Мультитенантная архитектура в SaaS-приложениях

Юра Тюрин

- CTO в МД Аудит
- 12+ лет в IT
- Ментор

О чем будем говорить:

- Предпосылки внедрения мультитенантности
- Что такое мультитенантность
- Как мы внедряли мультитенантность
- Итоги нескольких лет в продакшене

МД АУДИТ

помогаем розничным сетям управлять качеством на торговых точках

8+ лет на рынке

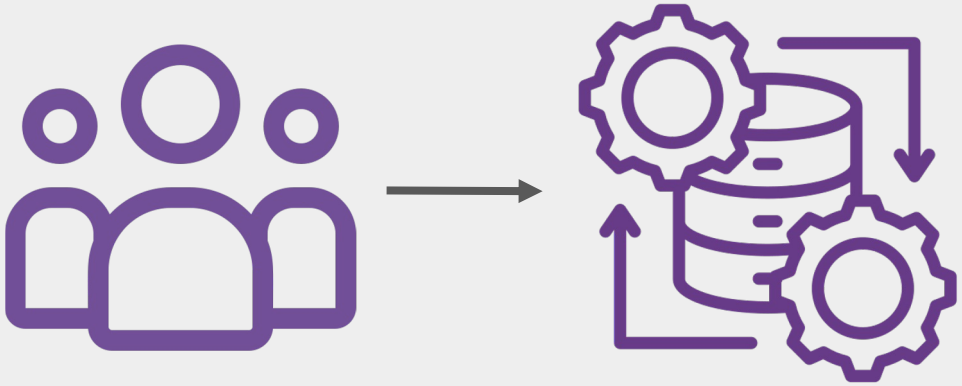
150+ клиентов (5 - 10 000 пользователей на клиента)

