. **Что такое Prometheus? Как работает Prometheus?**

Prometheus - система мониторинга и база временных рядов; полюбилась за простоту, модель pull и язык запросов PromQL.

Как работает: Prometheus периодически scrape (опрашивает) HTTP-эндпоинты, которые экспонируют метрики в Prometheus exposition format; собранные метрики хранятся как временные ряды и доступны для запросов/алертинга.

10. **Концепции Prometheus**

Pull-модель (scraping), targets, metrics exposition, time series (metric name + labels), PromQL (язык запросов), rules/alerting, retention & TSDB, pushgateway (для кратковременных/пушевых задач).

11. **Типы метрик**

- **Counter** - монотонно растущий счётчик (requests_total).
- **Gauge** - текущее значение (memory_usage, temperature).
- **Histogram** - распределение значений + суммарное значение/количество (latency buckets).
- **Summary** - квантильные сводки (p95/p99) с локальным накоплением (summary vs histogram различаются по способу агрегации и лейблам).

12. **Запрос и преобразование данных Grafana**

Grafana выполняет запросы к источнику данных (каждый data source имеет свой язык/редактор).

Трансформации: переименование полей, join/merge наборов, математические операции, агрегирование и фильтрация - всё это делается на этапе преобразования перед визуализацией.

13. **Источники данных Grafana**

Поддерживает множество: **Prometheus, Loki, Tempo, InfluxDB, Graphite, Elastic, MySQL/Postgres, Cloud vendors, OpenTelemetry/OTLP**

(через плагин/collector) и т.д. (каждый источник реализует свои query capabilities).

14. **Панели мониторинга Grafana**

Dashboard состоит из панелей (graph, stat, table, logs, heatmap и т.д.). Панели связываются с запросами к источникам и трансформациями; поддерживают переменные (templating), аннотации, alerting и совместное использование.

15. **Процесс обработки журналов Grafana Loki**

Loki хранит логи как наборы строк с **labels** (метки). Процесс: *приём → индексация только меток (не содержимого) → хранение чанков с логами → запрос по лейблам и фильтрам*; цель - экономия индекса и экономичное хранение по сравнению с полным индексированием.

16. **Варианты сборки журналов для Grafana Loki**

- **Promtail/ALLOY**
- **Fluentd / Fluent Bit** - универсальные шипперы, хорошо масштабируются; часто используются вместо Promtail.
- **Vector, Filebeat, Logstash** - другие варианты; также можно отправлять лог-стримы через HTTP/gRPC в Collector, который далее экспортирует в Loki.