ТЕСТИРОВАНИЕ В ФИНТЕХЕ:

НА ГРЕБНЕ ВОЛНЫ ИМПОРТОЗАМЕЩЕНИЯ

Андрей Ахметов

РСХБ РСХБ цифра

Что такое финтех?

РСХБ РСХБ цифра

Что такое финтех?

Финтех — это про удобство распоряжения деньгами в банке

Что такое финтех?

Финтех — это про удобство распоряжения деньгами в банке...

Что такое финтех?

Финтех — это про удобство распоряжения деньгами в банке...

Для пользователей

Как вы видите банк

Банк раньше

Как вы видите банк

Банк раньше

Банк сейчас

РСХБ РСХБ цифра

Что такое финтех?

Финтех — это про удобство распоряжения деньгами в банке...

Для пользователей

Для сотрудников

Как мы видим банк

Финтех-системы и команды в банке

Финтех-системы и команды в банке

ДБО ФЛ

РСХБ

РСХБ цифра

Финтех-системы и команды в банке

ДБО ФЛ

АБС

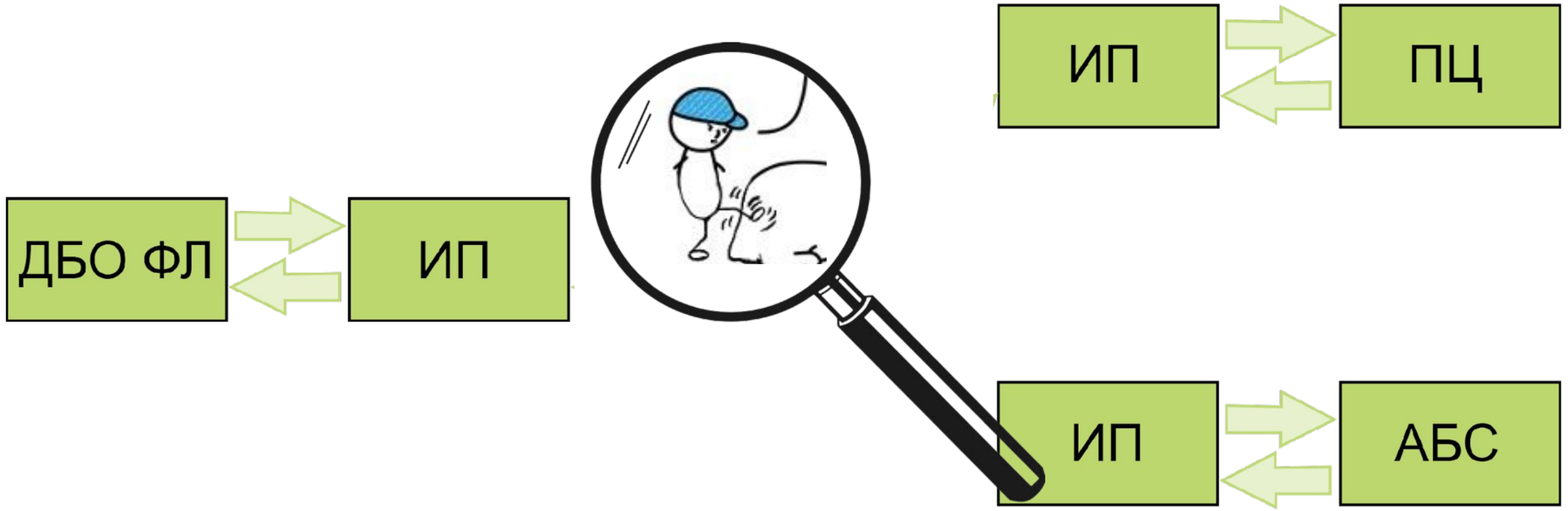
Финтех-системы и команды в банке

ДБО ФЛ

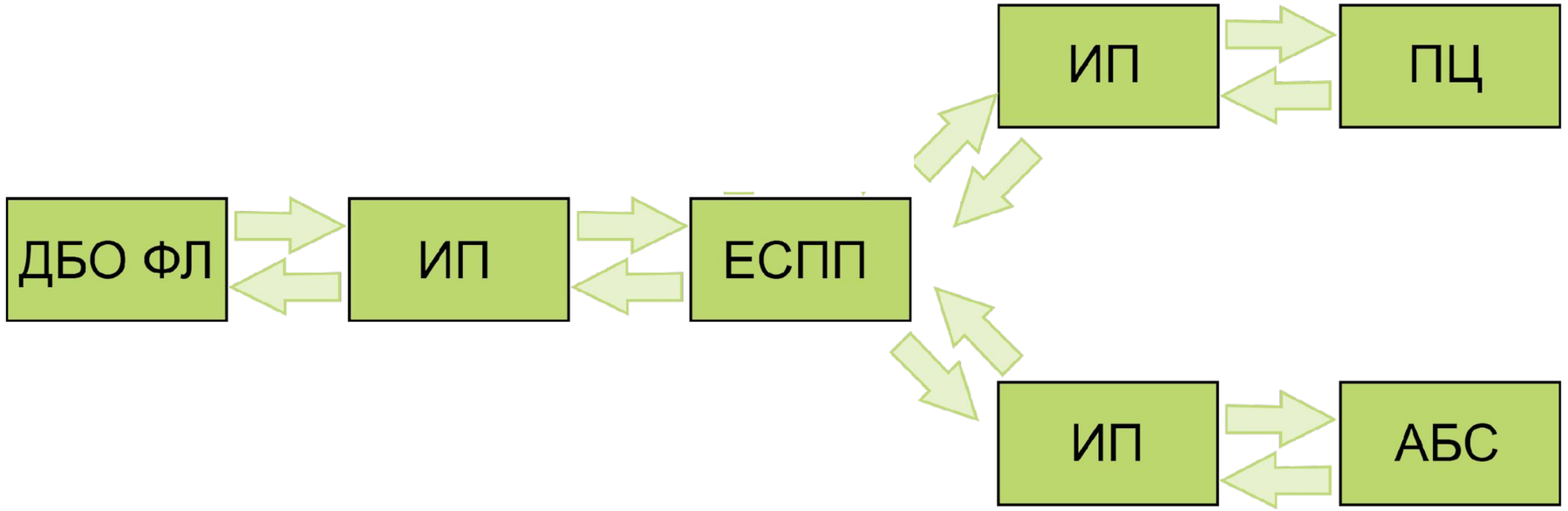
ПЦ

АБС

Финтех-системы и команды в банке



Финтех-системы и команды в банке



ЕСПП — комплексная система для обработки платежей

ЕСПП — комплексная система для обработки платежей

- Создание и запуск ЕСПП в 2017 году

ЕСПП — комплексная система для обработки платежей

- Создание и запуск ЕСПП в 2017 году
- ЕСПП начиналось с коробочного решения, которое было закуплено для обработки платежей

ЕСПП — комплексная система для обработки платежей

- Создание и запуск ЕСПП в 2017 году
- ЕСПП начиналось с коробочного решения, которое было закуплено для обработки платежей
- Со временем система эволюционировала до комплексного решения, способного обрабатывать транзакции с разных каналов: интернет-банкинг, банкоматы, мобильные устройства

ЕСПП — комплексная система для обработки платежей

Сложности, которые возникли в процессе роста:

- Появление новых типов платежей

ЕСПП — комплексная система для обработки платежей

Сложности, которые возникли в процессе роста:

- Появление новых типов платежей
- Увеличение количества транзакций, которые тянут за собой

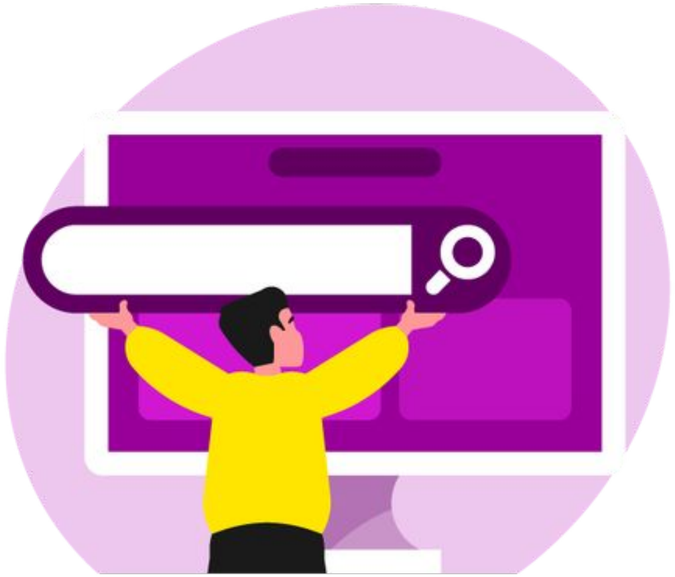
ЕСПП — комплексная система для обработки платежей

Сложности, которые возникли в процессе роста:

- Появление новых типов платежей
- Увеличение количества транзакций, которые тянут за собой
- Новые интеграции и новые цепочки взаимодействий

ДБО в процессе тестирования важно

убедиться, что нужный элемент интерфейса появляется в нужное время в процессе оплаты, их больше волнует дизайн и верстка.



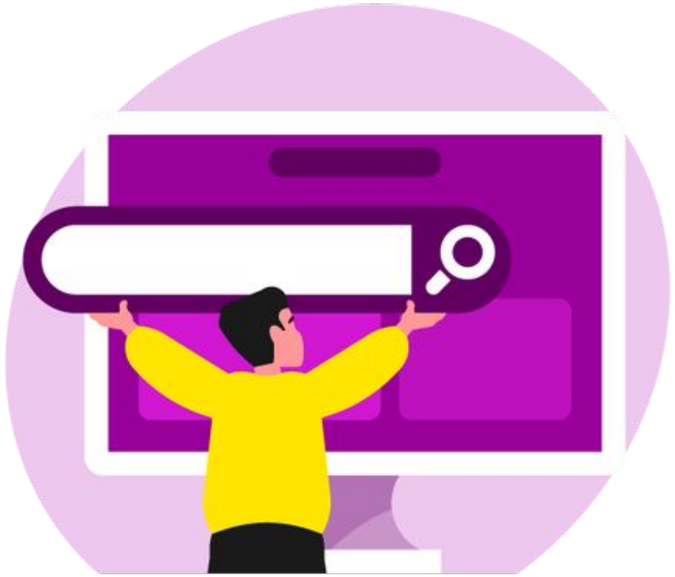
ЕСПП

ДБО в процессе тестирования важно

убедиться, что нужный элемент интерфейса появляется в нужное время в процессе оплаты, их больше волнует дизайн и верстка.