99% клиентов используют SaaS модель подписки

30+ микросервисов

Предпосылки внедрения **мультитенантности**

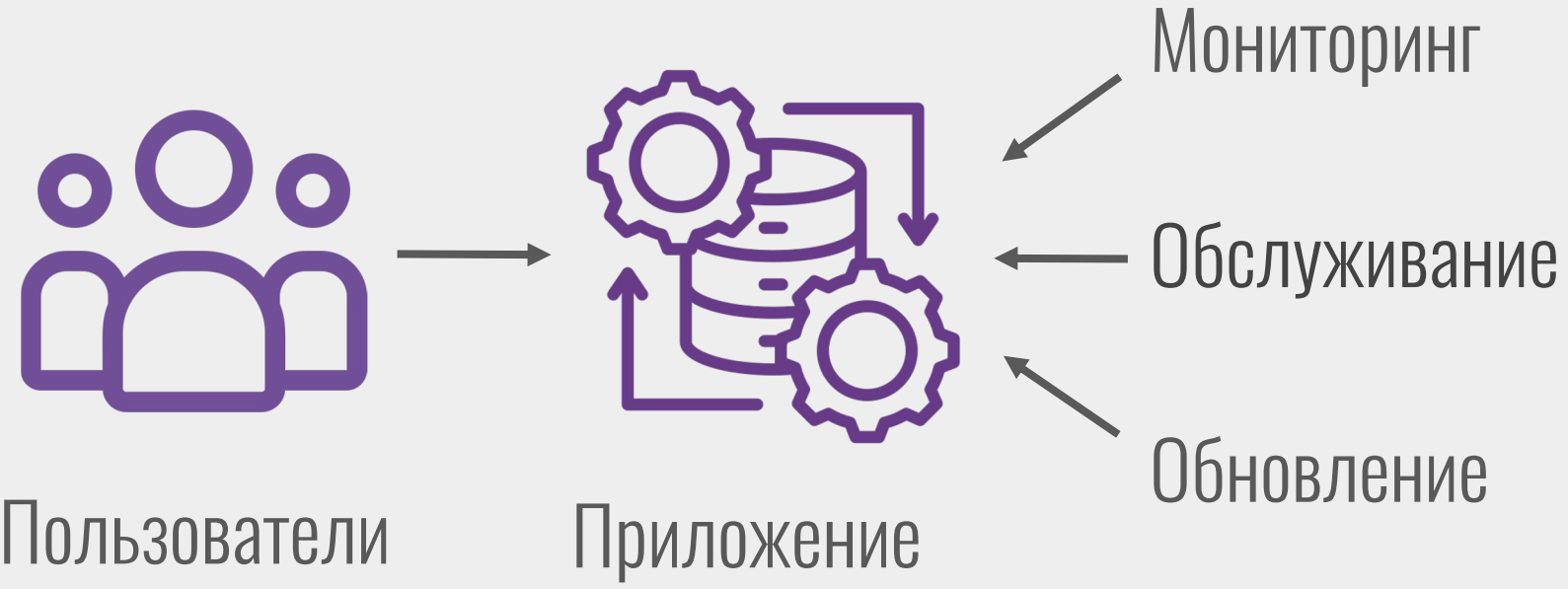
Клиент #1



Пользователи

Приложение

Клиент #1


Мониторинг
Обслуживание
Обновление
Пользователи
Приложение

Клиент #1

