

# ТЕСТИРОВАНИЕ В ФИНТЕХЕ:

## НА ГРЕБНЕ ВОЛНЫ ИМПОРТОЗАМЕЩЕНИЯ



**Андрей  
Ахметов**

 **РСХБ**

**РСХБ** | цифра

# Что такое финтех?

# 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

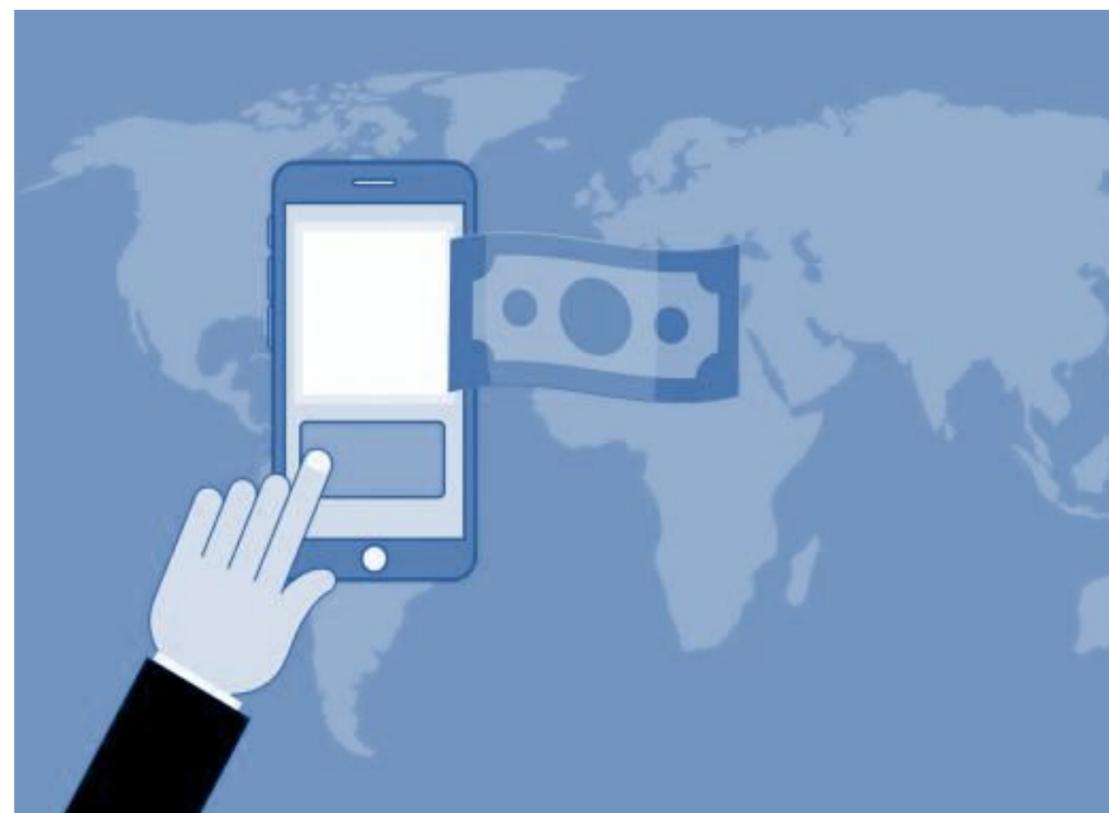


# Как вы видите банк

Банк раньше



Банк сейчас



# 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



# Финтех-системы и команды в банке

ДБО ФЛ



АБС

# Финтех-системы и команды в банке

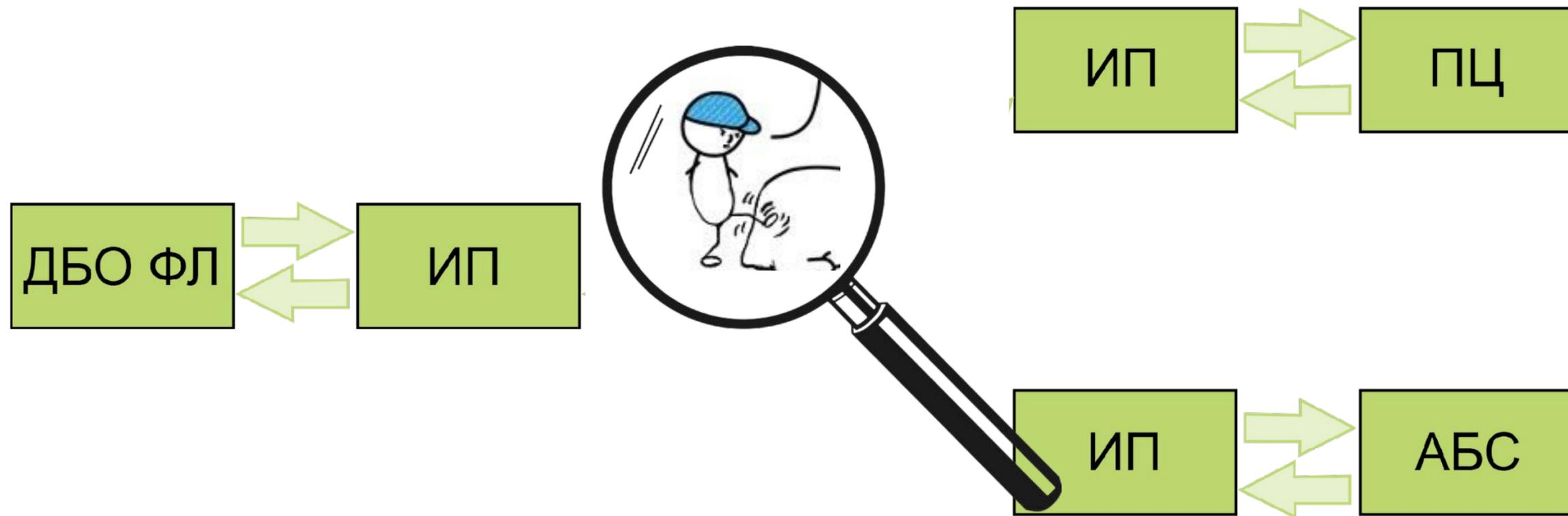
ДБО ФЛ



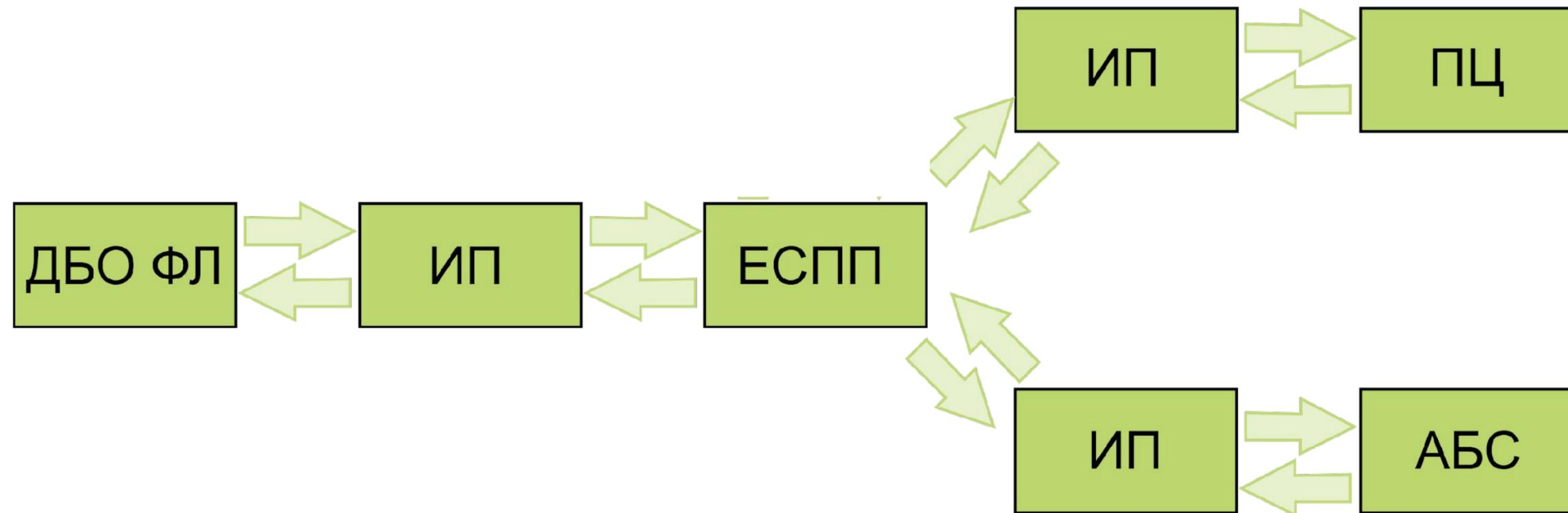
ПЦ

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.