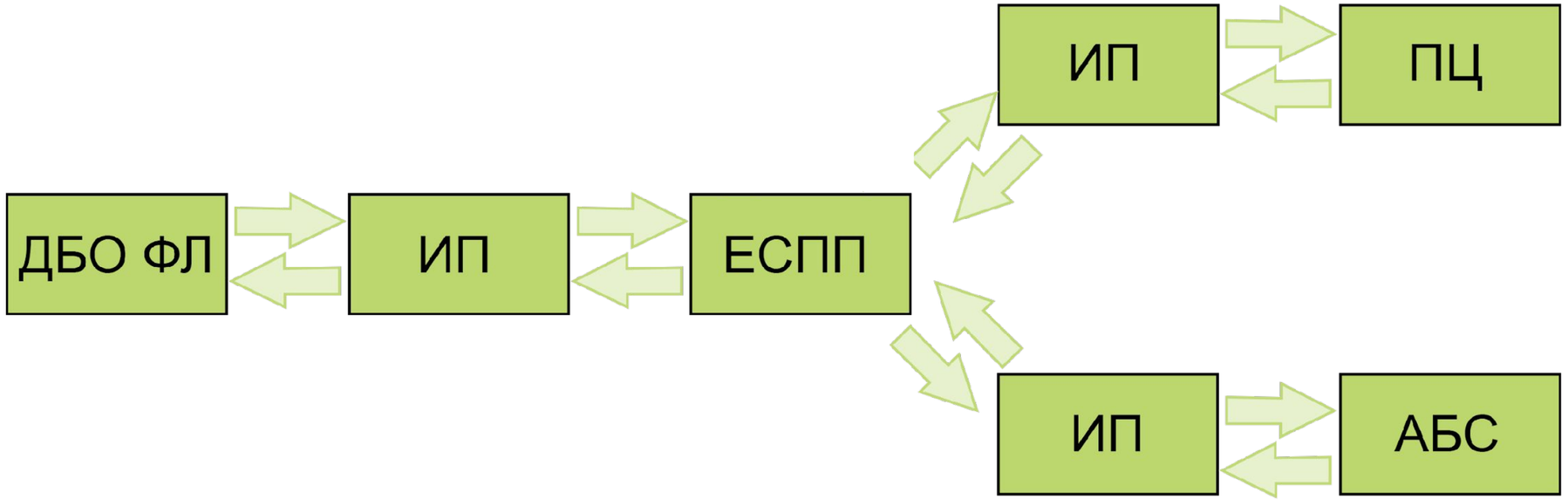
в процессе тестирования важно **АБС**

пришли корректные счета и правильно сформированные бухгалтерские проводки, важна скорость с которой их система отвечает на запросы

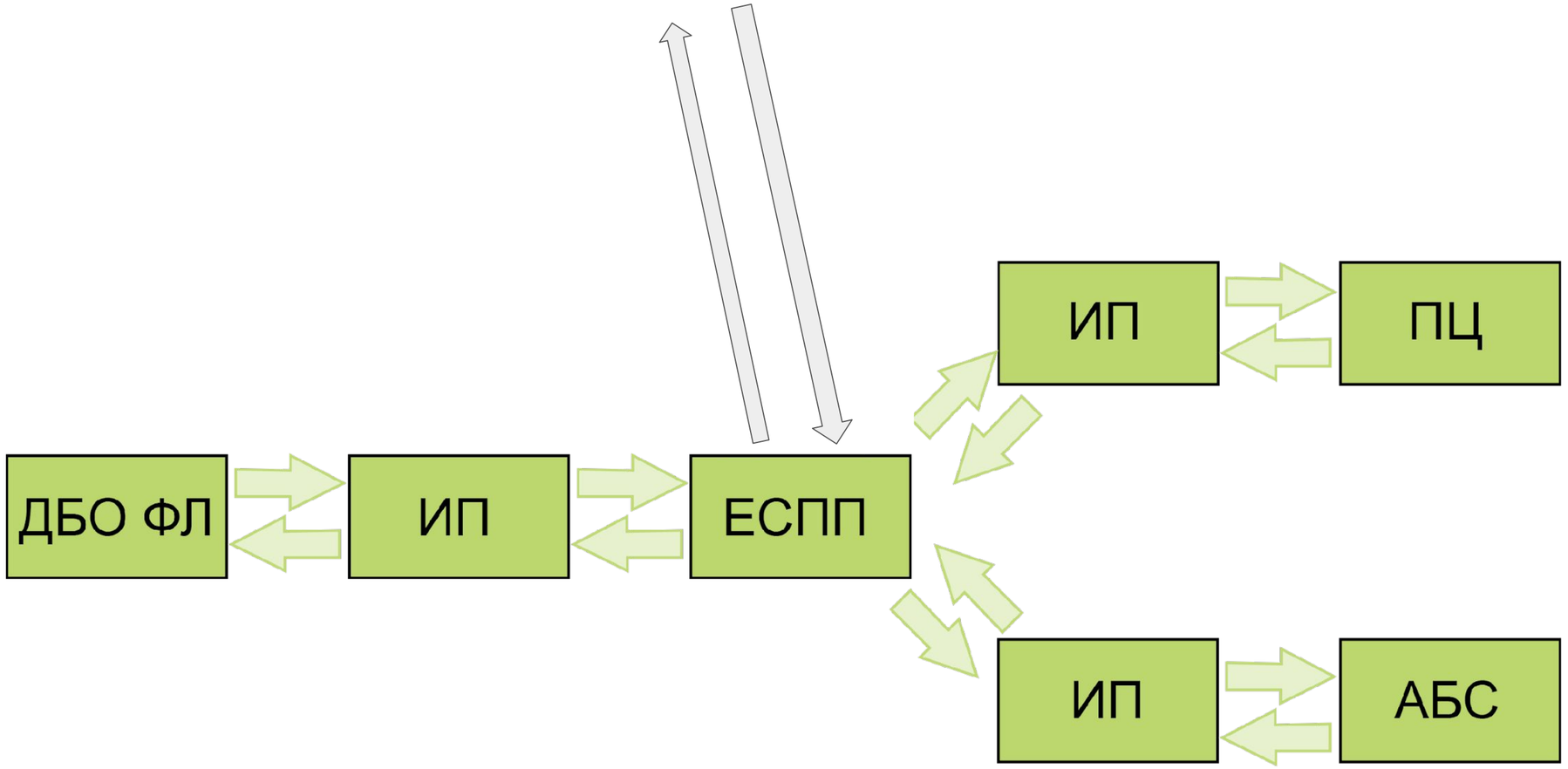


ЕСПП

Миддл-системы — интеграция фронта и бэка

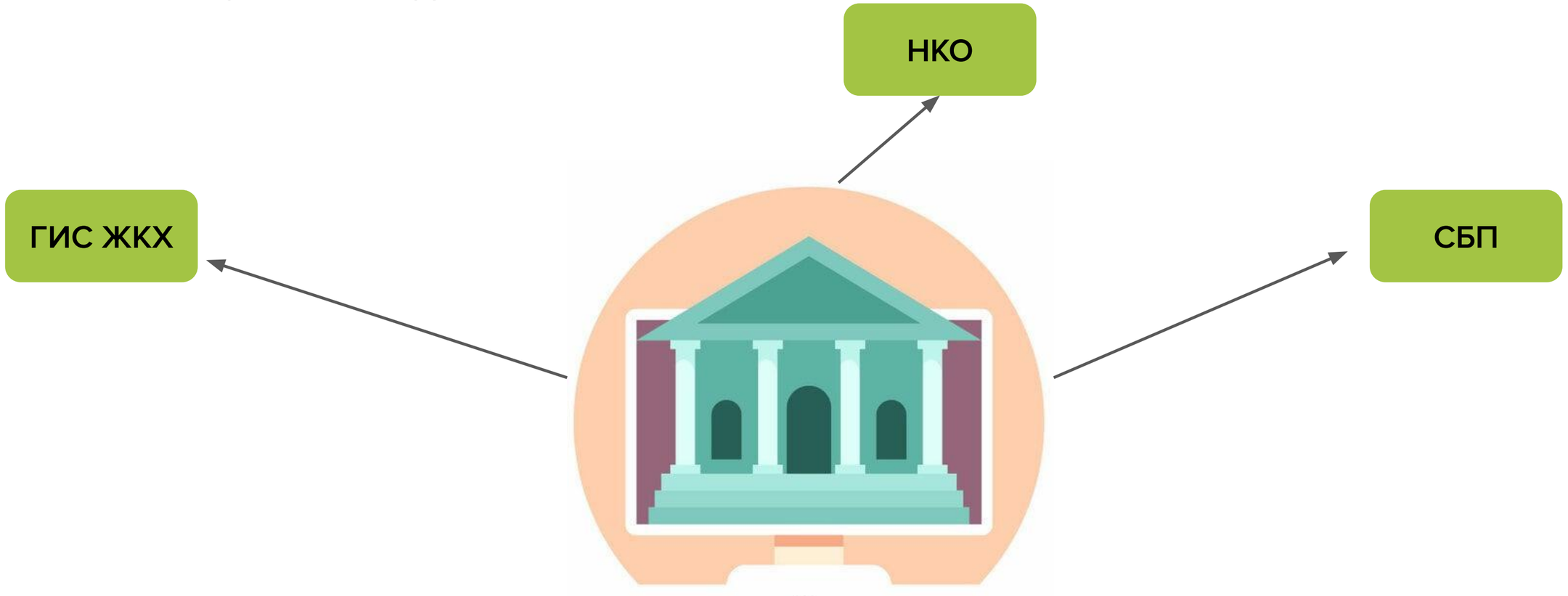


ЕСПП - Миддл система



Миддл системы

Параллельно мы можем обращаться за пределы нашего банка, если речь идет о платеже через какого-нибудь интегратора, будь то ГИС ЖКХ, НКО от сбербанка или СБП и этим системам тоже важно чтобы мы передавали корректные данные.