Пользователи

Приложение

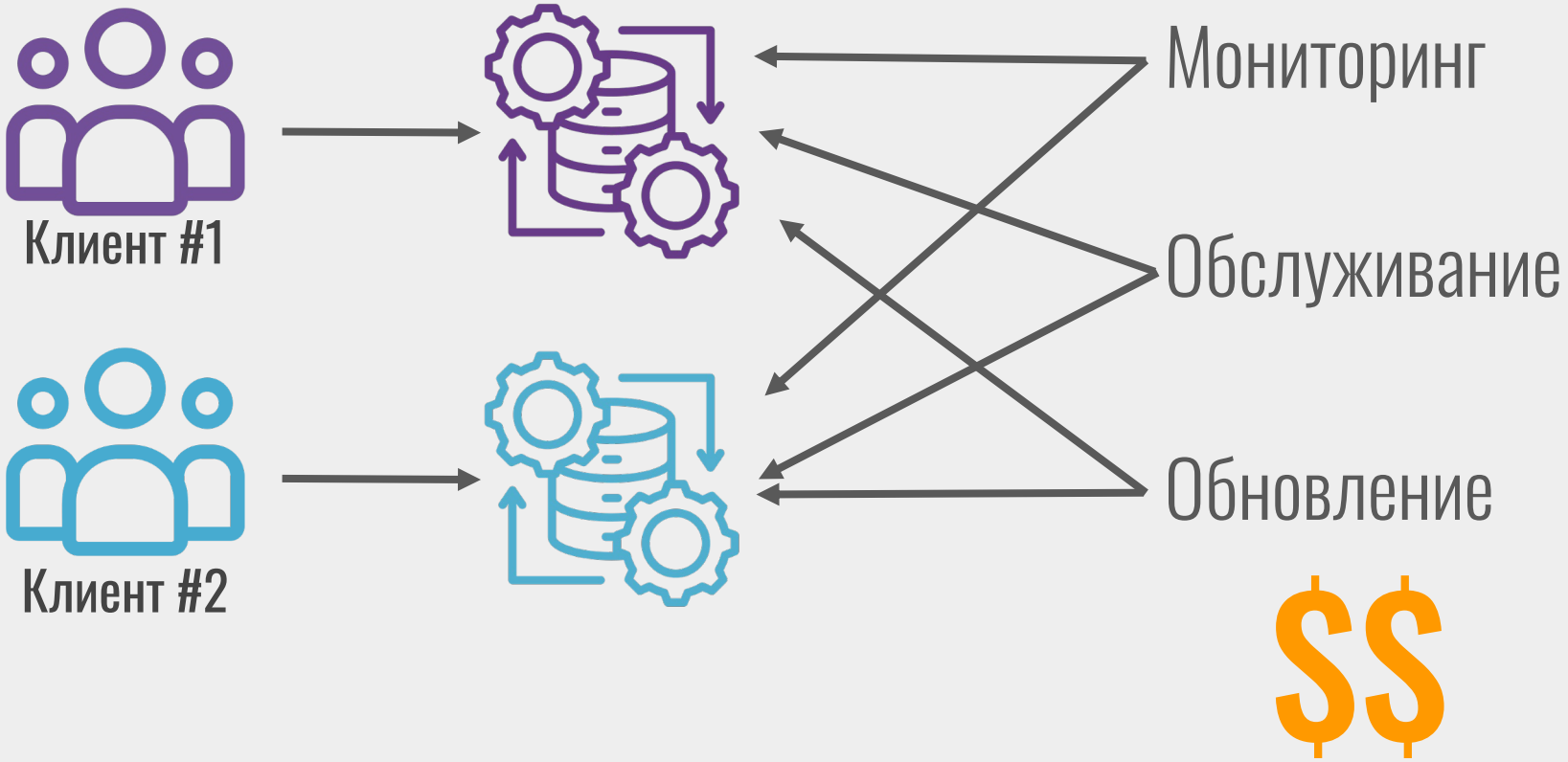
Мониторинг

Обслуживание

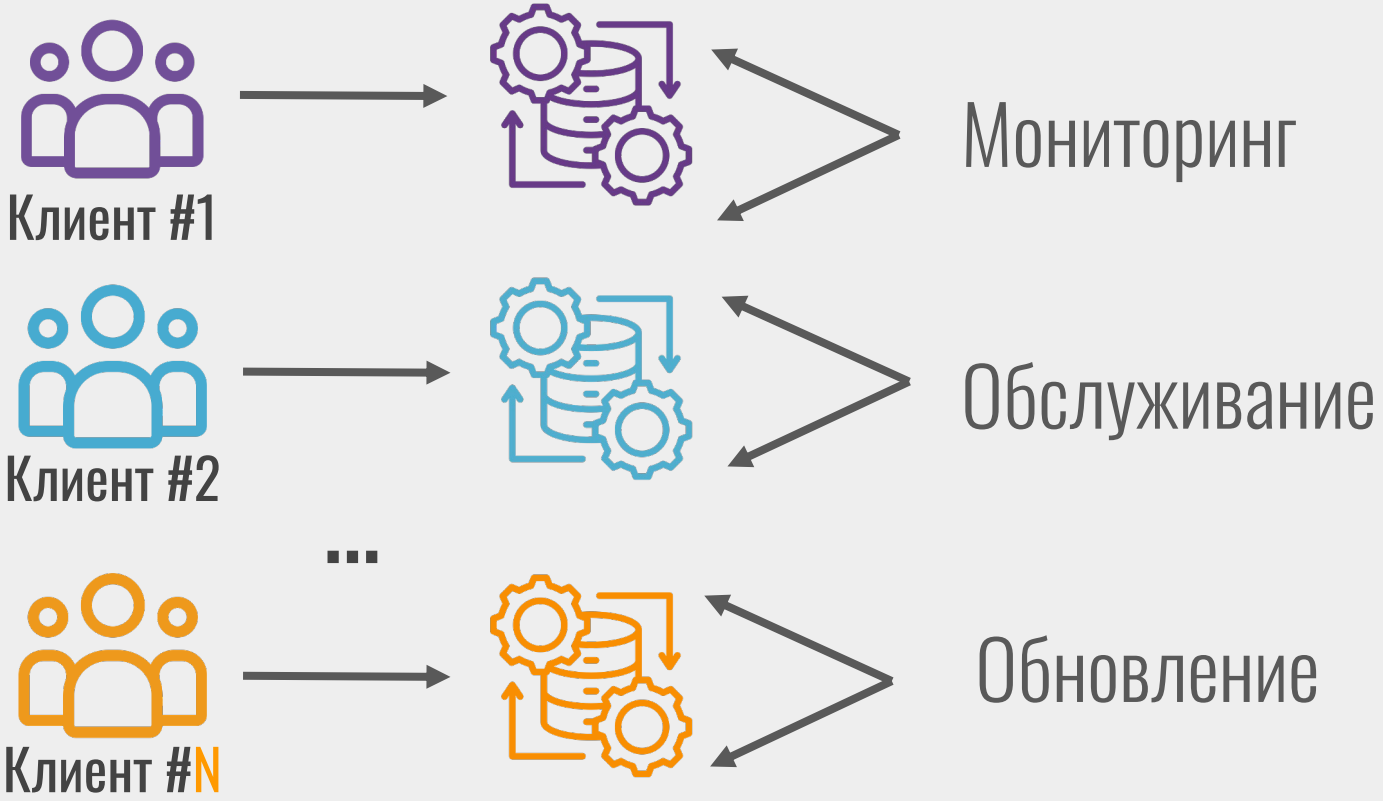
Обновление

$

Масштабирование



Масштабирование



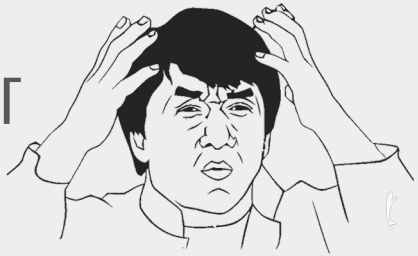
Масштабирование

Масштабирование

$$$$$...