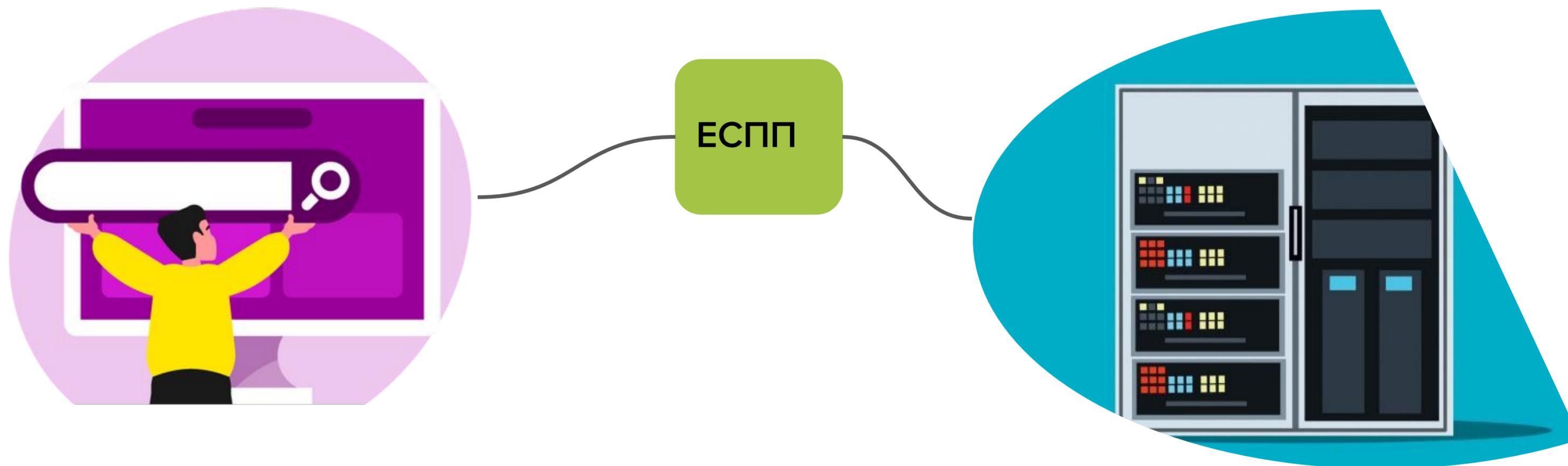


# ДБО

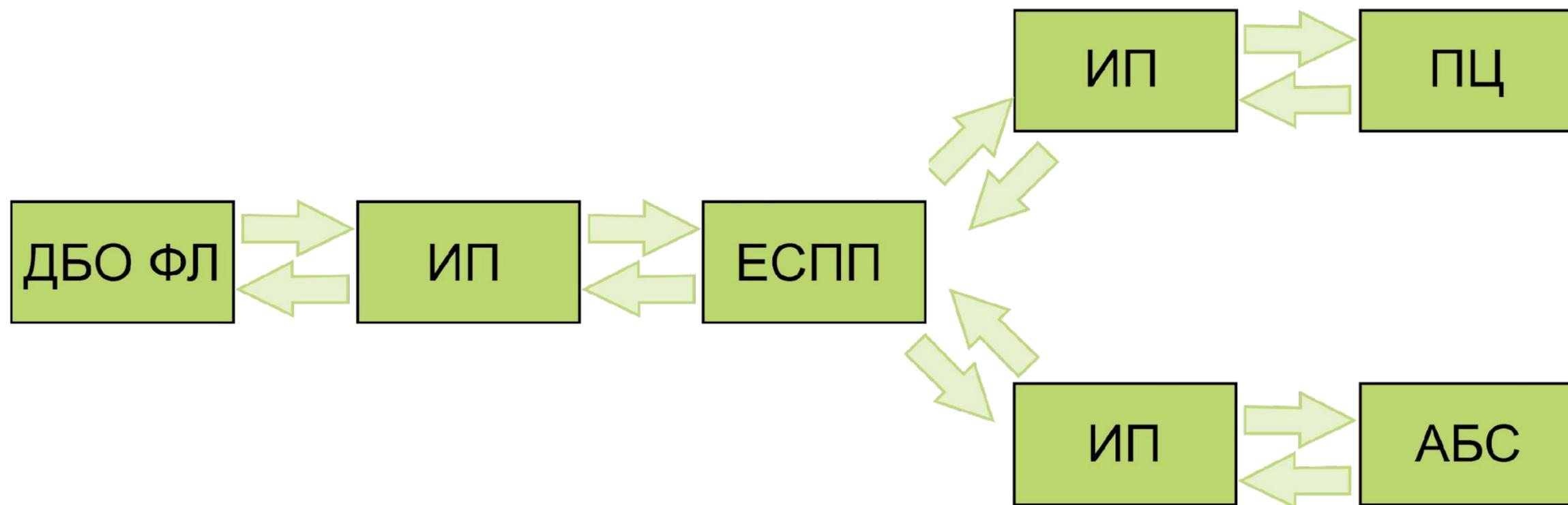
в процессе тестирования важно убедиться, что нужный элемент интерфейса появляется в нужное время в процессе оплаты, их больше волнует дизайн и верстка.