Регулятор

закон, правила, требования, сжатые сроки, четкие дедлайны

НКО

ГИС ЖКХ

СБП

Проблемы в работе старой системы

Система раньше

Я – маленькая, со мной норм

Проблемы в работе старой системы

Система раньше

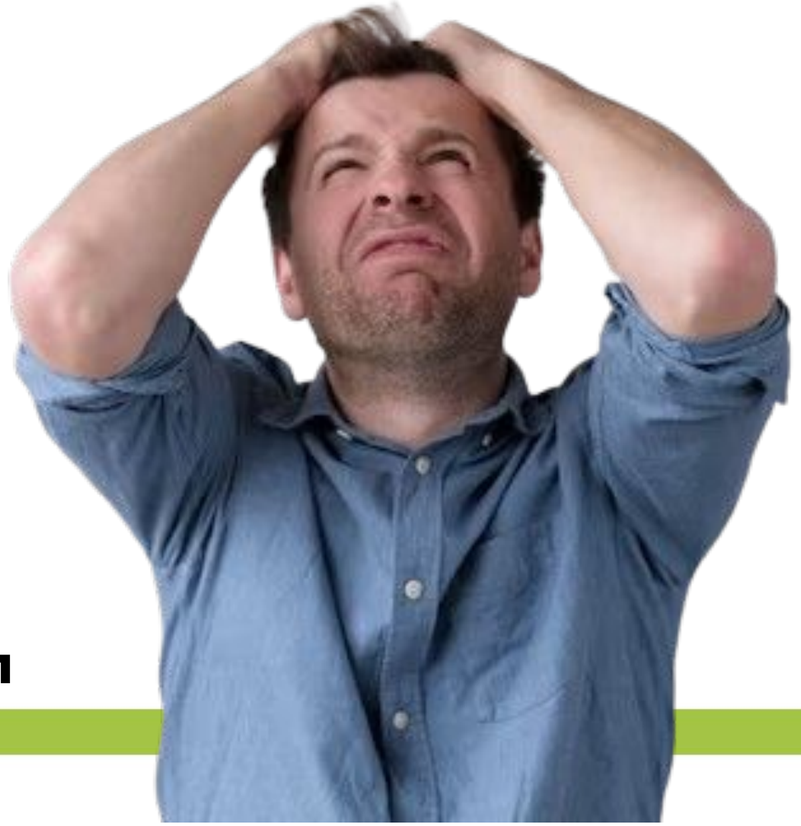
Я – маленькая, со мной норм

Система сейчас

Логи разрознены: ядро на сервере, контейнеры, взаимодействие в Kibana

Модули: ~137 кастомных, ~17 коробочных
Модули как собственной разработки, так и созданные вендором

АААААААААААААААААА



может всё-таки проп

РСХБ РСХБ цифра

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

- Использование декларативного подхода к конфигурации

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений
- Автоматическая настройка и проверка совместимости

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений
- Автоматическая настройка и проверка совместимости
- Минимизация ручных операций

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений
- Автоматическая настройка и проверка совместимости
- Минимизация ручных операций
- Повышение точности и надежности интеграций

Переход наApp.Farm— собственную разработку

Текущая ситуация

- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений
- Автоматическая настройка и проверка совместимости
- Минимизация ручных операций
- Повышение точности и надежности интеграций
- Мониторинг пути прохождения данных

Переход наApp.Farm— собственную разработку

Текущая ситуация

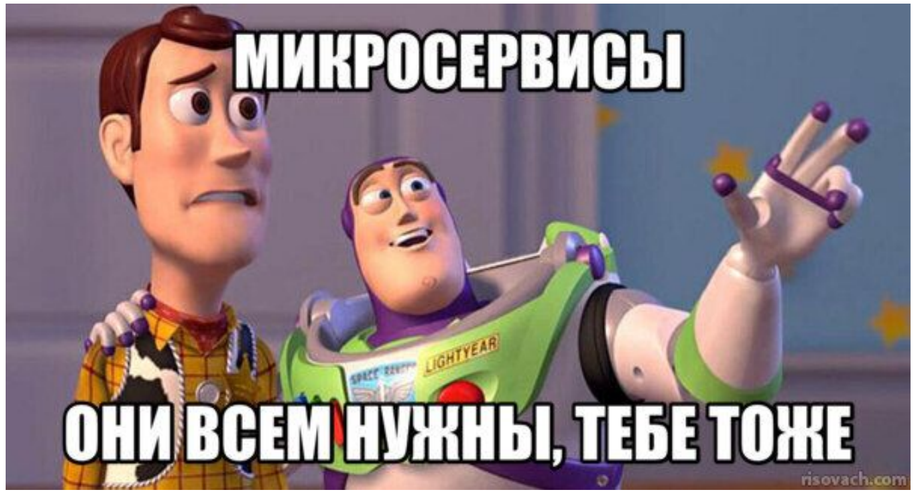
- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

В процессе поиска дефекта, приходится держать все это взаимодействие в голове

К чему стремимся

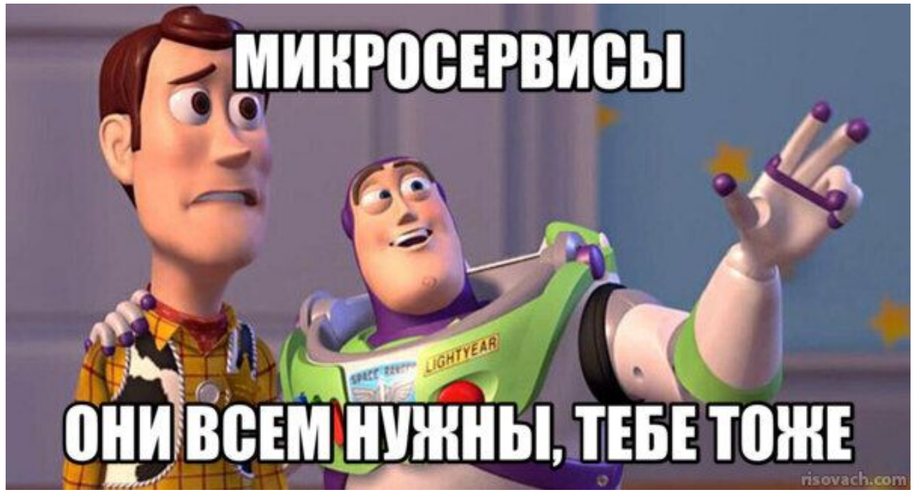
- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений
- Автоматическая настройка и проверка совместимости
- Минимизация ручных операций
- Повышение точности и надежности интеграций
- Мониторинг пути прохождения данных
- Анализ и оптимизация трафика

Переход наApp.Farm— собственную разработку



App.Farm был разработан с нуля для решения специфических задач банка:

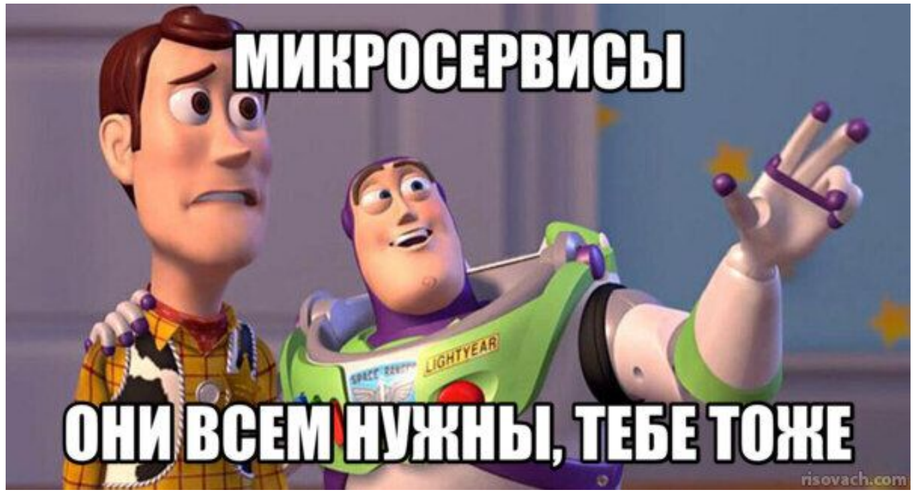
Переход наApp.Farm— собственную разработку



App.Farm был разработан с нуля для решения специфических задач банка:

- Централизованное логирование

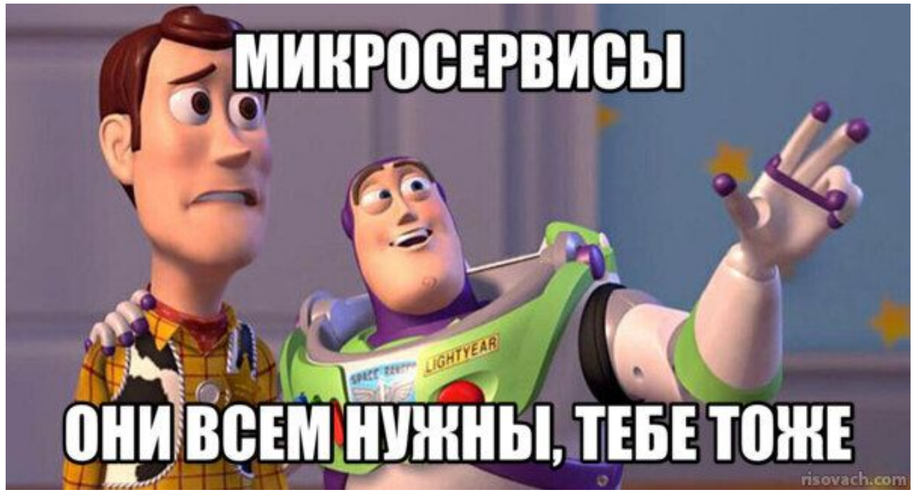
Переход наApp.Farm— собственную разработку



App.Farm был разработан с нуля для решения специфических задач банка:

- Централизованное логирование
- Трассировка запросов

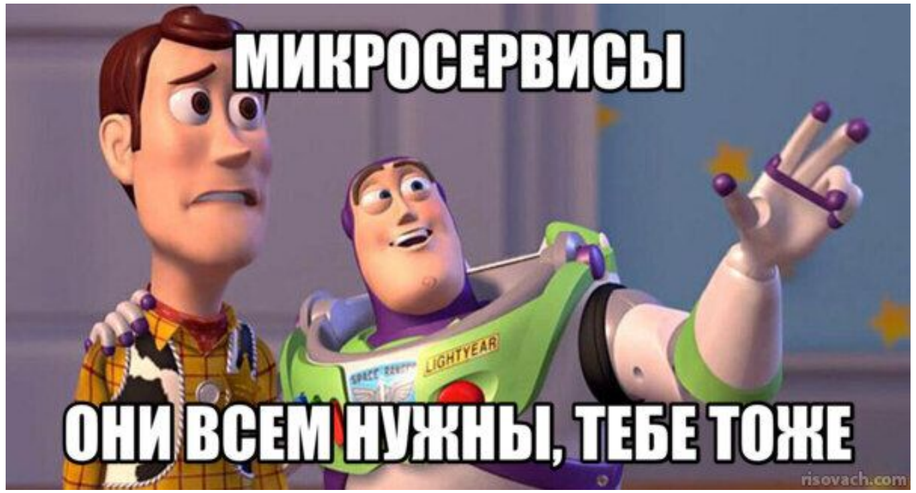
Переход наApp.Farm— собственную разработку



App.Farm был разработан с нуля для решения специфических задач банка:

- Централизованное логирование
- Трассировка запросов
- Автоматизация тестирования и деплоя