Клиент #1

Мониторинг



Клиент #2

Обслуживание

...

Клиент #N

Обновление

Минусы

1. Сложно разворачивать, поддерживать и обновлять
2. Ресурсы нужно закладывать с учетом пиков нагрузки
3. Пробный период оплачивается поставщиком
4. Высокая итоговая стоимость инфраструктуры для клиента

Автоматизация vs ?

- Автоматизация инфраструктуры
- Автоматизация развертывания и обновления
- Автоскейлинг
- Авто...

Что такое

Мультитенантность

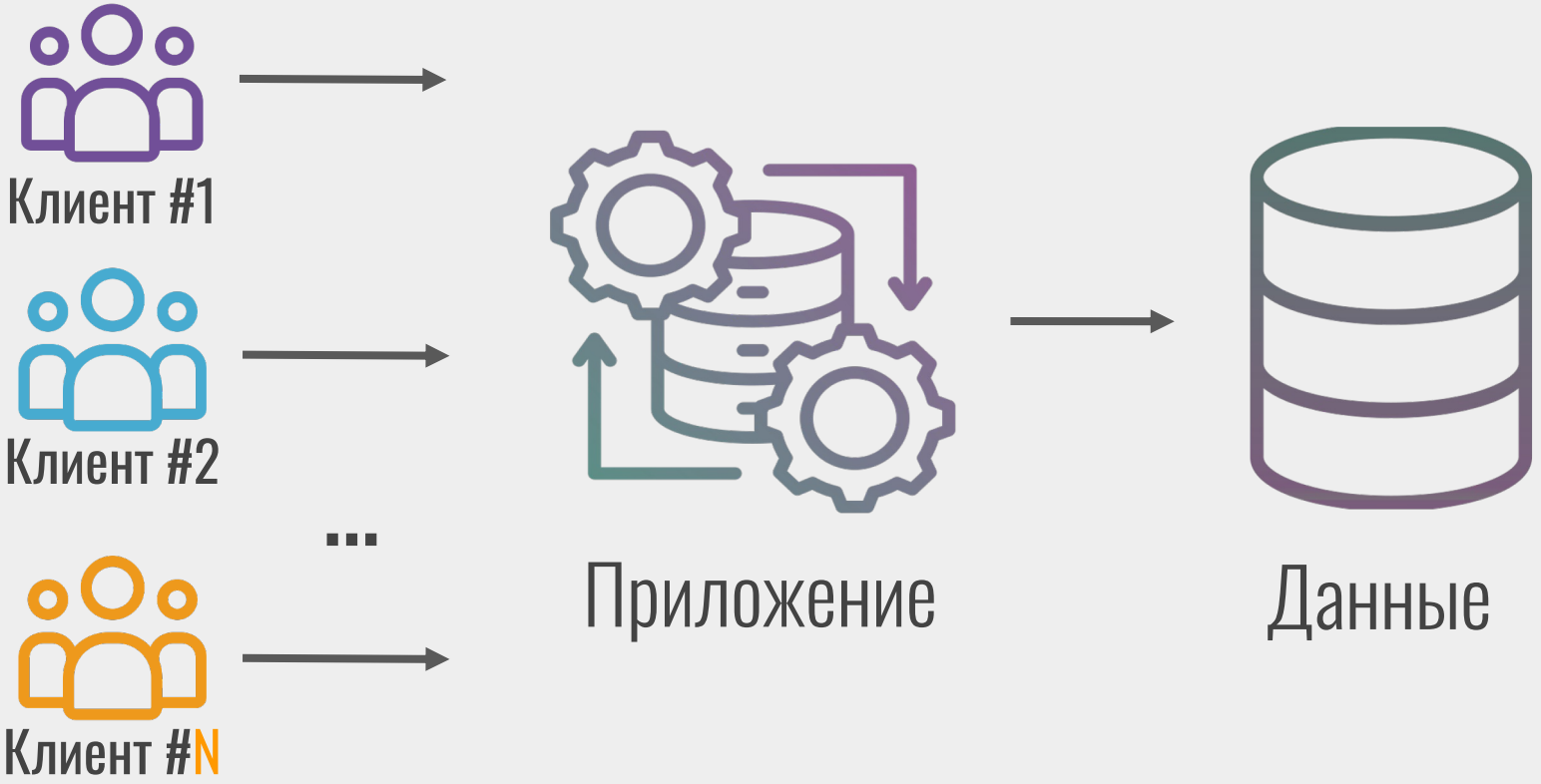
Мультитенантная архитектура

Tenant (англ.) - арендатор

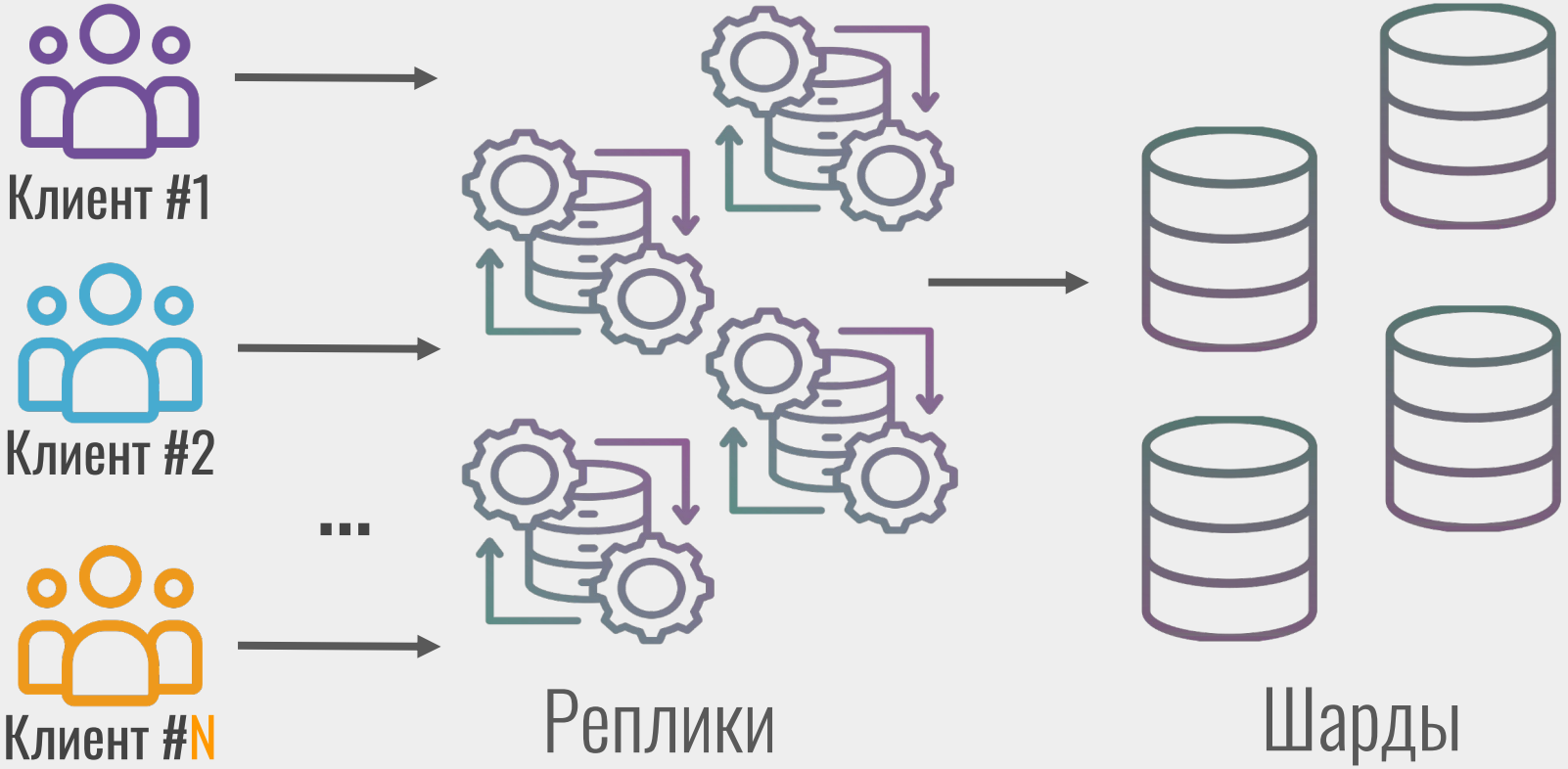
Мультитенантная архитектура

Multitenancy - возможность использования объекта несколькими арендаторами