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

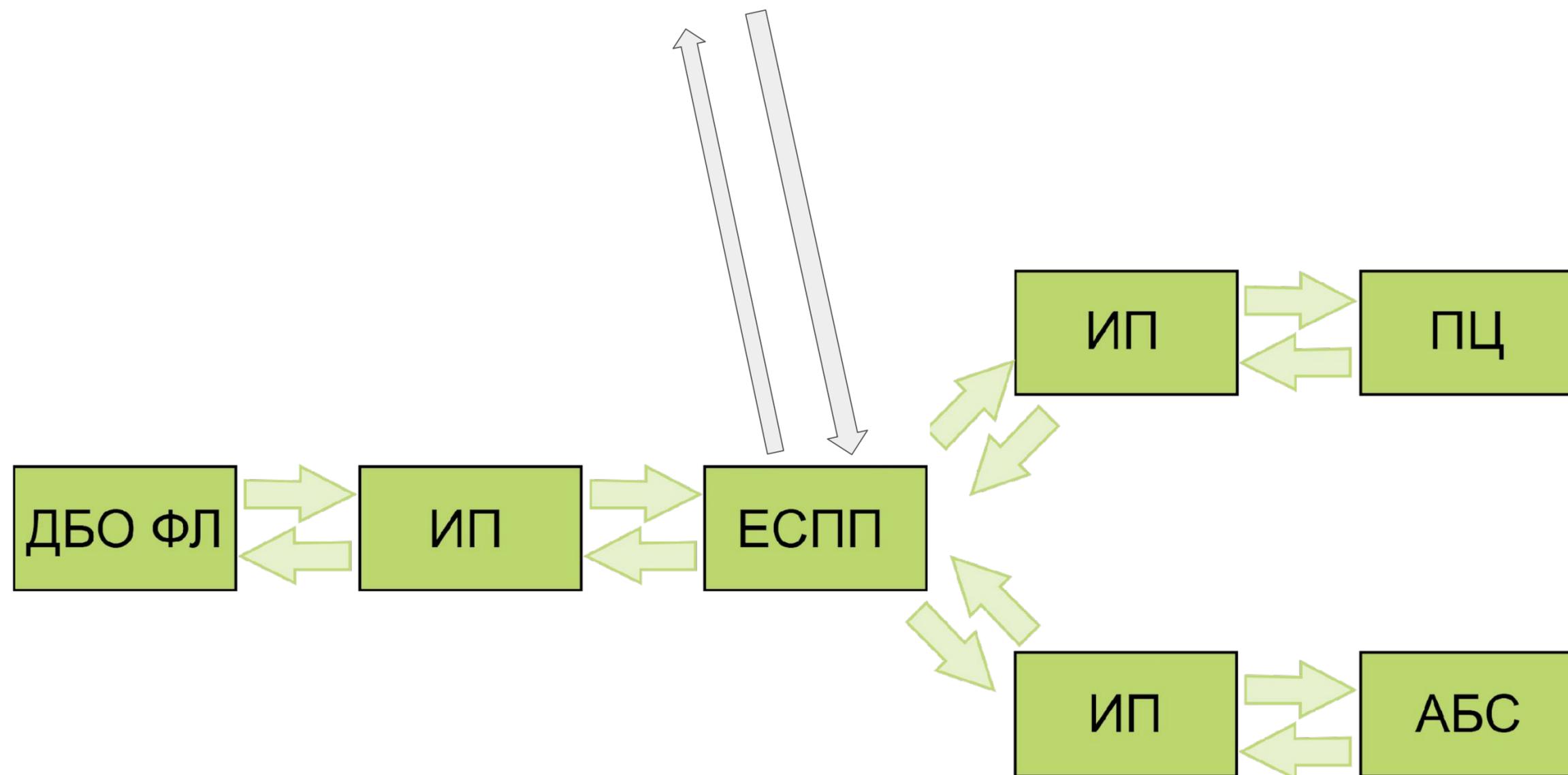
# АБС



# Миддл-системы — интеграция фронта и бэка

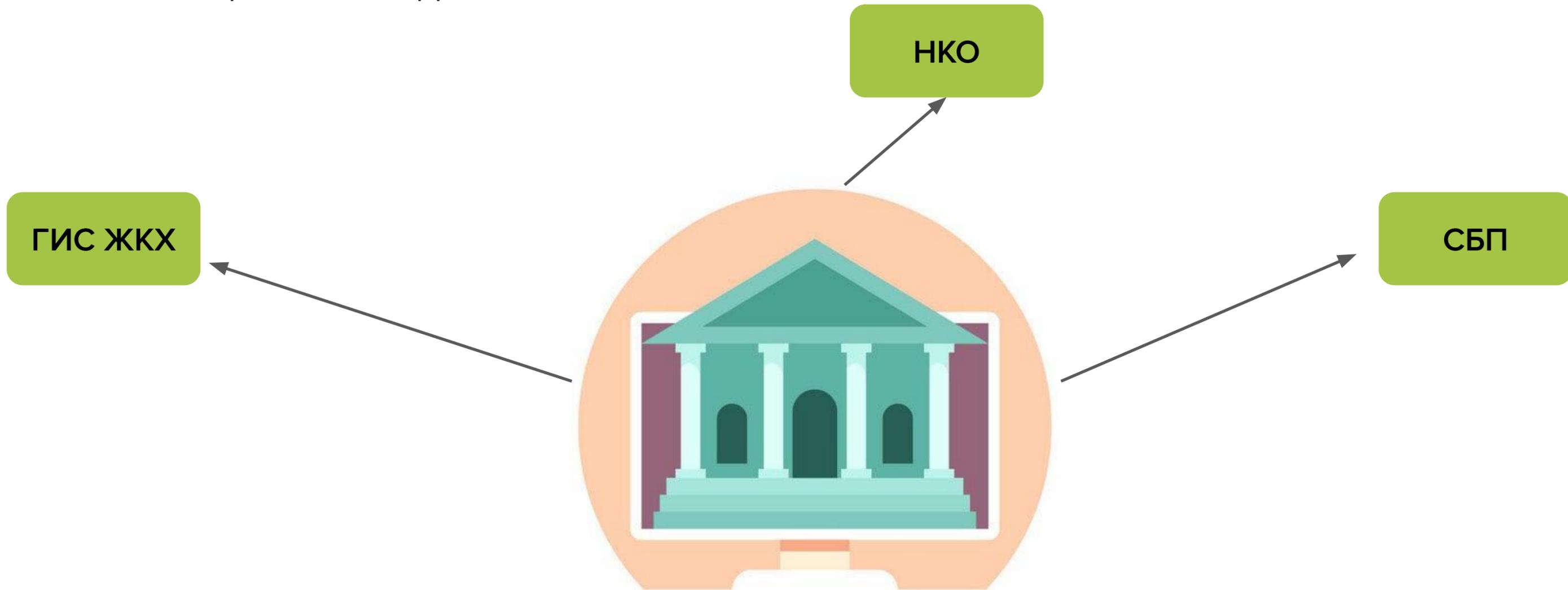


# ЕСПП - Миддл система



# 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



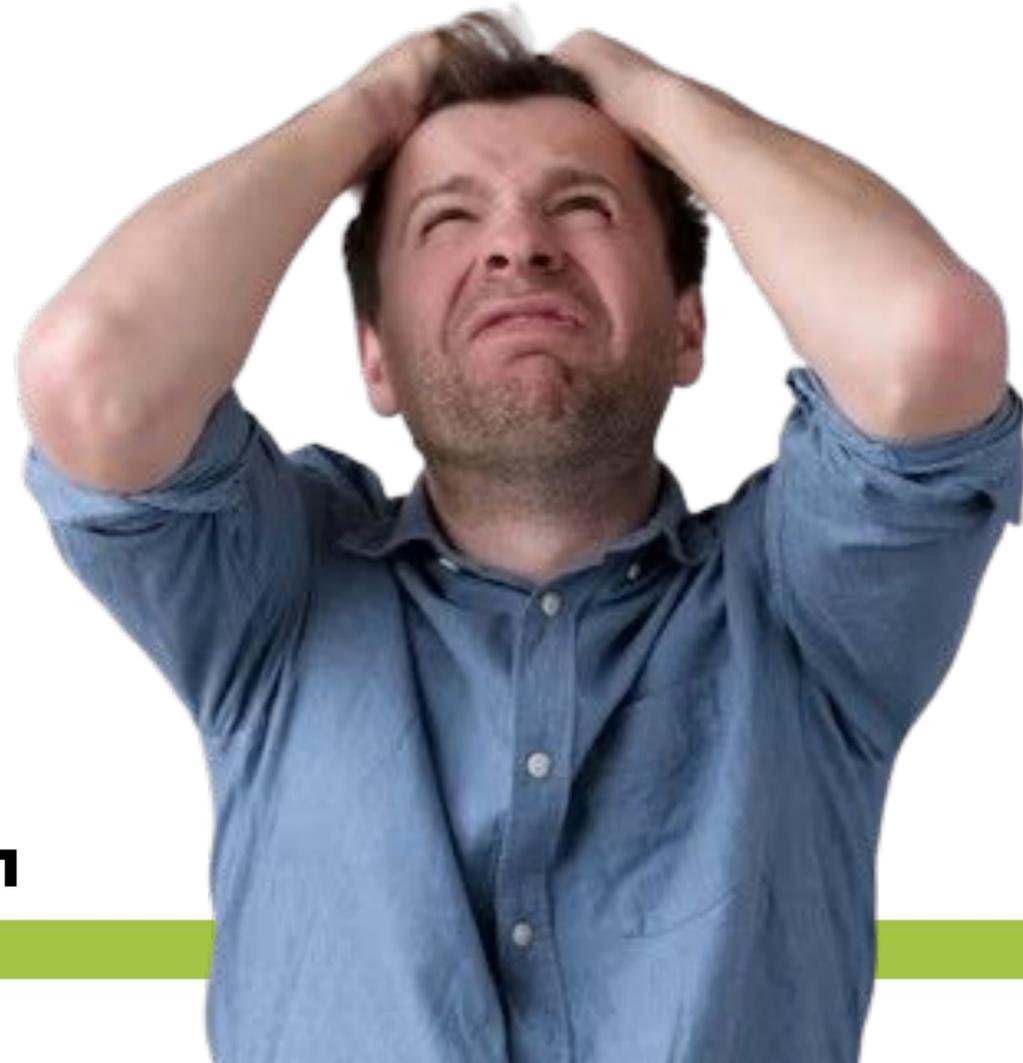
Система сейчас

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

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



**AAAAAAAAAAAAAAAAAAAA**



**может всё-таки проп**



# Переход на App.Farm— собственную разработку

## Текущая ситуация

- Коробочное решение

# Переход на App.Farm— собственную разработку

## Текущая ситуация

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

# Переход на App.Farm— собственную разработку

## Текущая ситуация

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

# Переход на App.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 на App.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 на App.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 на App.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 на App.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 на App.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 на App.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 на App.Farm— собственную разработку

## Текущая ситуация

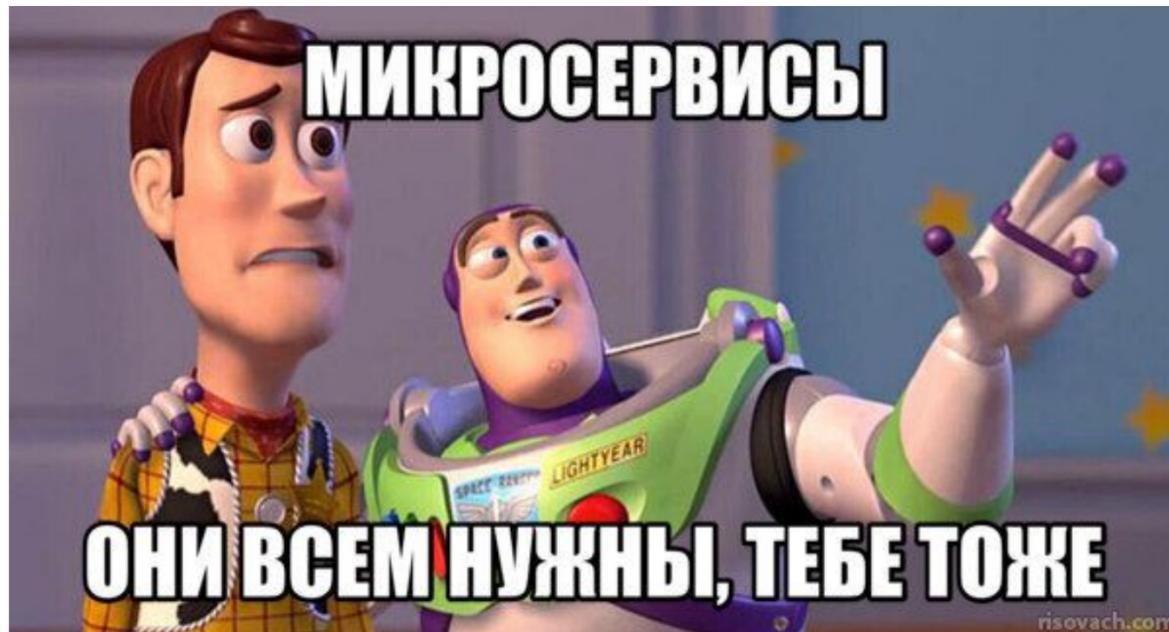
- Коробочное решение
- Кастомные модули, которые мы не можем попросить доработать вендора
- Логи и взаимодействия раскиданы по разным местам

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

## К чему стремимся

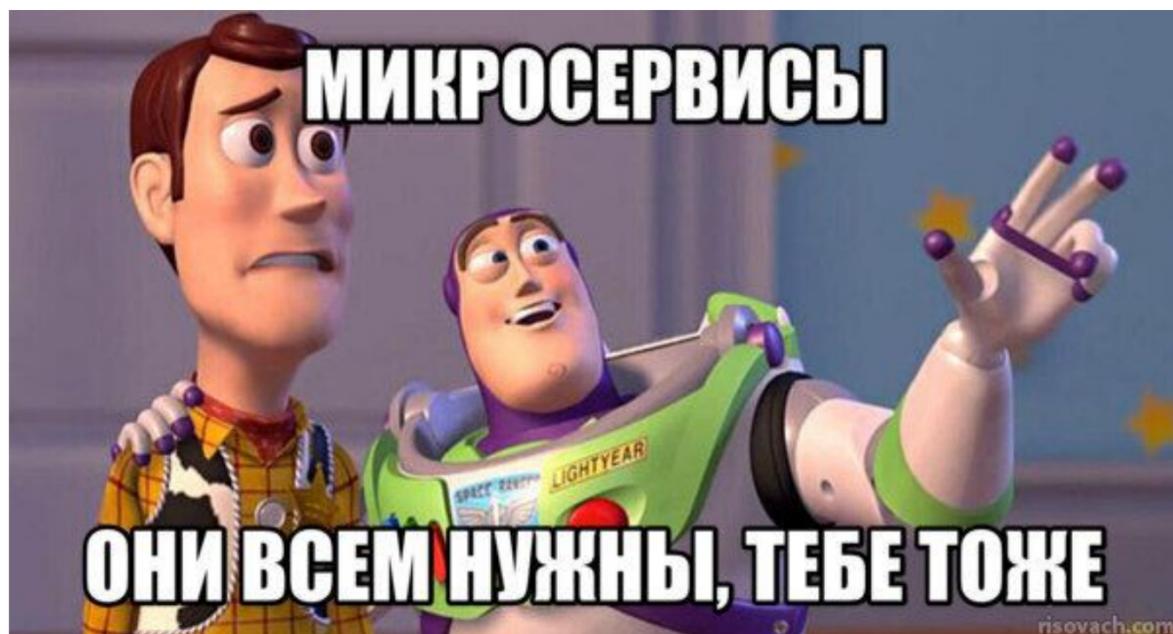
- Использование декларативного подхода к конфигурации
- Контроль версий и отслеживание изменений
- Автоматическая настройка и проверка совместимости
- Минимизация ручных операций
- Повышение точности и надежности интеграций
- Мониторинг пути прохождения данных
- Анализ и оптимизация трафика

# Переход на App.Farm— собственную разработку



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

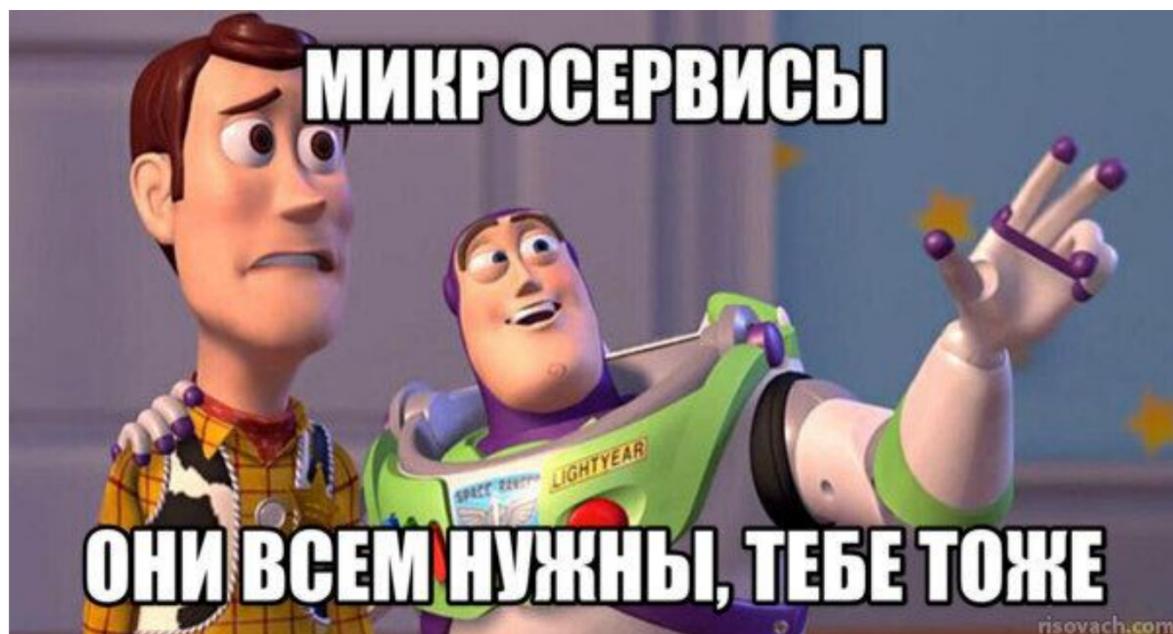
# Переход на App.Farm— собственную разработку