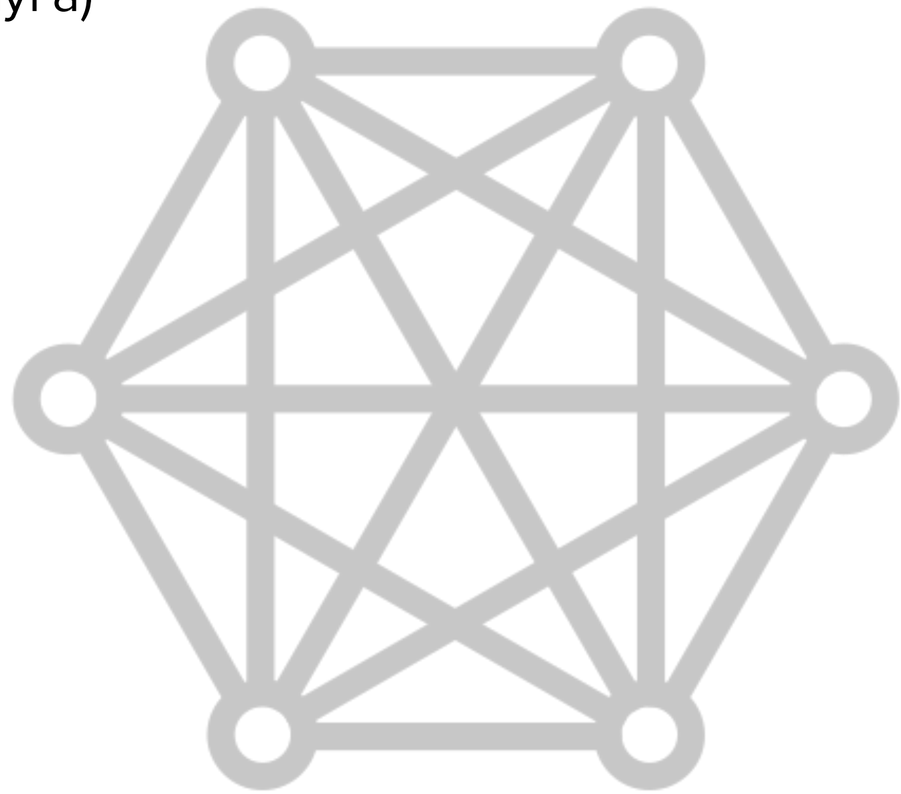
Переход наApp.Farm— собственную разработку



App.Farm был разработан с нуля для решения специфических задач банка:

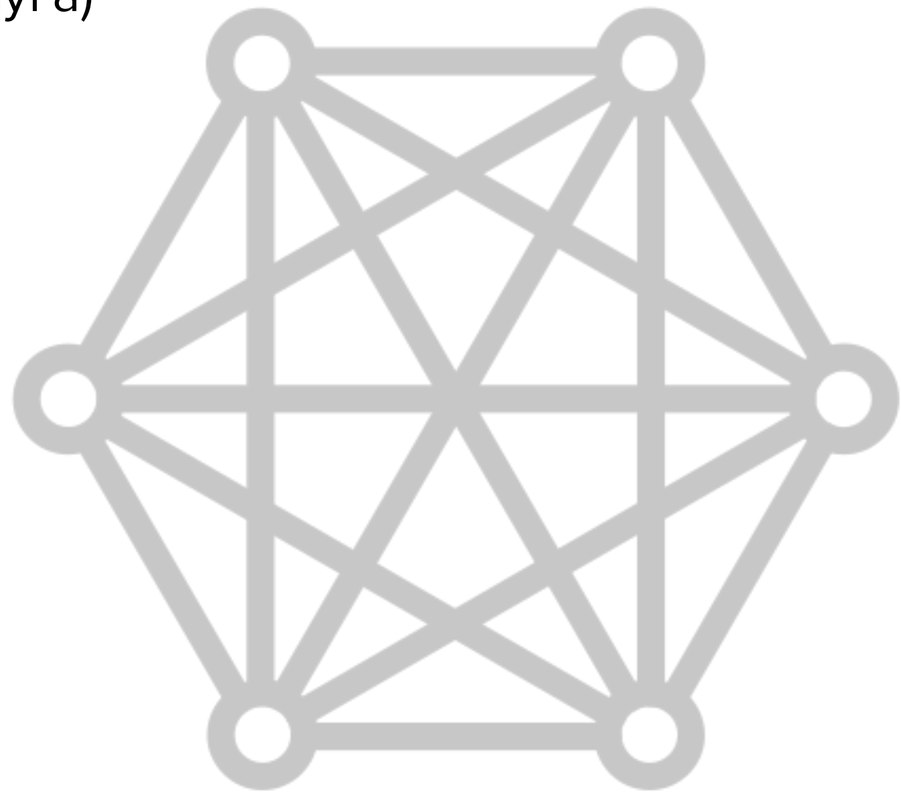
- Централизованное логирование
- Трассировка запросов
- Автоматизация тестирования и деплоя
- Использование Docker контейнеров

App.Farm – это PaaS (Platform as a Service / Платформа как услуга)



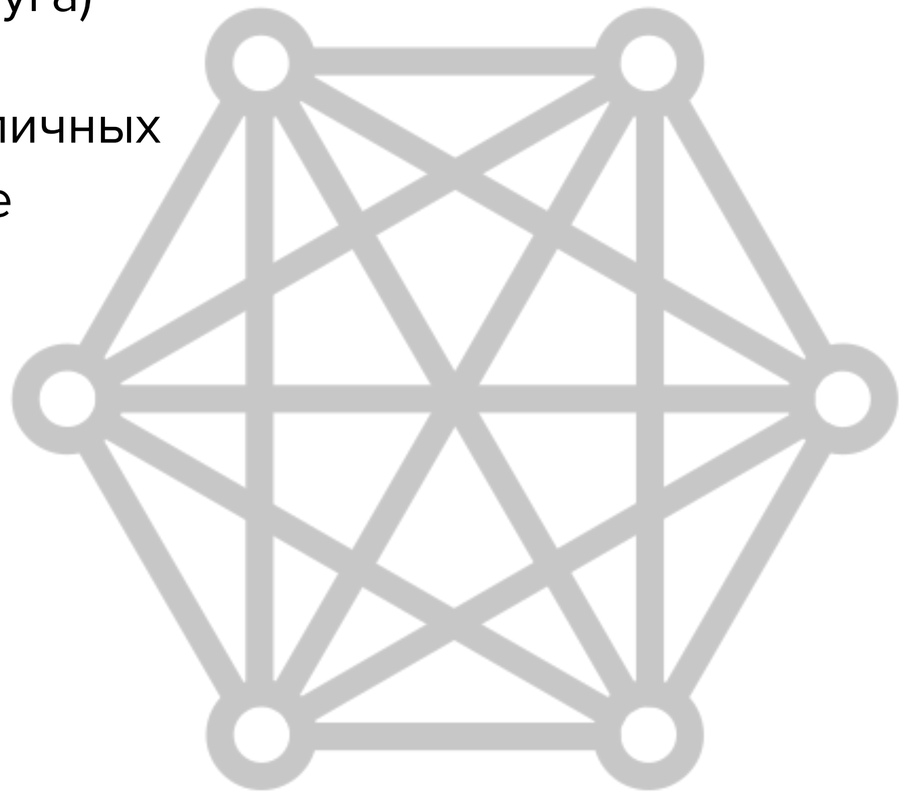
App.Farm – это PaaS (Platform as a Service / Платформа как услуга)

Опять коробка!!!



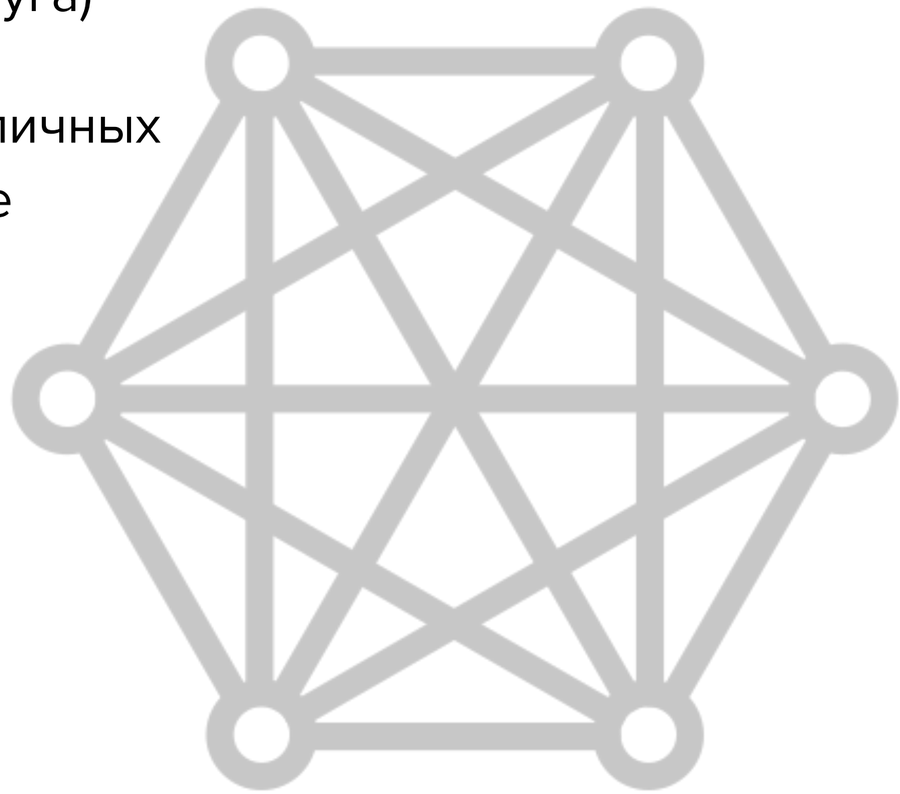
App.Farm – это PaaS (Platform as a Service / Платформа как услуга)

App.Farm – это единая неделимая система, состоящая из различных инфраструктурных компонентов, работающих вместе которые нужны для обеспечения работы самой платформы.