Мультитенантность



Мультитенантность



Как мы внедряли
Мультитенантность

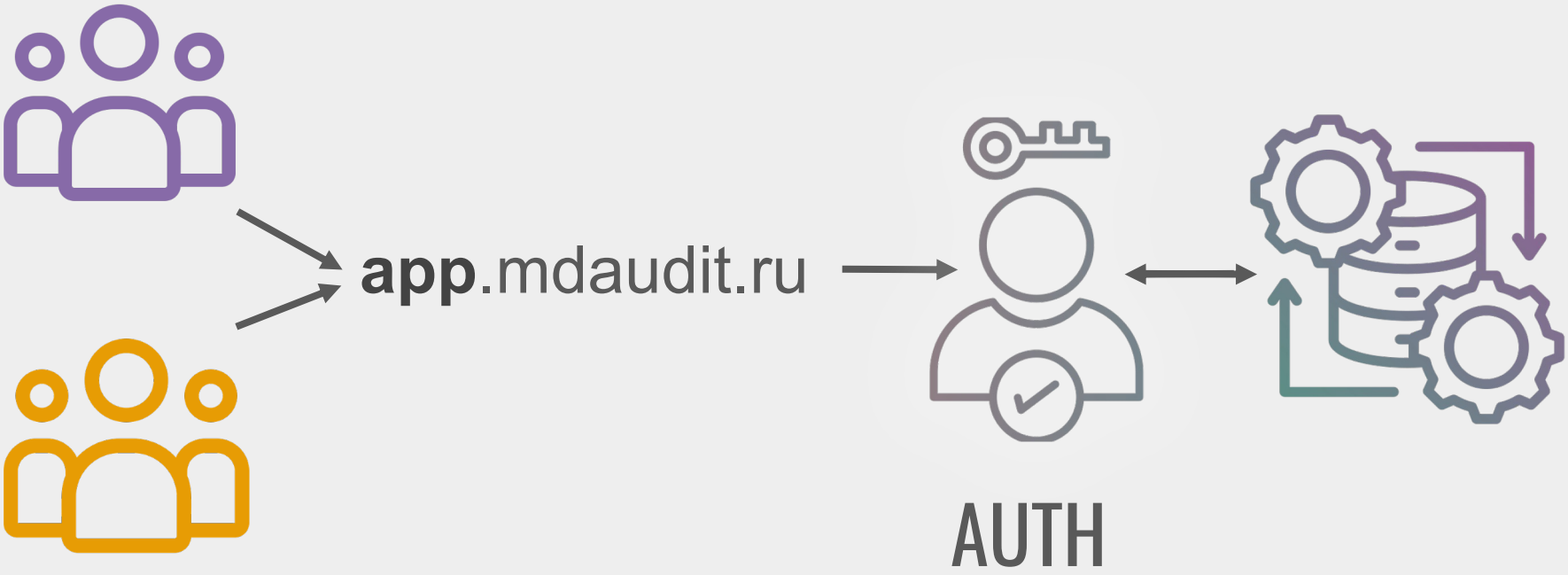
Какие задачи надо было решить?

1. Как пользователи попадают в свою систему?
2. Как система отличает одного клиента от другого?
3. Как хранить данные клиентов?

Как пользователи попадают в **свою** систему?