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

- Централизованное логирование

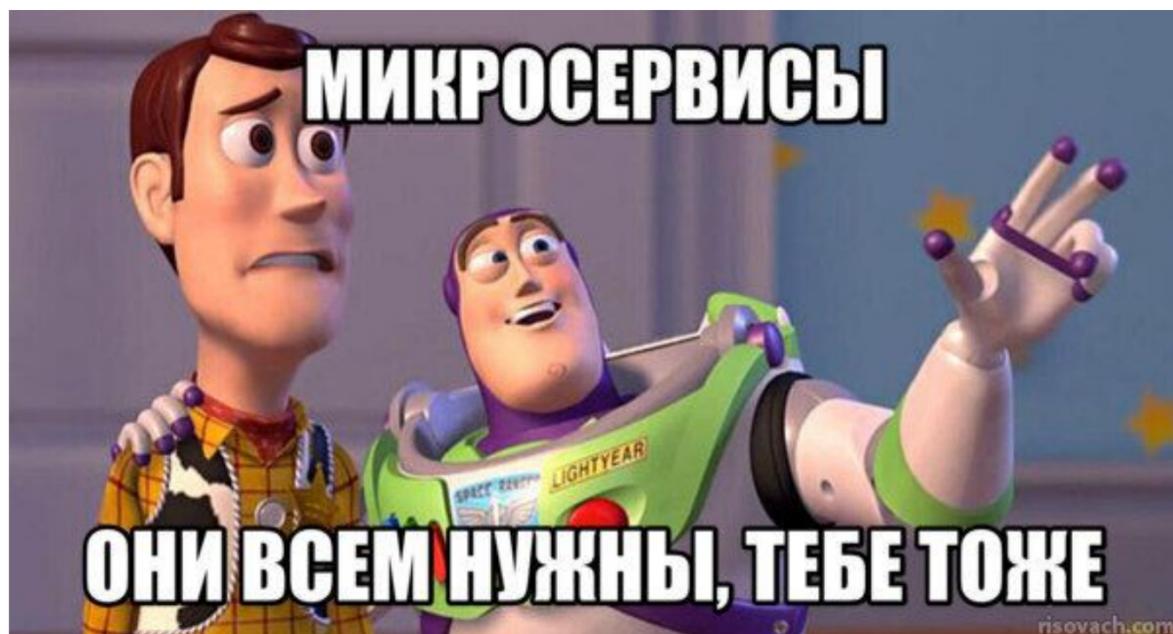
# Переход на App.Farm— собственную разработку



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

- Централизованное логирование
- Трассировка запросов

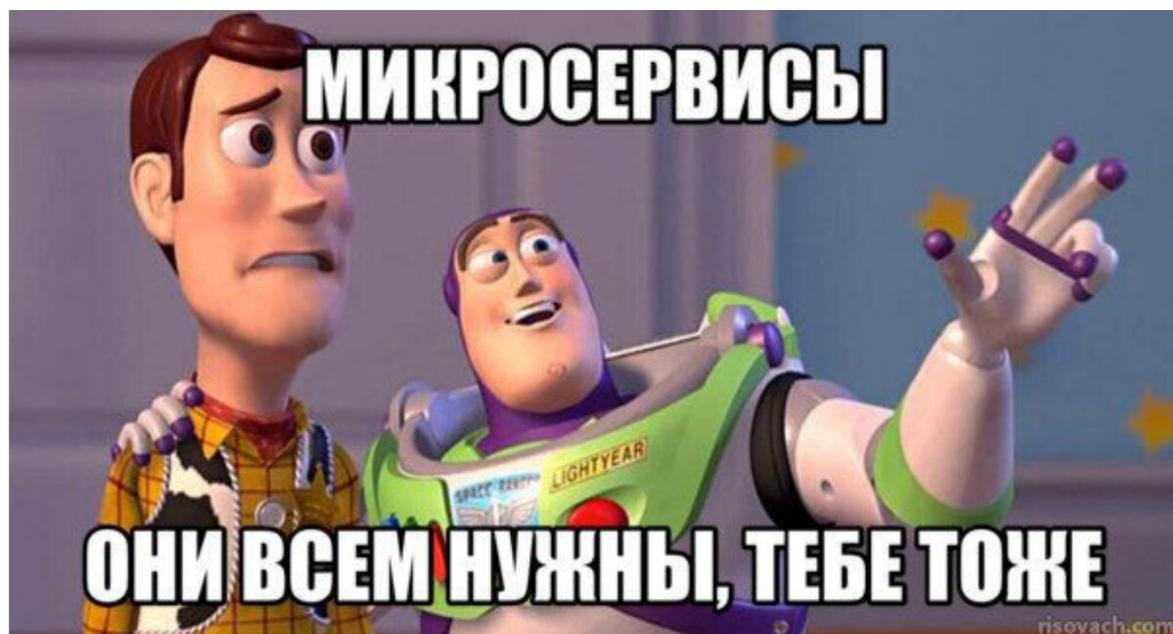
# Переход на App.Farm— собственную разработку



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

- Централизованное логирование
- Трассировка запросов
- Автоматизация тестирования и деплоя

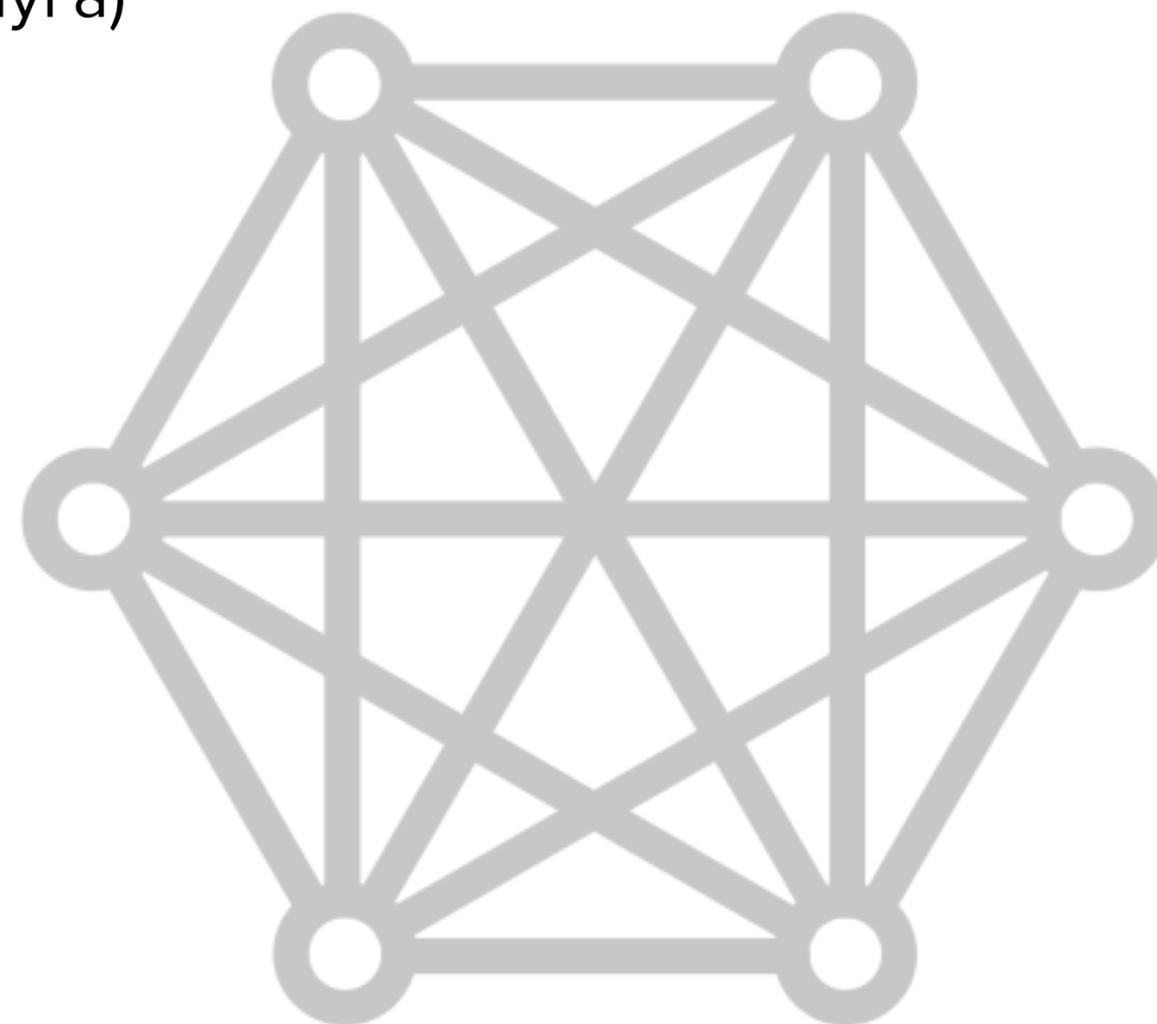
# Переход на App.Farm— собственную разработку



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

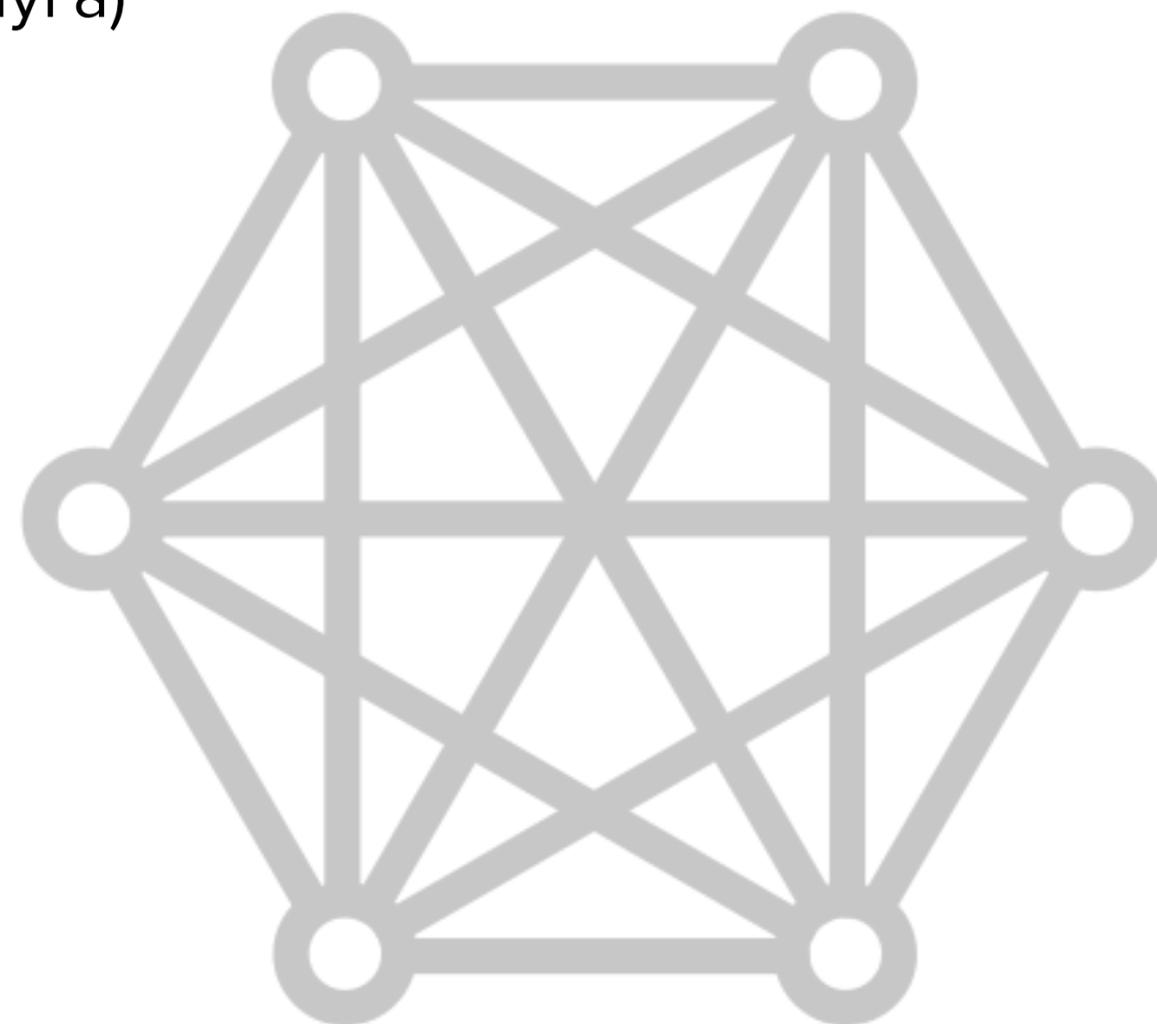
- Централизованное логирование
- Трассировка запросов
- Автоматизация тестирования и деплоя
- Использование Docker контейнеров