App.Farm – это PaaS (Platform as a Service / Платформа как услуга)

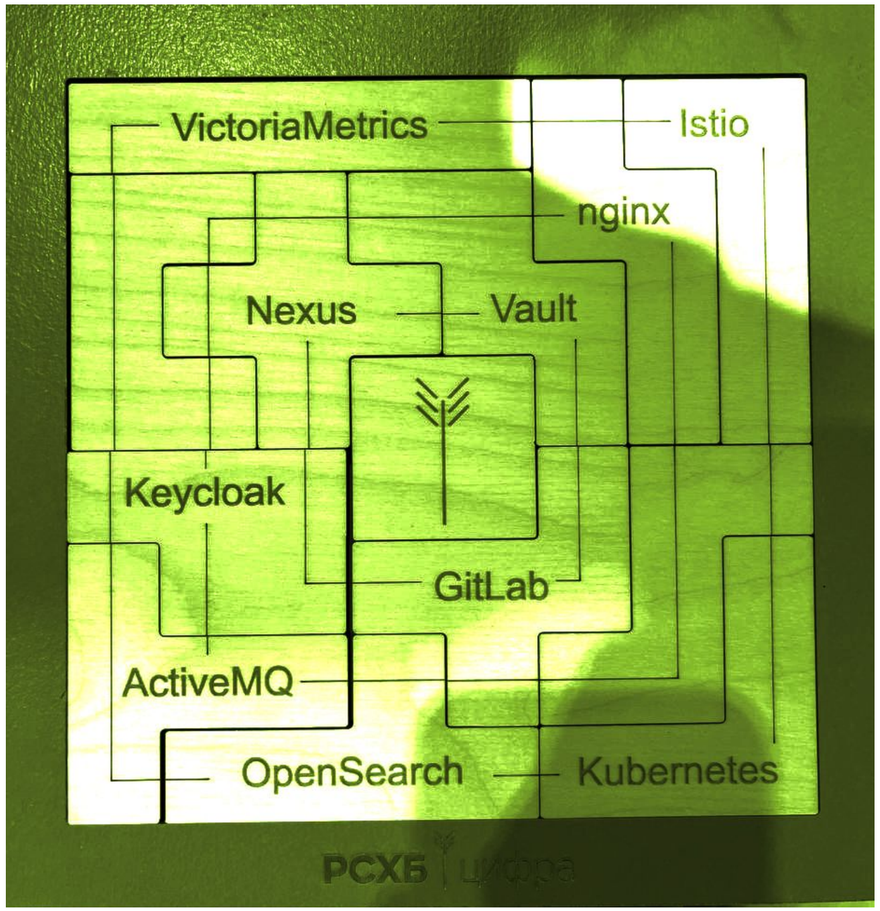
App.Farm – это единая неделимая система, состоящая из различных инфраструктурных компонентов, работающих вместе которые нужны для обеспечения работы самой платформы.



T.e. App.Farm не предоставляет Kubernetes, Nexus, Vault, Elasticsearch или другие подобные компоненты как сервис. Вместо этого есть API

App.Farm построена на следующем стеке открытых решений:

Kubernetes	Gitlab	OpenSearch
Jaeger	Grafana	Istio
VictoriaMetrics	и других открытых продуктах	



А также на инструментах собственной разработки:

- Россыпь платформенных операторов Kubernetes;
- Генератор докерфайлов;
- Адаптеры интеграций;
- Инструментарий деплоя на платформу

РСХБ РСХБ цифра

Инструменты App.Farm

Jaeger

Kibana

Grafana

Автоматизация и стандартизация

Использование комплекса из Jaeger, Kibana и Grafana помогает получить полное представление о производительности системы на разных уровнях:

- трассировка запросов

Автоматизация и стандартизация

Использование комплекса из Jaeger, Kibana и Grafana помогает получить полное представление о производительности системы на разных уровнях:

- трассировка запросов
- анализ логов

Автоматизация и стандартизация

Использование комплекса из Jaeger, Kibana и Grafana помогает получить полное представление о производительности системы на разных уровнях:

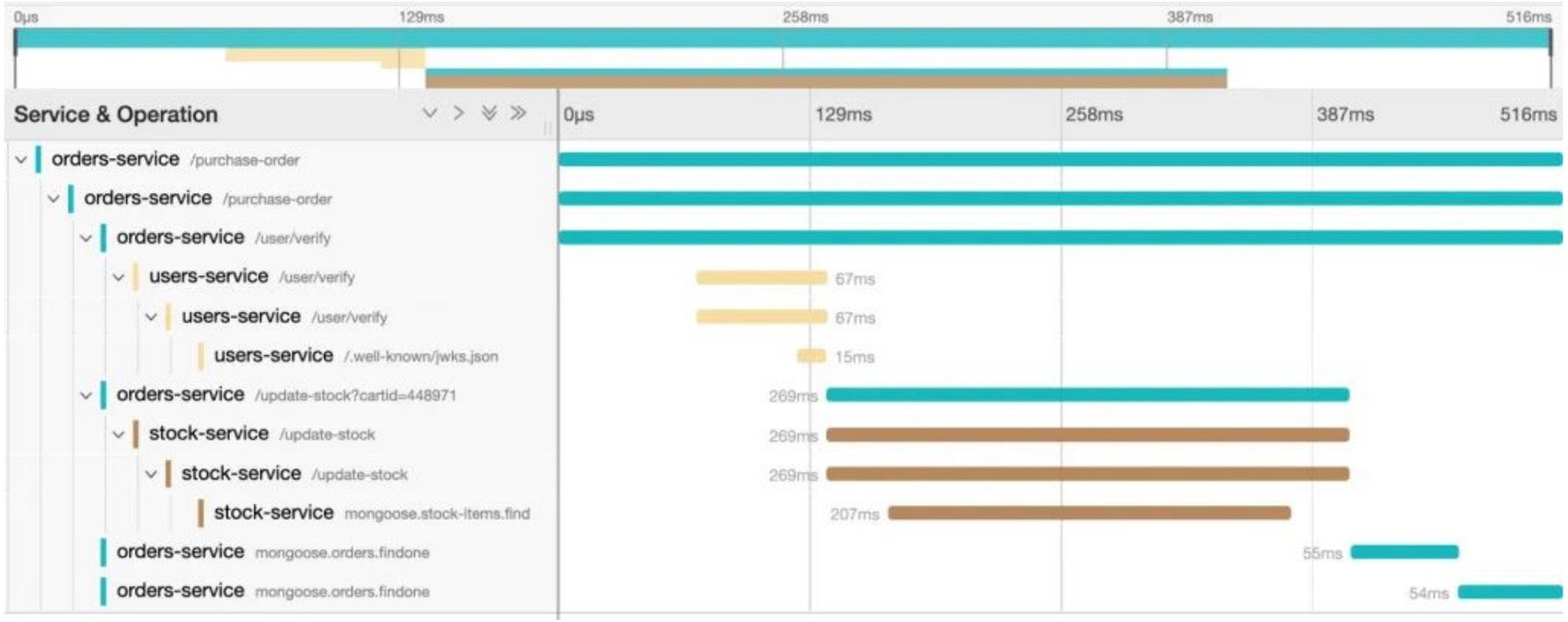
- трассировка запросов
- анализ логов
- мониторинг ресурсов

РСХБ РСХБ цифра

Jaeger

Цель: Профилирование и трассировка запросов

- Трассировка запросов



Jaeger

Цель: Профилирование и трассировка запросов

- Трассировка запросов: Jaeger позволяет визуализировать пути запросов через систему, отслеживая их время выполнения на каждом этапе.
- Анализ зависимостей

Jaeger

Цель: Профилирование и трассировка запросов

- Трассировка запросов: Jaeger позволяет визуализировать пути запросов через систему, отслеживая их время выполнения на каждом этапе.
- Анализ зависимостей: Можно прослеживать, какие сервисы взаимодействуют между собой, и, таким образом, определить, где возможны задержки или проблемы.
- Время выполнения:

Jaeger

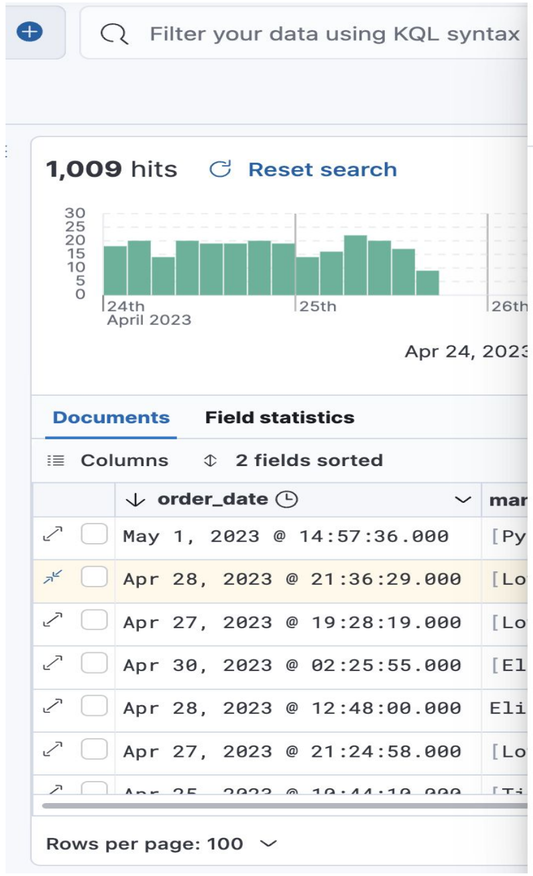
Цель: Профилирование и трассировка запросов

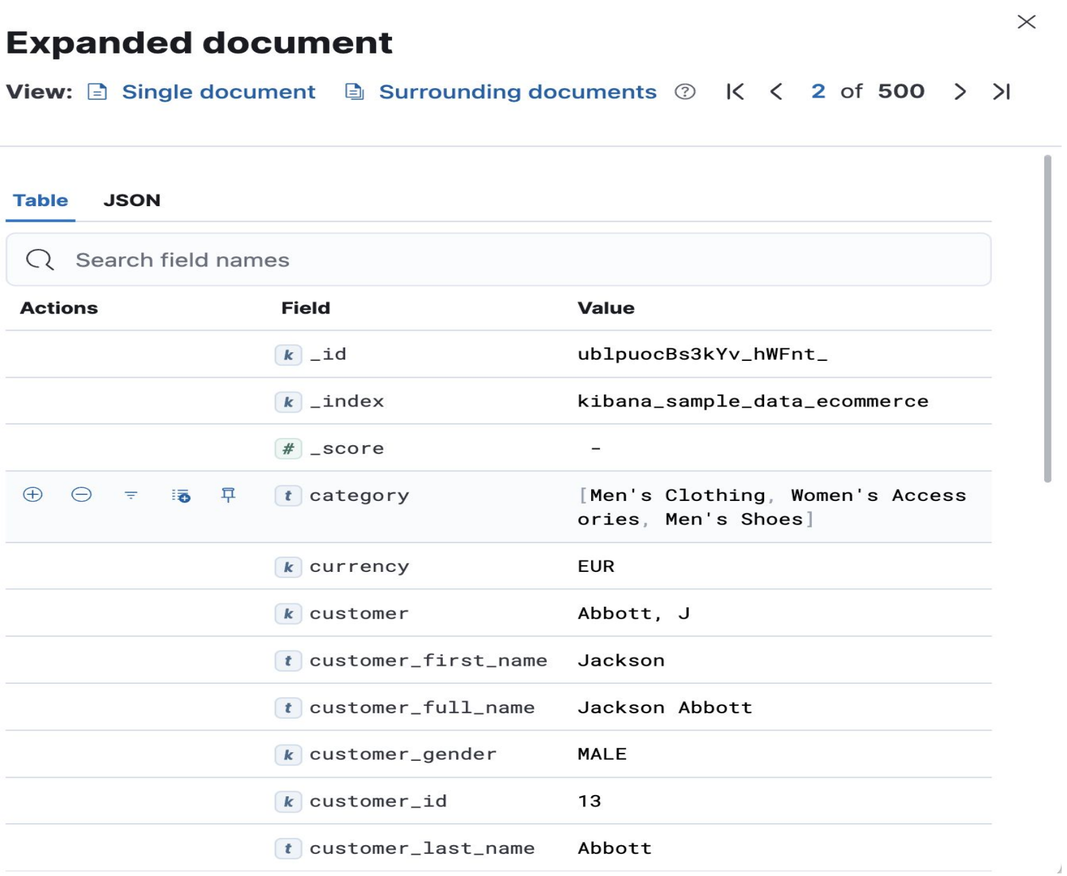
- Трассировка запросов: Jaeger позволяет визуализировать пути запросов через систему, отслеживая их время выполнения на каждом этапе.
- Анализ зависимостей: Можно прослеживать, какие сервисы взаимодействуют между собой, и, таким образом, определить, где возможны задержки или проблемы.
- Время выполнения: Jaeger предоставляет данные о времени, проведенном в каждом сервисе, что помогает выявить проблемы, связанные с медленными компонентами.
- Сравнение производительности:

Kibana

Цель: Анализ логов и мониторинг состояния приложения

- Сбор и анализ логов:



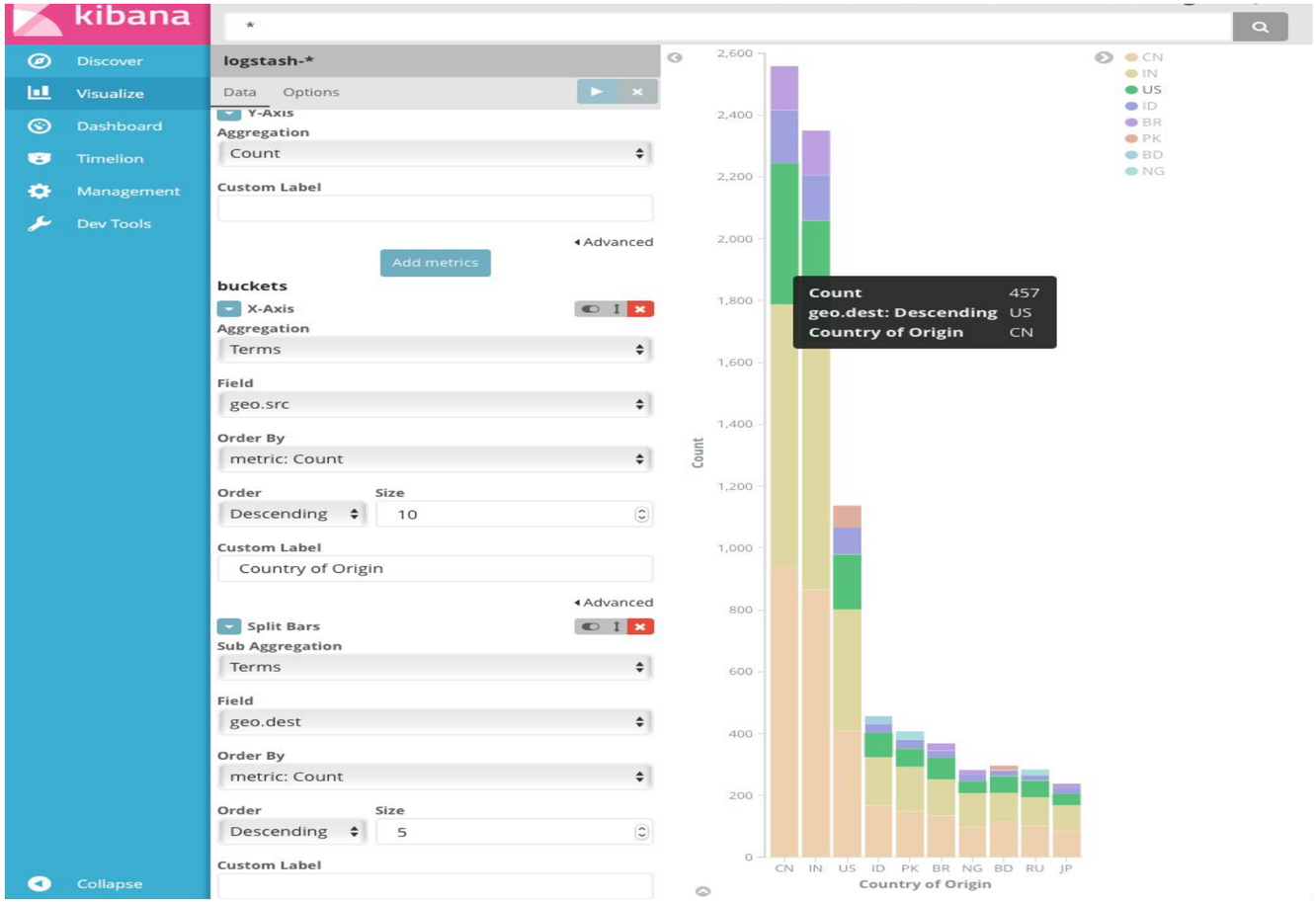


РСХБ РСХБ цифра

Kibana

Цель: Анализ логов и мониторинг состояния приложения

- Визуализация данных:



Kibana

Цель: Анализ логов и мониторинг состояния приложения

- Сбор и анализ логов: Kibana позволяет собирать и анализировать логи приложения намного быстрее, чем использовать простые текстовые файлы
- Визуализация данных: Создание дашбордов и графиков, чтобы визуально представлять данные из логов
- Поиск по логам:

Kibana

Цель: Анализ логов и мониторинг состояния приложения

- Сбор и анализ логов: Kibana позволяет собирать и анализировать логи приложения намного быстрее, чем использовать простые текстовые файлы
- Визуализация данных: Создание дашбордов и графиков, чтобы визуально представлять данные из логов
- Поиск по логам: Удобный поиск по логам позволяет находить конкретные сообщения об ошибках или события, что экономит время при отладке и тестировании
- Анализ отклонений:

Grafana

Цель: Мониторинг ресурсов и производительности

- Метрики производительности:

Grafana

Цель: Мониторинг ресурсов и производительности

- Метрики производительности: Grafana позволяет собирать и визуализировать метрики производительности в реальном времени, такие как загрузка CPU, использование памяти, задержки запросов и т.д.
- Анализ нагрузки:

Grafana

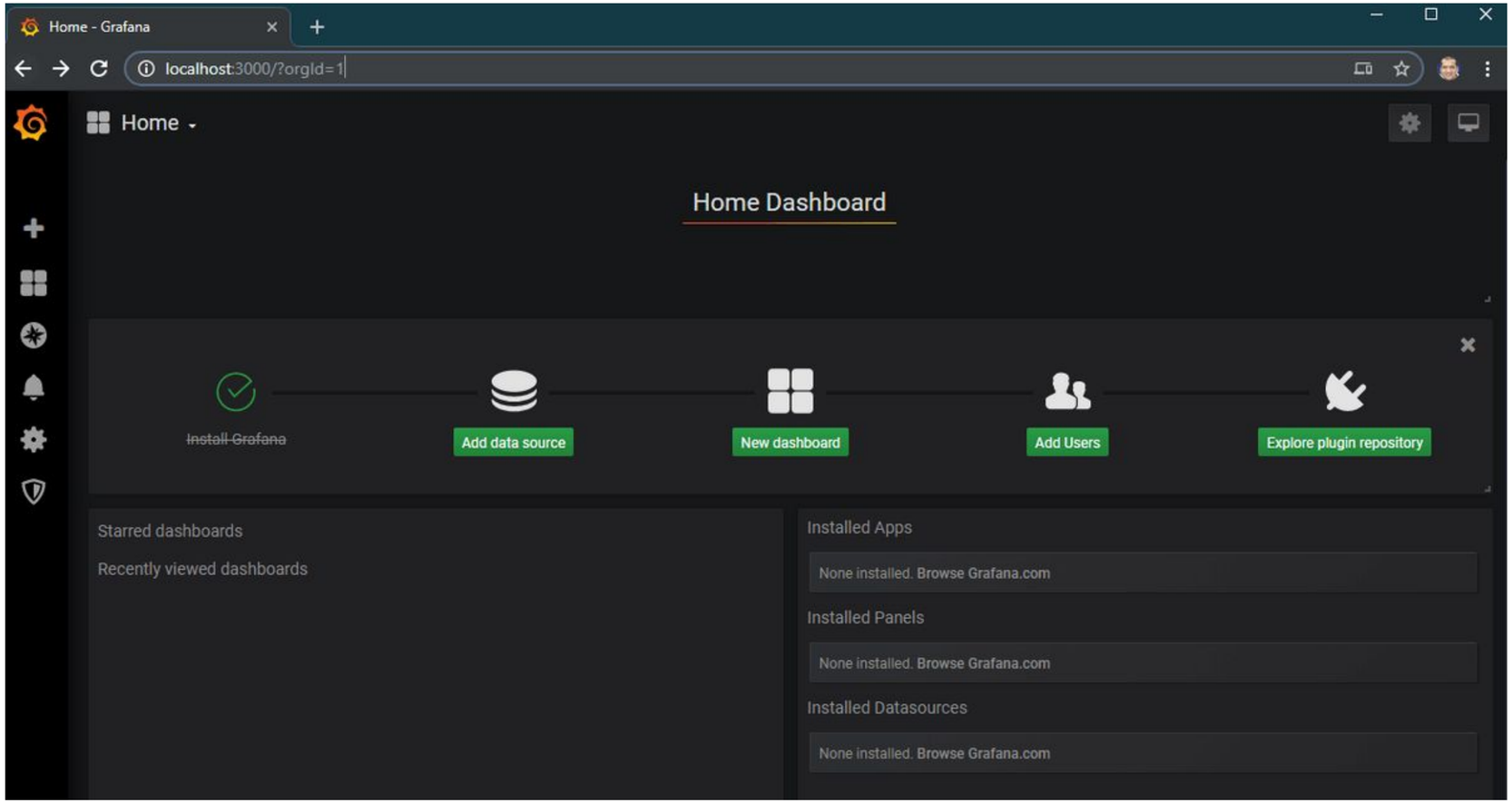
Цель: Мониторинг ресурсов и производительности