ikea.mdaudit.ru

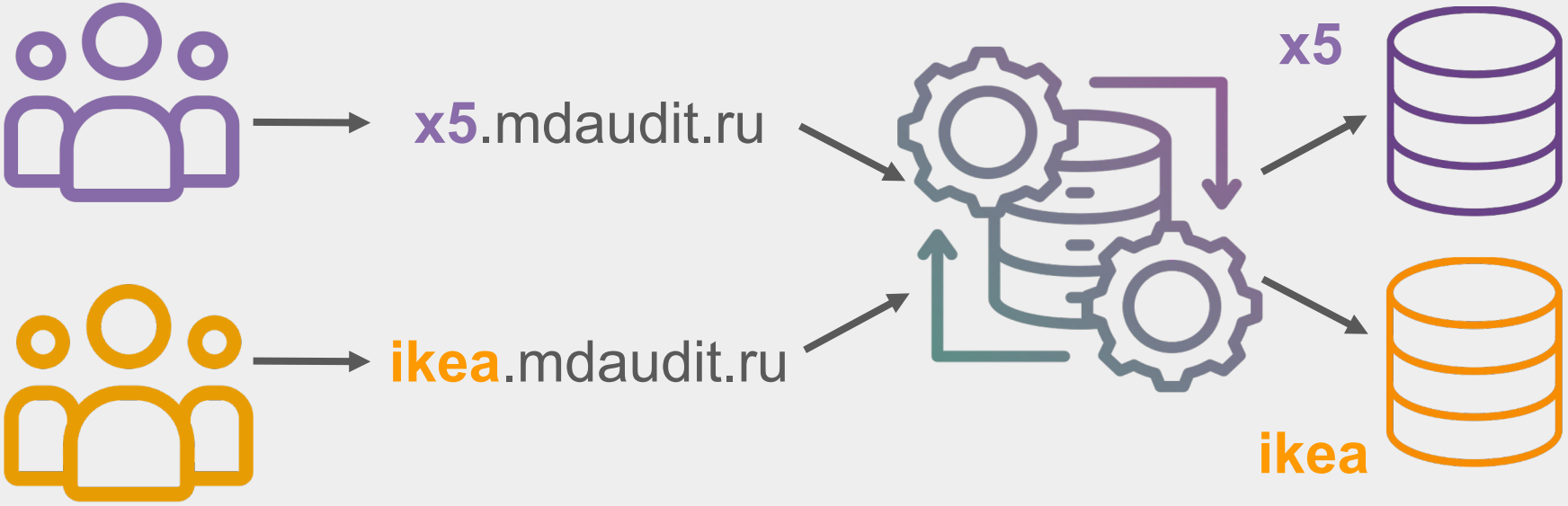
Вариант 1: Сквозная аутентификация



Минусы

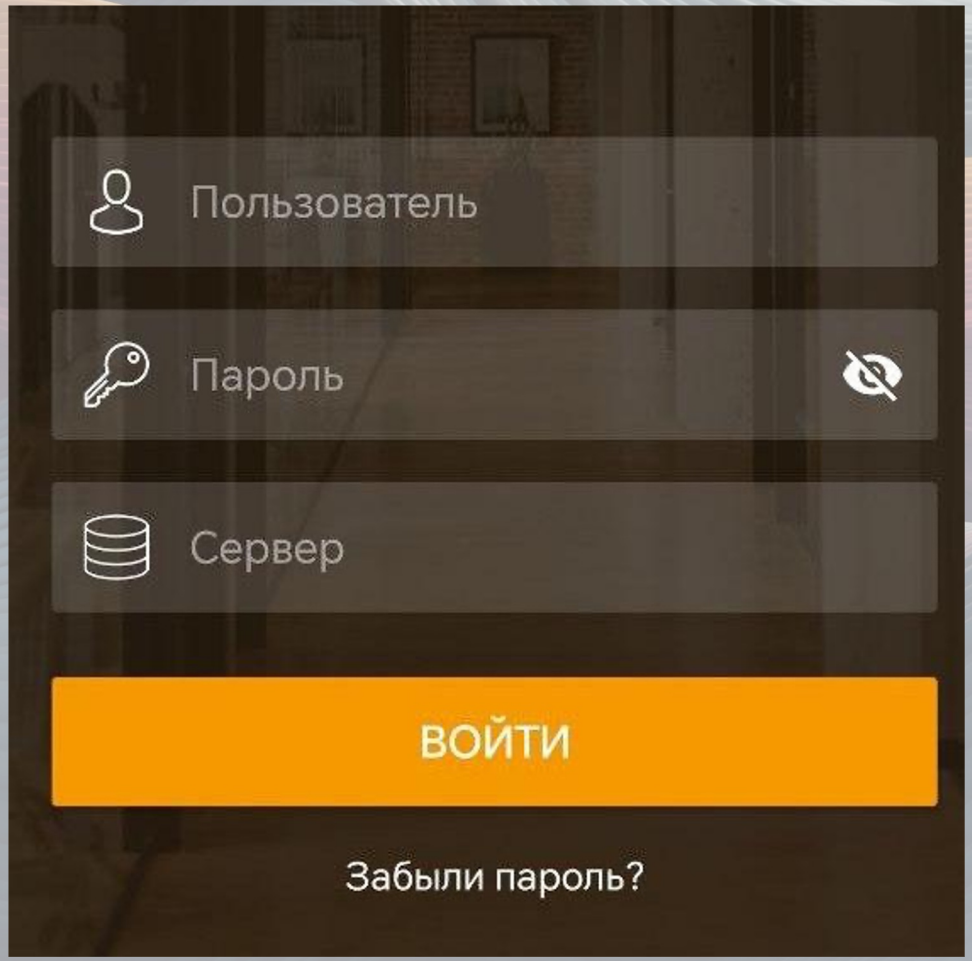
- Нужно переписать модуль аутентификации
- Нужно мигрировать пользователей
- 99.999% пользователей состоят в одной организации