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



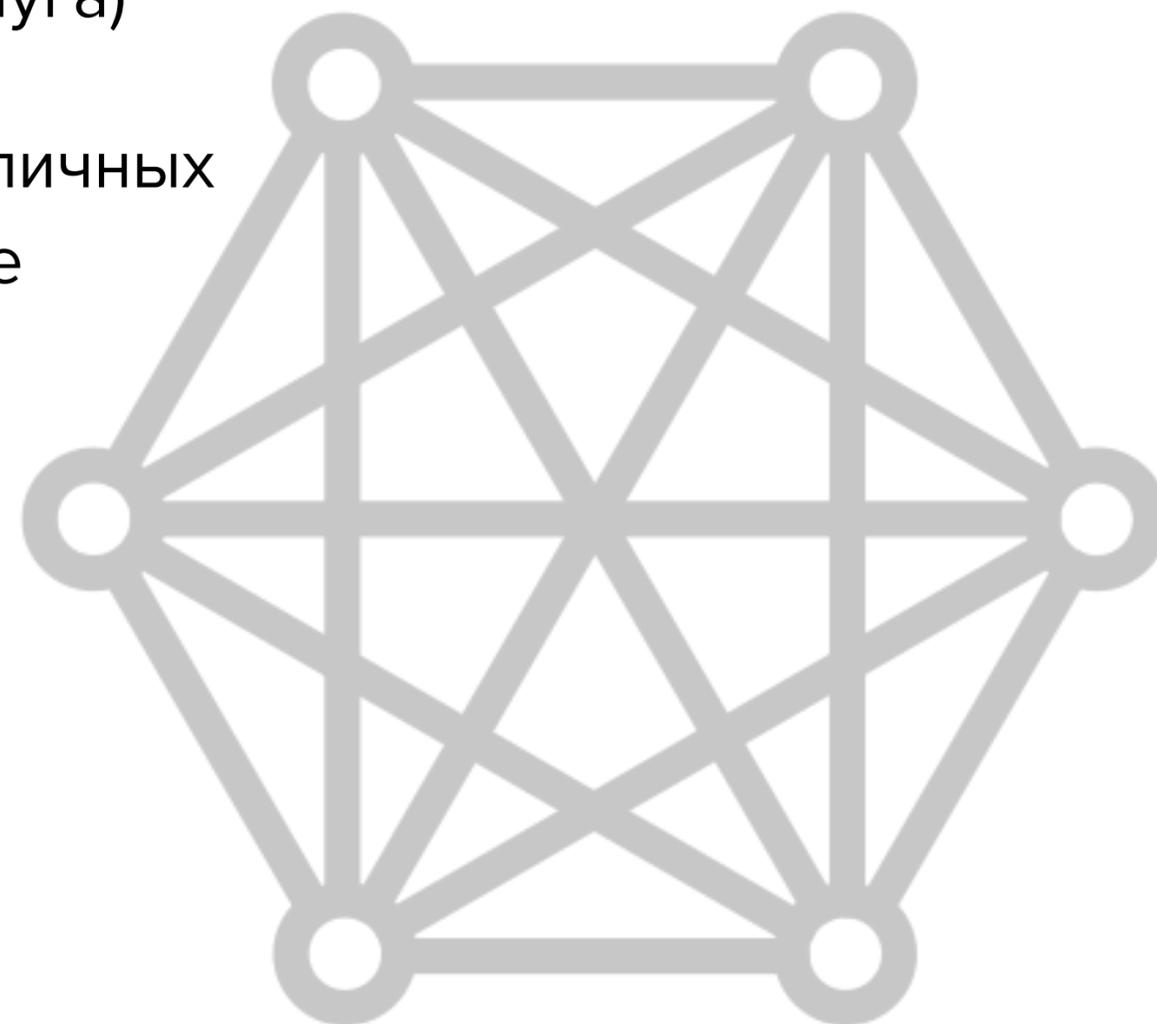
App.Farm – это PaaS (Platform as a Service / Платформа как услуга)

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.

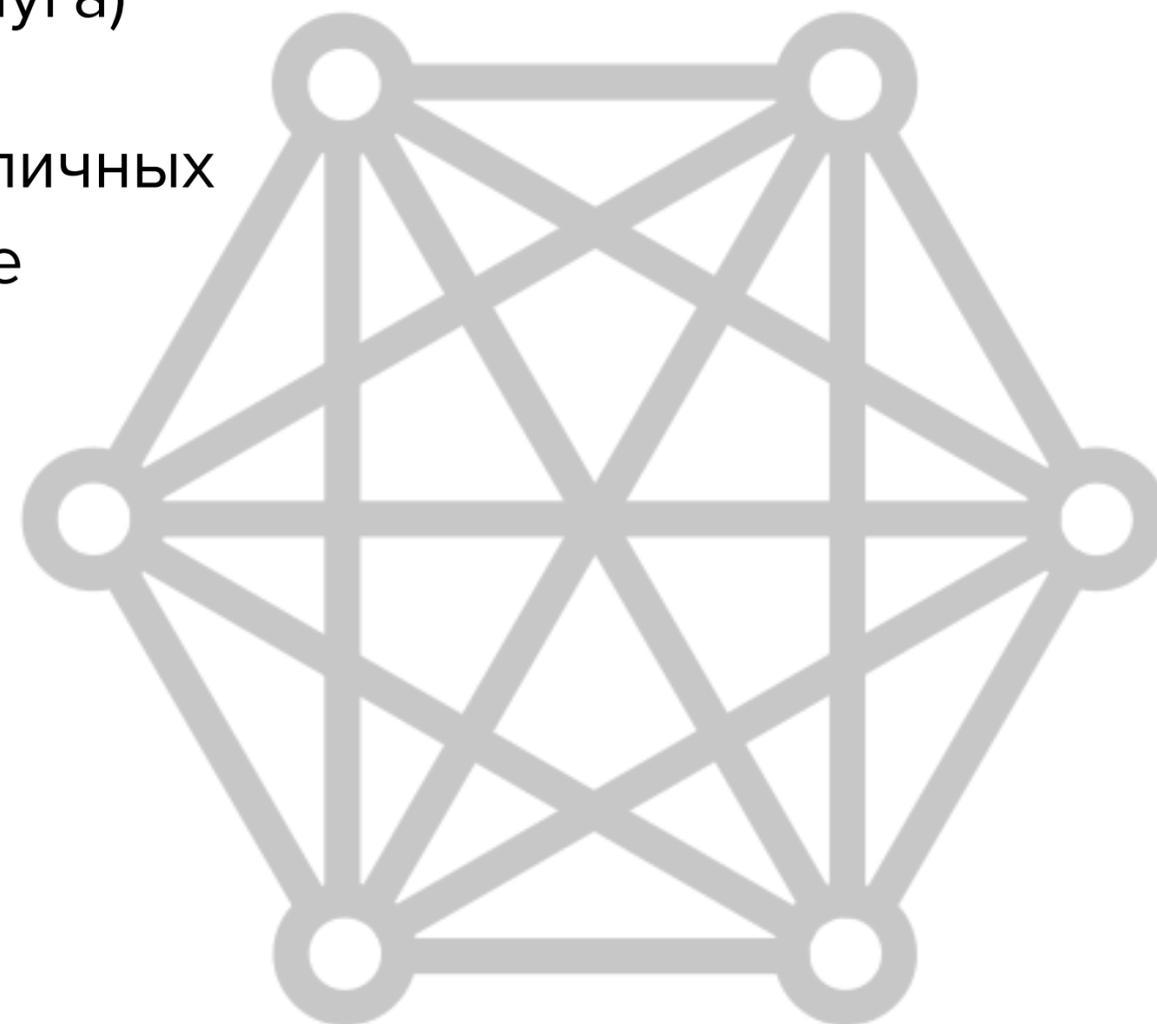


App.Farm – это PaaS (Platform as a Service / Платформа как услуга)

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

Т.е. App.Farm не предоставляет Kubernetes, Nexus, Vault, Elasticsearch или другие подобные компоненты как сервис.