- Метрики производительности: Grafana позволяет собирать и визуализировать метрики производительности в реальном времени, такие как загрузка CPU, использование памяти, задержки запросов и т.д.
- Анализ нагрузки: С Grafana можно отслеживать, как распределяется нагрузка между разными компонентами системы
- Создание дашбордов:

Grafana

Цель: Мониторинг ресурсов и производительности

- Создание дашбордов:

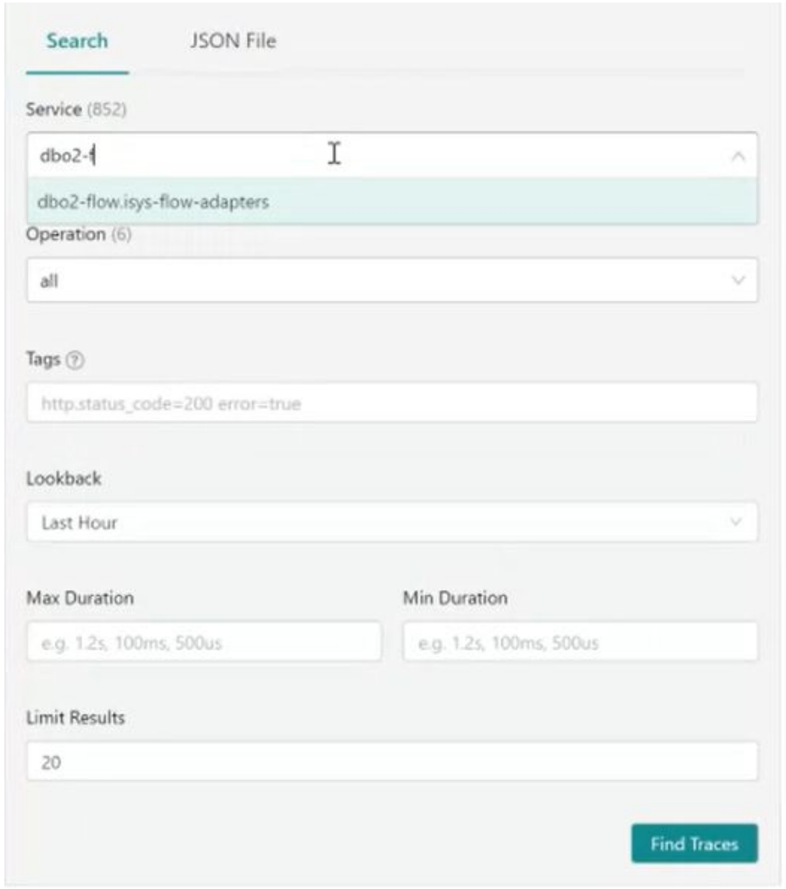


Grafana

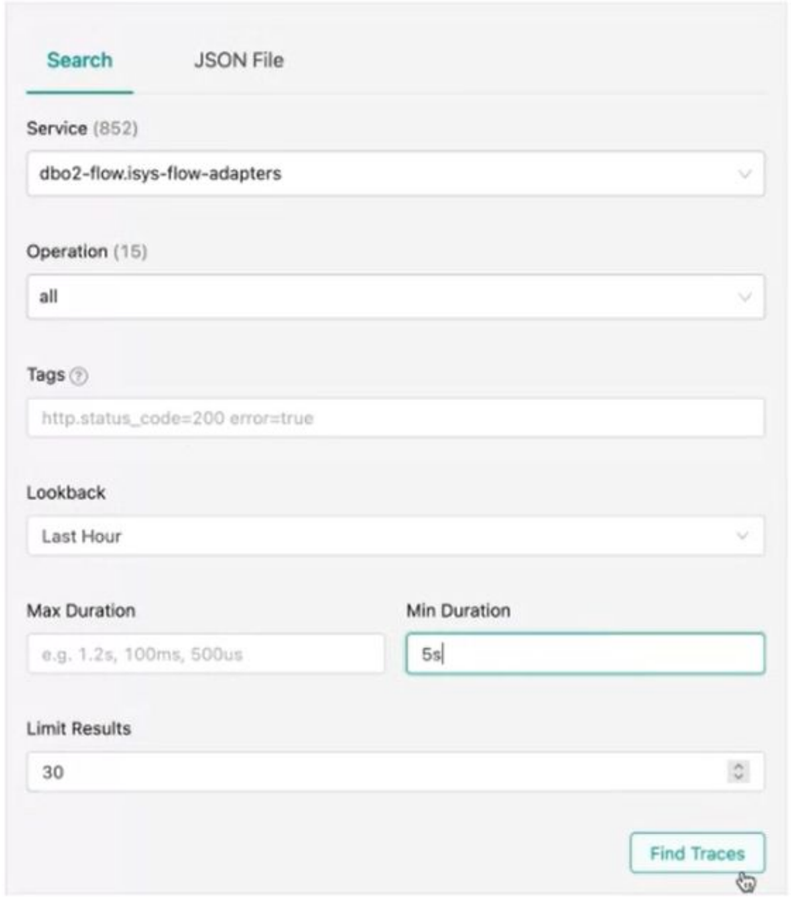
Цель: Мониторинг ресурсов и производительности

- Метрики производительности: Grafana позволяет собирать и визуализировать метрики производительности в реальном времени, такие как загрузка CPU, использование памяти, задержки запросов и т.д.
- Анализ нагрузки: С Grafana можно отслеживать, как распределяется нагрузка между разными компонентами системы
- Создание дашбордов: Можно создать дашборды, комбинируя данные из разных источников (включая Jaeger и Kibana), чтобы получить целостное представление о производительности и состоянии системы во время тестирования
- Предупреждения:

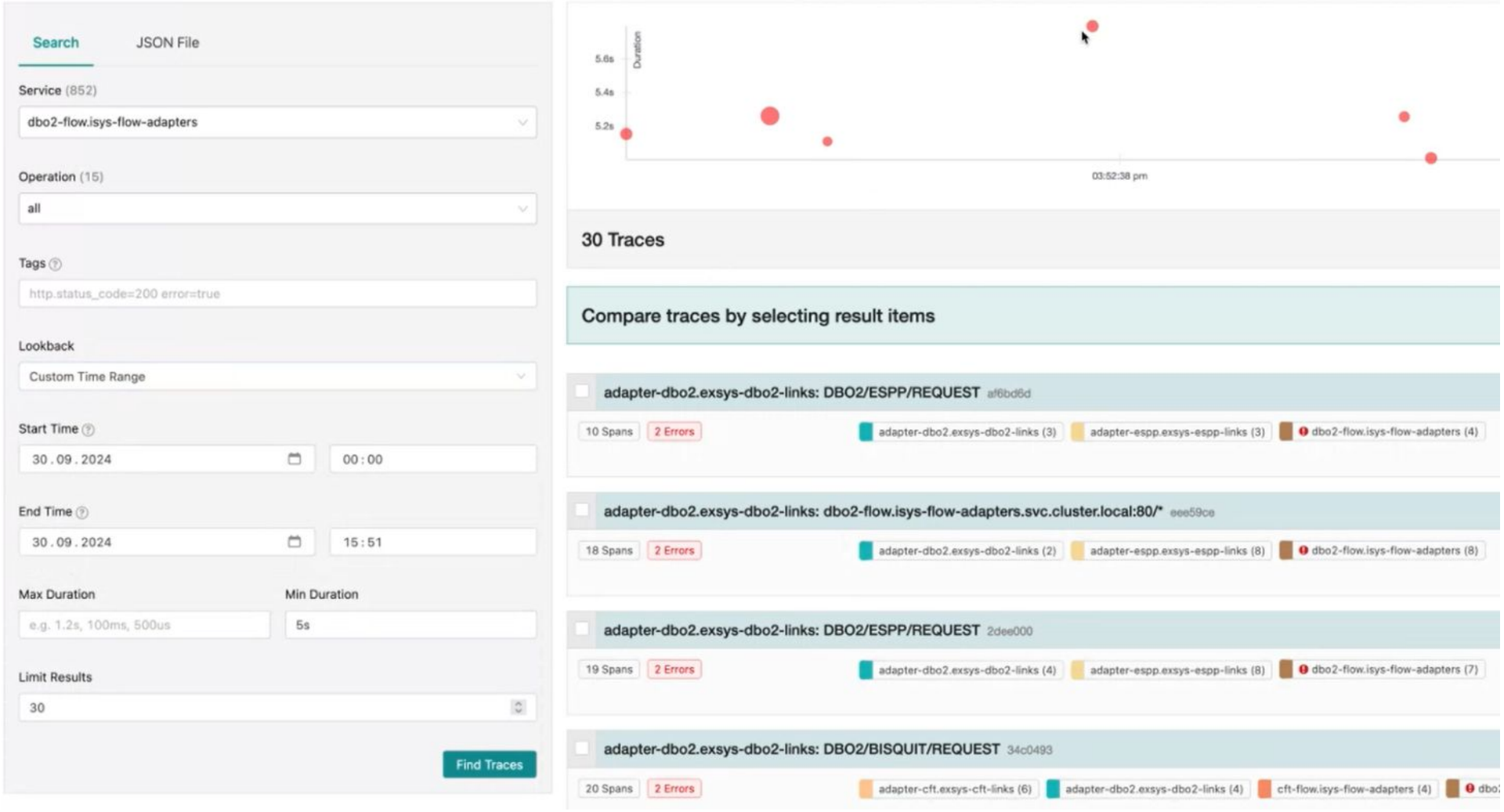
Пример в Jaeger:



Пример в Jaeger:



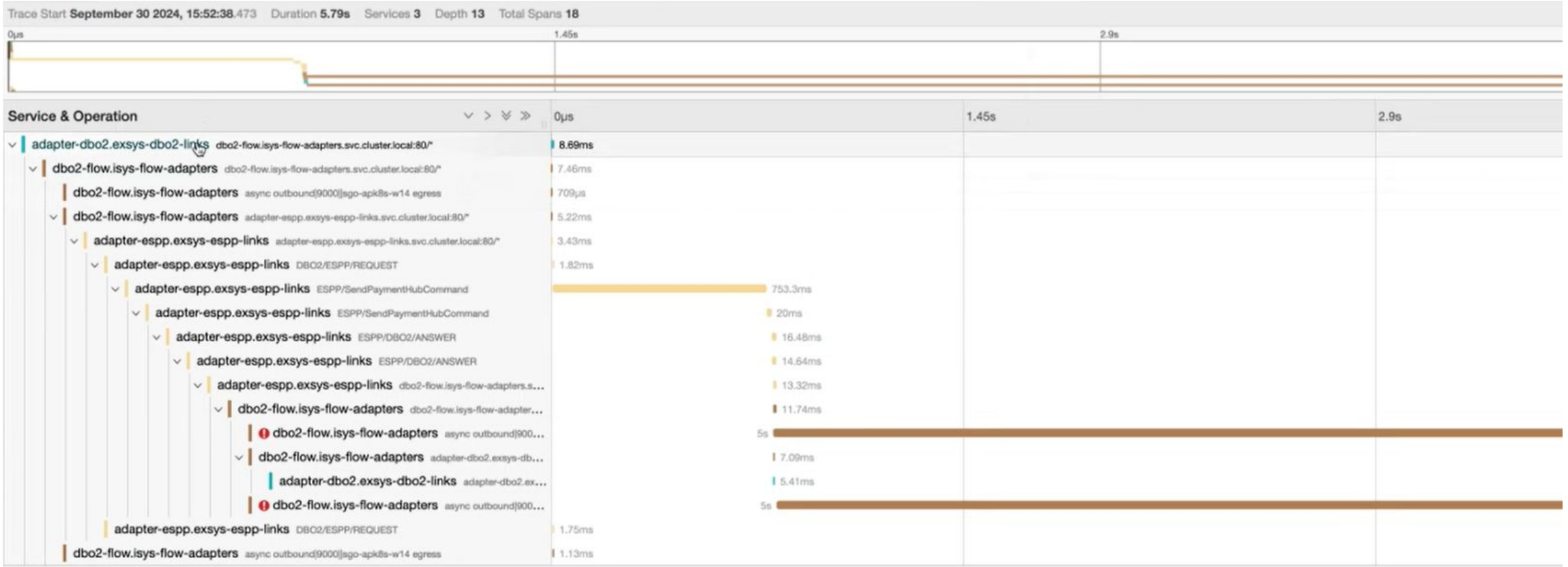
Пример в Jaeger:



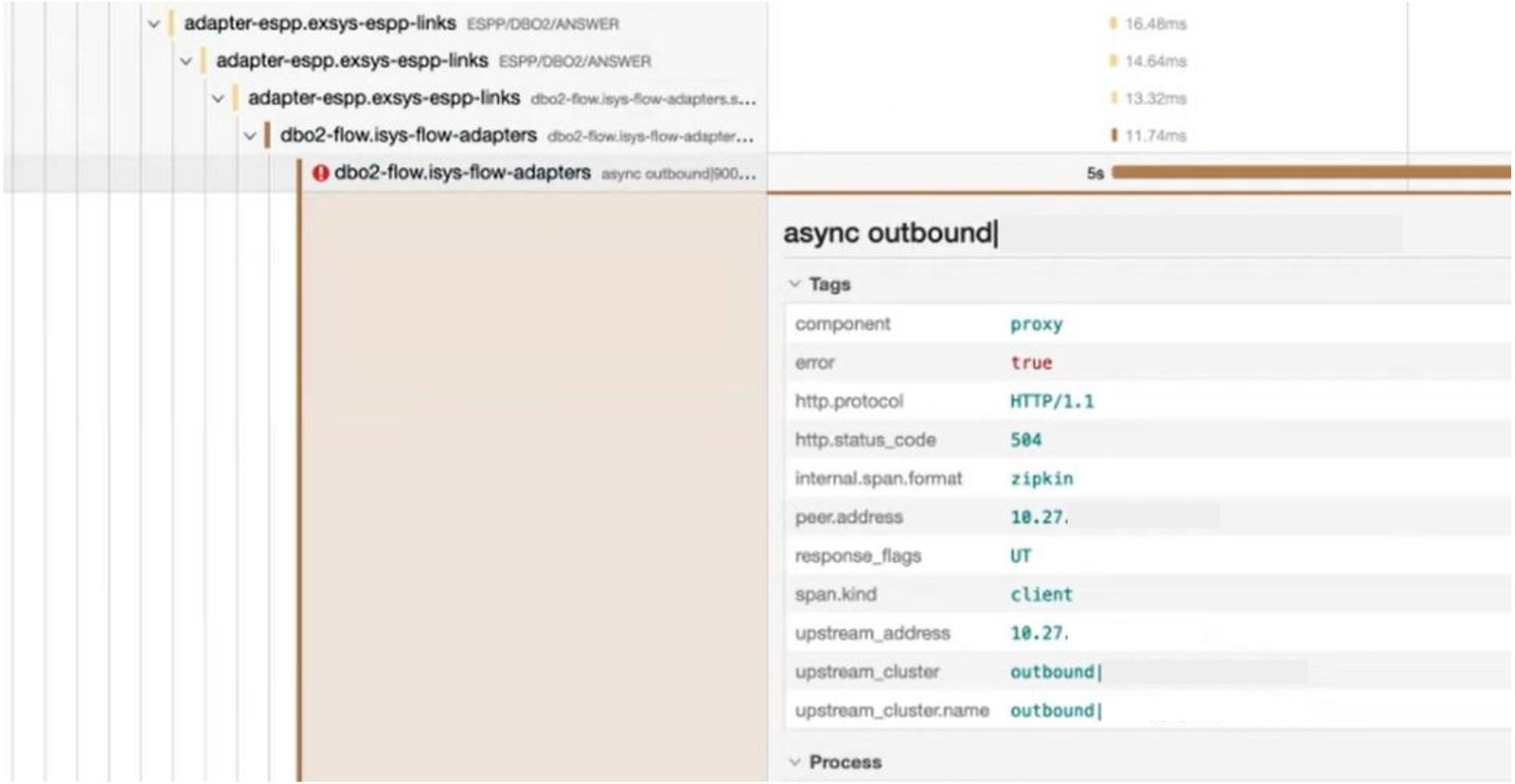
РСХБ

РСХБ цифра

Пример в Jaeger:



Пример в Jaeger:



В процессе миграции в AppFarm

- Мигрируем в App.Farm наши собственные модули (написаны на C#)

В процессе миграции в AppFarm

- Мигрируем в App.Farm наши собственные модули (написаны на C#)
- Встраиваем в них нужные метрики

РСХБ РСХБ цифра

В процессе миграции в AppFarm

- Мигрируем в App.Farm наши собственные модули (написаны на C#)
- Встраиваем в них нужные метрики
- Создаем дашборд, где будет видно и типовые и кастомные значения

РСХБ РСХБ цифра

В процессе миграции в AppFarm

- Мигрируем в App.Farm наши собственные модули (написаны на C#)
- Встраиваем в них нужные метрики
- Создаем дашборд, где будет видно типовые значения
- Метрики смотрим в Grafana

В процессе миграции в AppFarm

- Мигрируем в App.Farm наши собственные модули (написаны на C#)
- Встраиваем в них нужные метрики
- Создаем дашборд, где будет видно типовые значения
- Метрики смотрим в Grafana
- Логирование тоже можно настроить!

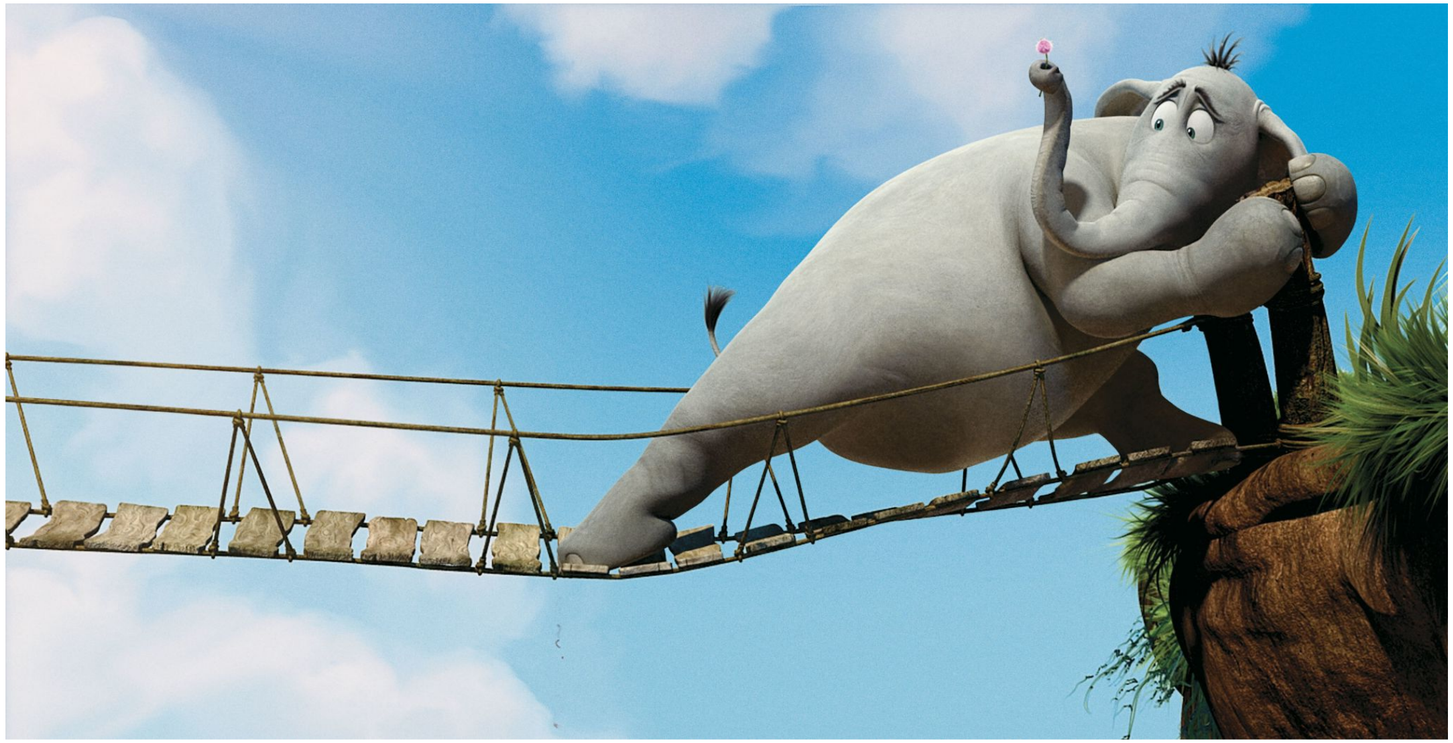
РСХБ РСХБ цифра

Дорогу осилит идущий!

Не бойтесь вступать на дорогу импортозамещения.

Это может выглядеть сложно, будет неудобно в процессе, но это - рост

И неизбежное повышение эффективности в будущем



РСХБ РСХБ цифра

Дорогу осилит идущий!

«Обрастать костылями» — не наш путь

Наш путь – искать максимально удобные решения, переходить к собственным решениям, чтобы легко управлять процессом

РСХБ РСХБ цифра

Дорогу осилит идущий!

Всем спасибо, с вами был Ахметов Андрей

@UFA_Akhmetov_Andrey

подробнее про App.Farm

Спасибо за внимание!

РСХБ РСХБ цифра