Вместо этого есть API

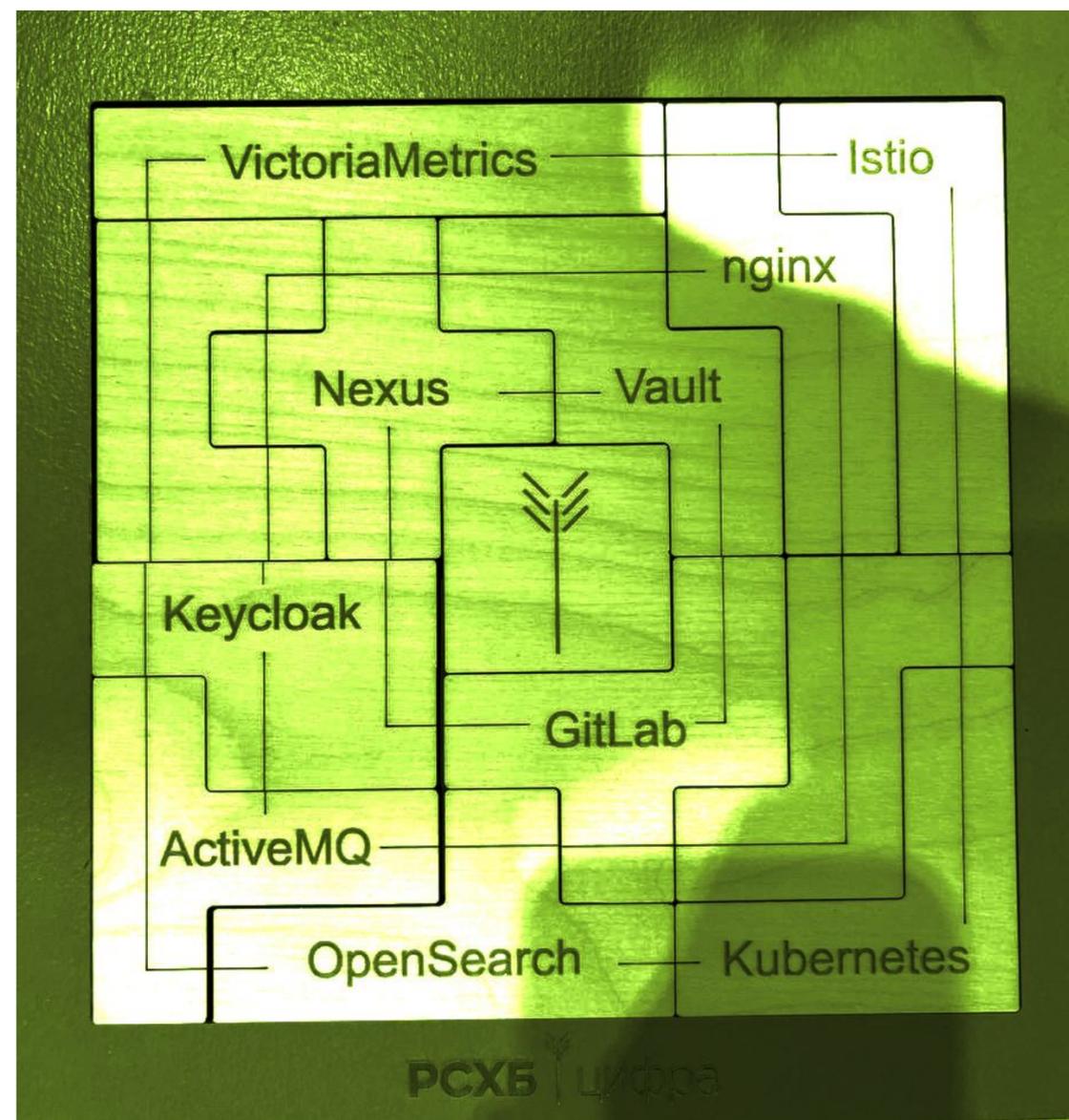


App.Farm построена на следующем стеке открытых решений:

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



# 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:

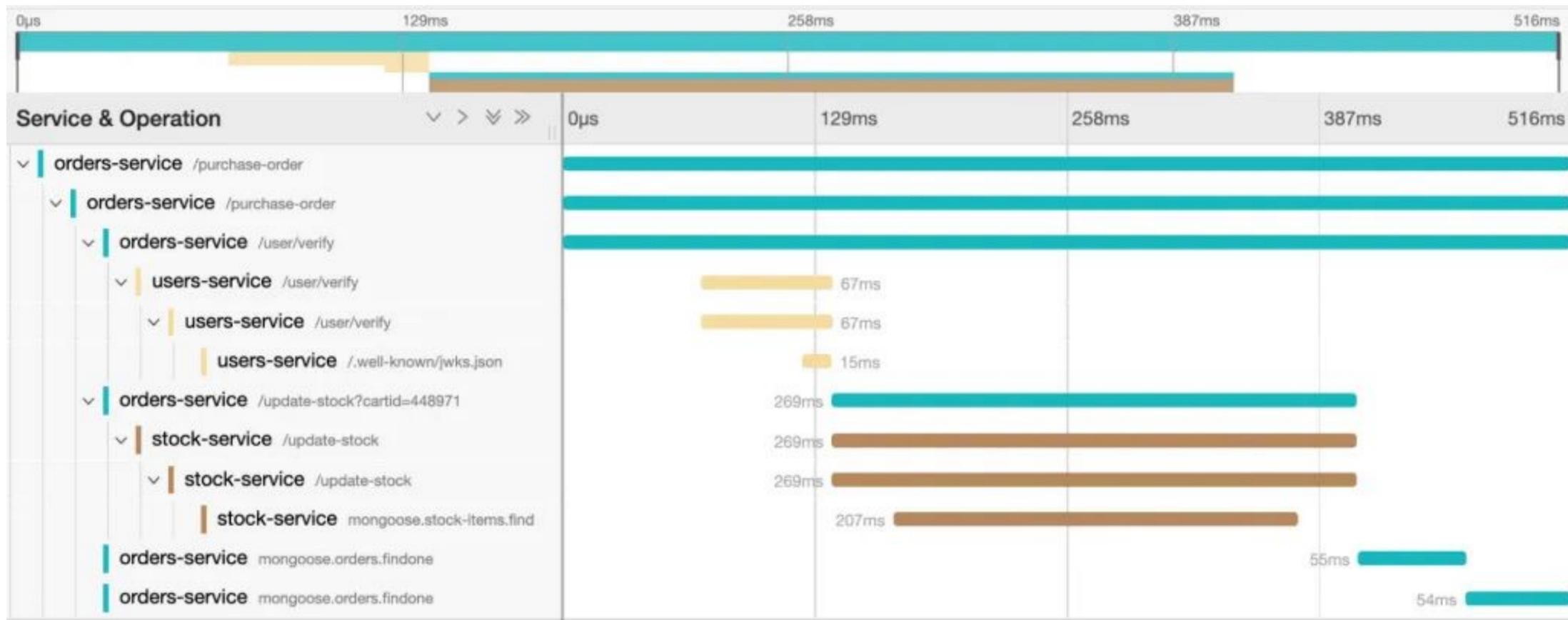
- трассировка запросов
- анализ логов
- мониторинг ресурсов



# 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 проследживать, какие сервисы взаимодействуют между собой, и, таким образом, определить, где возможны задержки или проблемы.
- Время выполнения:



# Jaeger

Цель: Профилирование и трассировка запросов

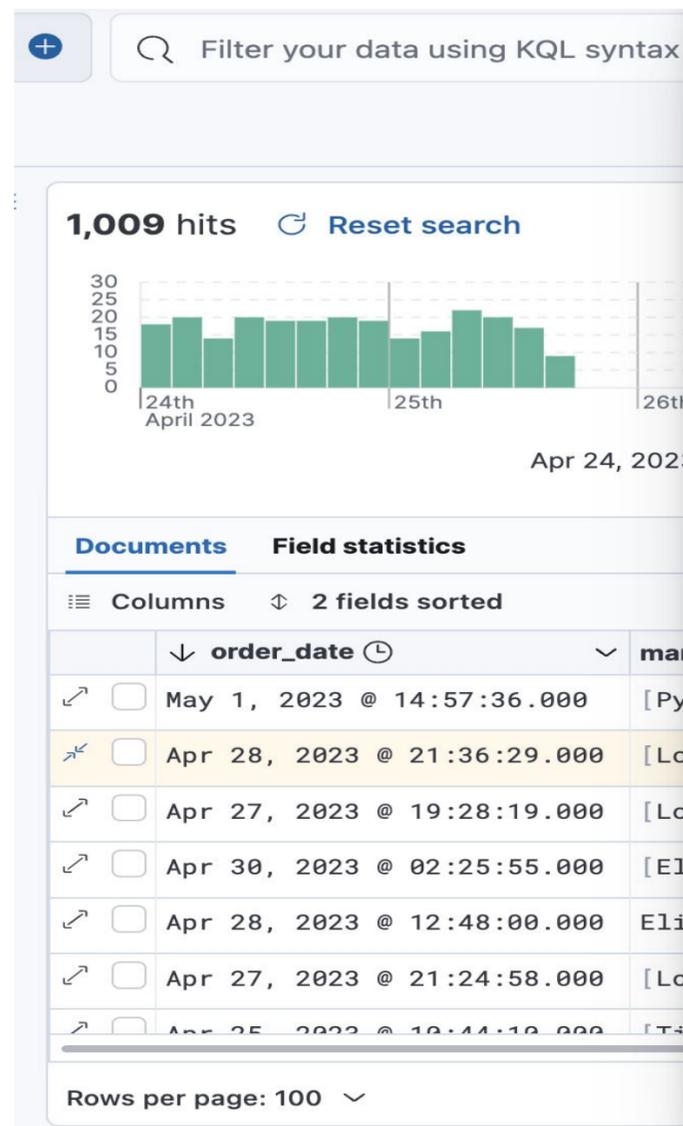
- Трассировка запросов: Jaeger позволяет визуализировать пути запросов через систему, отслеживая их время выполнения на каждом этапе.
- Анализ зависимостей: Можно проследживать, какие сервисы взаимодействуют между собой, и, таким образом, определить, где возможны задержки или проблемы.
- Время выполнения: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



## Expanded document

View: [Single document](#) [Surrounding documents](#) [?](#) [K](#) [<](#) [2 of 500](#) [>](#) [|](#)

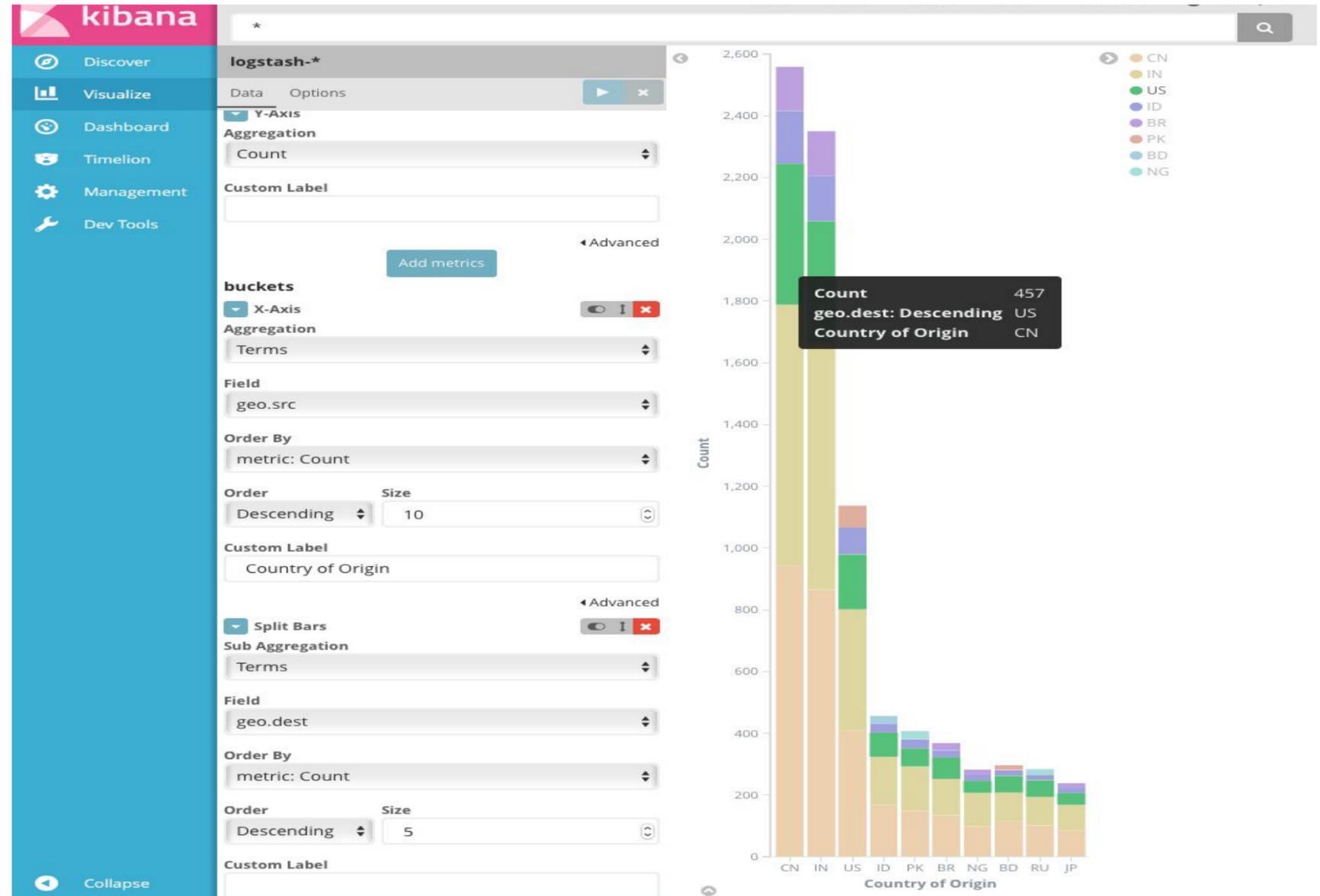
**Table** **JSON**

Actions	Field	Value
	<b>k</b> _id	ub1puocBs3kYv_hWFnt_
	<b>k</b> _index	kibana_sample_data_ecommerce
	<b>#</b> _score	-
<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<b>t</b> category	[Men's Clothing, Women's Accessories, Men's Shoes]
	<b>k</b> currency	EUR
	<b>k</b> customer	Abbott, J
	<b>t</b> customer_first_name	Jackson
	<b>t</b> customer_full_name	Jackson Abbott
	<b>k</b> customer_gender	MALE
	<b>k</b> customer_id	13
	<b>t</b> customer_last_name	Abbott

# Kibana

Цель: Анализ логов и мониторинг состояния приложения

- Визуализация данных:



# Kibana

Цель: Анализ логов и мониторинг состояния приложения



kibana

- 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



# kibana

- 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 событиях, 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

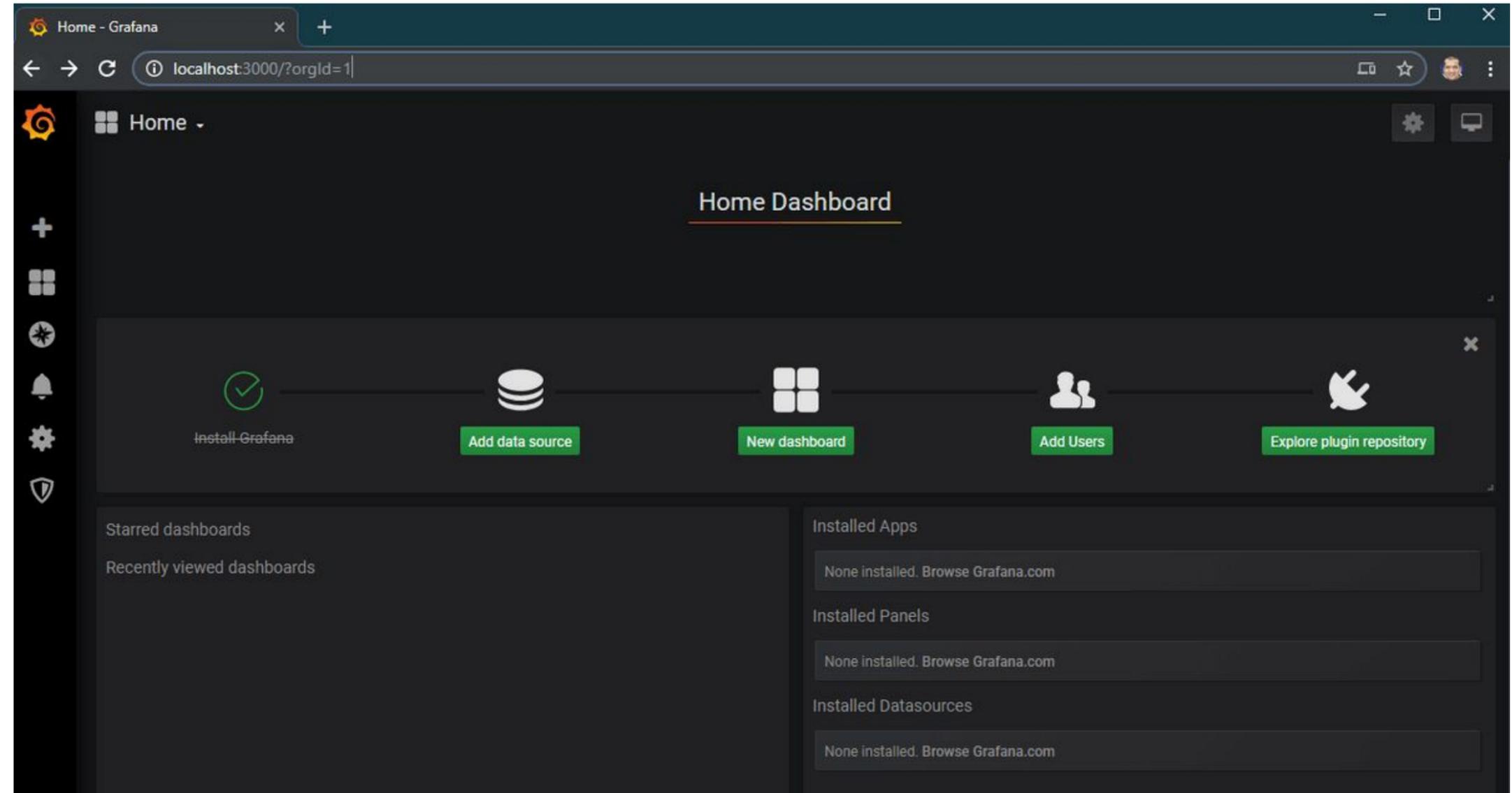
- Метрики производительности: Grafana позволяет собирать и визуализировать метрики производительности в реальном времени, такие как загрузка CPU, использование памяти, задержки запросов и т.д.
- Анализ нагрузки: С Grafana можно отслеживать, как распределяется нагрузка между различными компонентами системы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

Search JSON File

Service (852)

dbo2-

dbo2-flow.isys-flow-adapters

Operation (6)

all

Tags ?

http.status\_code=200 error=true

Lookback

Last Hour

Max Duration

Min Duration

Limit Results

20

Find Traces



# Пример в Jaeger:

Search JSON File

Service (852)  
dbo2-flow.isys-flow-adapters

Operation (15)  
all

Tags ⓘ  
http.status\_code=200 error=true

Lookback  
Last Hour

Max Duration  
e.g. 1.2s, 100ms, 500us

Min Duration  
5s

Limit Results  
30

Find Traces



# Пример в Jaeger:

**Search** JSON File

Service (852)  
dbo2-flow.isys-flow-adapters

Operation (15)  
all

Tags ⓘ  
http.status\_code=200 error=true

Lookback  
Custom Time Range

Start Time ⓘ  
30.09.2024 00:00

End Time ⓘ  
30.09.2024 15:51

Max Duration  
e.g. 1.2s, 100ms, 500us

Min Duration  
5s

Limit Results  
30

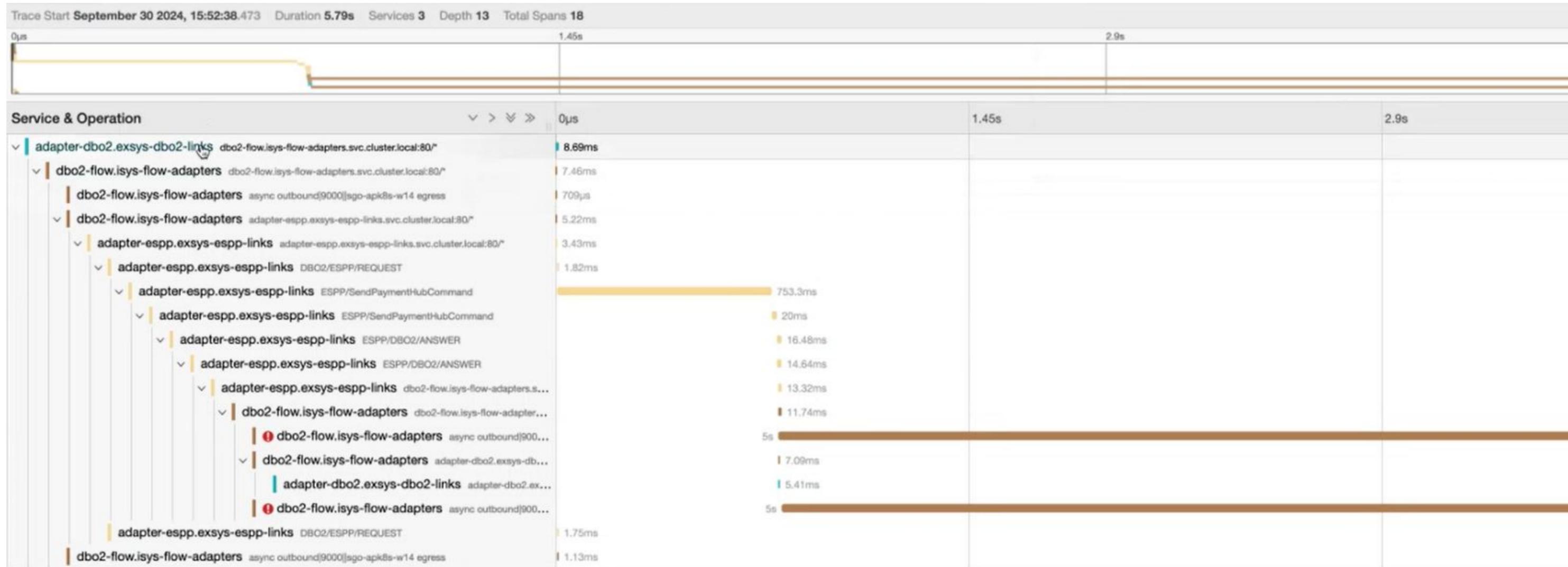
**Find Traces**

**30 Traces**

Compare traces by selecting result items

- adapter-dbo2.exsys-dbo2-links: DBO2/ESPP/REQUEST a16bd6d  
10 Spans 2 Errors  
adapter-dbo2.exsys-dbo2-links (3) adapter-espp.exsys-espp-links (3) dbo2-flow.isys-flow-adapters (4)
- adapter-dbo2.exsys-dbo2-links: dbo2-flow.isys-flow-adapters.svc.cluster.local:80/\* eee59ce  
18 Spans 2 Errors  
adapter-dbo2.exsys-dbo2-links (2) adapter-espp.exsys-espp-links (8) dbo2-flow.isys-flow-adapters (8)
- adapter-dbo2.exsys-dbo2-links: DBO2/ESPP/REQUEST 2dee000  
19 Spans 2 Errors  
adapter-dbo2.exsys-dbo2-links (4) adapter-espp.exsys-espp-links (8) dbo2-flow.isys-flow-adapters (7)
- adapter-dbo2.exsys-dbo2-links: DBO2/BISQUIT/REQUEST 34c0493  
20 Spans 2 Errors  
adapter-cft.exsys-cft-links (6) adapter-dbo2.exsys-dbo2-links (4) cft-flow.isys-flow-adapters (4) dbo2-flow.isys-flow-adapters (4)

# Пример в Jaeger:



# Пример в Jaeger:

The screenshot displays a Jaeger trace with the following structure:

- adapter-espp.exsys-espp-links ESPP/DBO2/ANSWER (16.48ms)
- adapter-espp.exsys-espp-links ESPP/DBO2/ANSWER (14.64ms)
- adapter-espp.exsys-espp-links dbo2-flow.isys-flow-adapters.s... (13.32ms)
- dbo2-flow.isys-flow-adapters dbo2-flow.isys-flow-adapt... (11.74ms)
- dbo2-flow.isys-flow-adapters async outbound|900...** (5s)

The selected span, **async outbound|**, shows the following tags:

Tag	Value
component	proxy
error	true
http.protocol	HTTP/1.1
http.status_code	504
internal.span.format	zipkin
peer.address	10.27.
response_flags	UT
span.kind	client
upstream_address	10.27.
upstream_cluster	outbound
upstream_cluster.name	outbound

The **Process** section is also visible but not expanded.

# 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

# 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

# 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

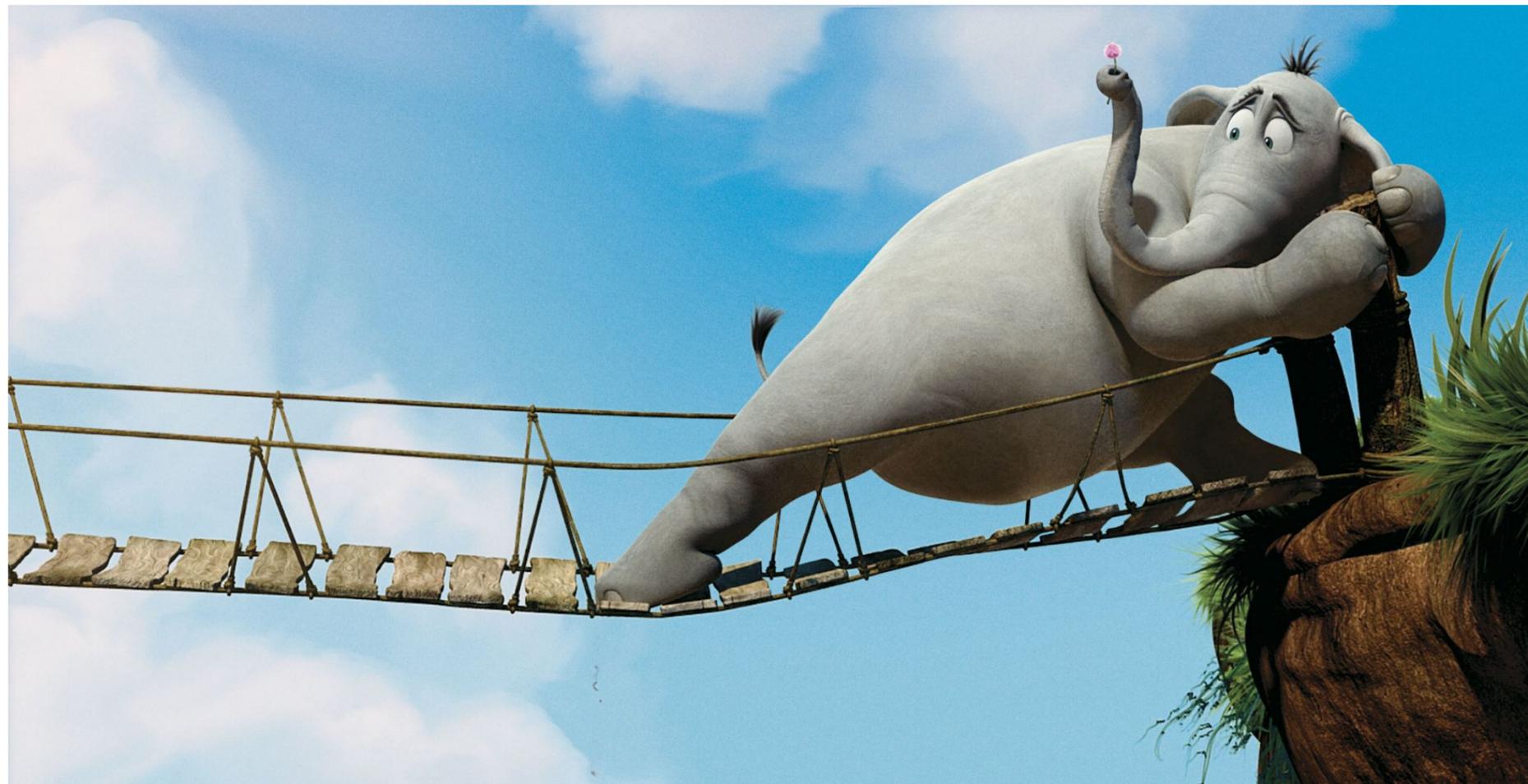
- Мигрируем в App.Farm наши собственные модули (написаны на C#)
- Встраиваем в них нужные метрики
- Создаем дашборд, где будет видно типовые значения
- Метрики смотрим в Grafana
- Логирование тоже можно настроить!

# Дорогу осилит идущий!

Не бойтесь вступать на дорогу импортозамещения.

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

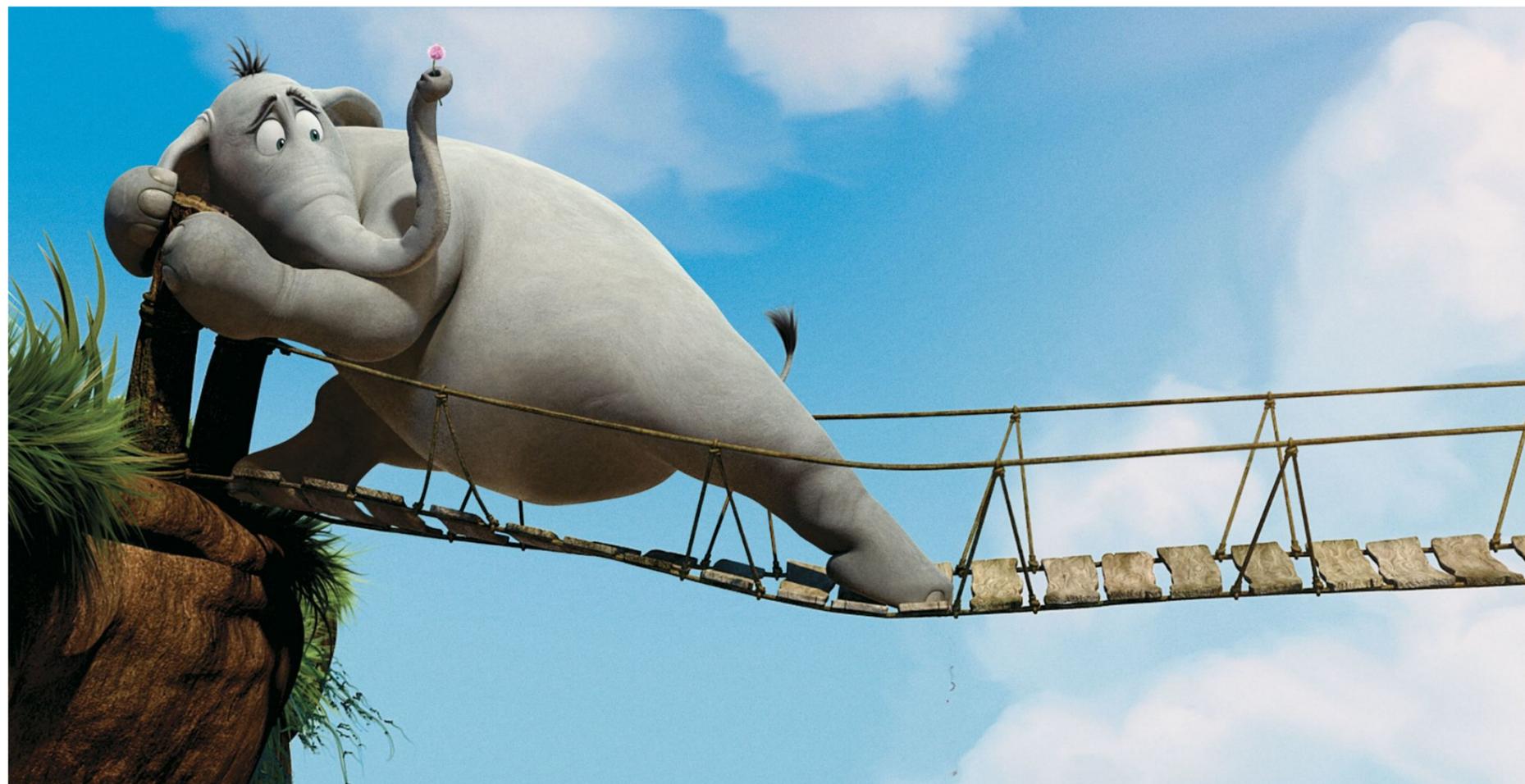
И неизбежное повышение эффективности в будущем



# Дорогу осилит идущий!

«Обрастать костылями» — не наш путь

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



# Дорогу осилит идущий!

Всем спасибо, с вами был Ахметов Андрей



[@UFA\\_Akhmetov\\_Andrey](https://t.me/@UFA_Akhmetov_Andrey)



подробнее про App.Farm

# Спасибо за внимание!