Вариант 2: Оставляем как есть

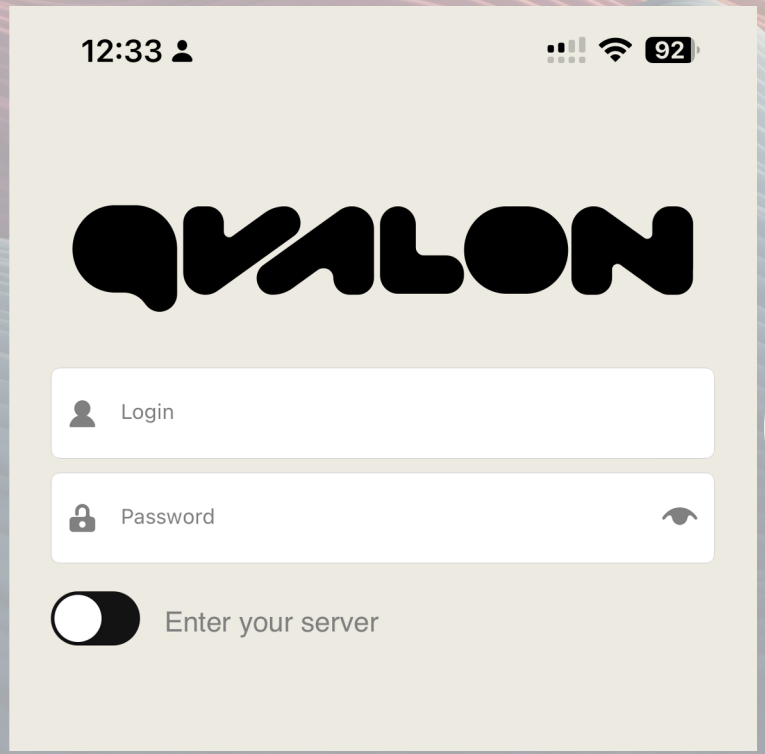


А что с мобильным приложением?

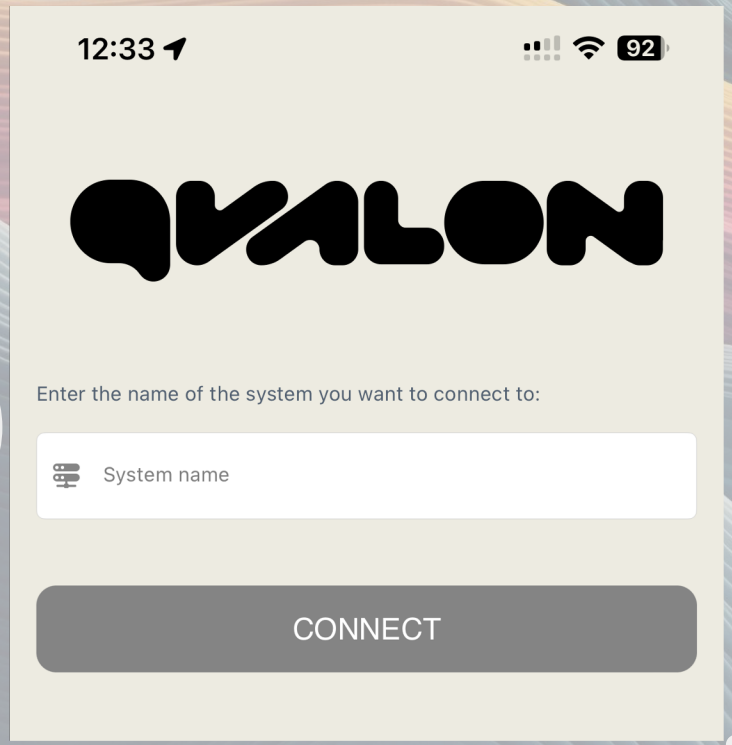
ДО



ПОСЛЕ


12:33
92
QVALON
Login
Password
Enter your server

0.01%


12:33
92
QVALON
Enter the name of the system you want to connect to:
System name
CONNECT

Как система отличает одного клиента от другого?

Как реализовать логику внутри приложения?

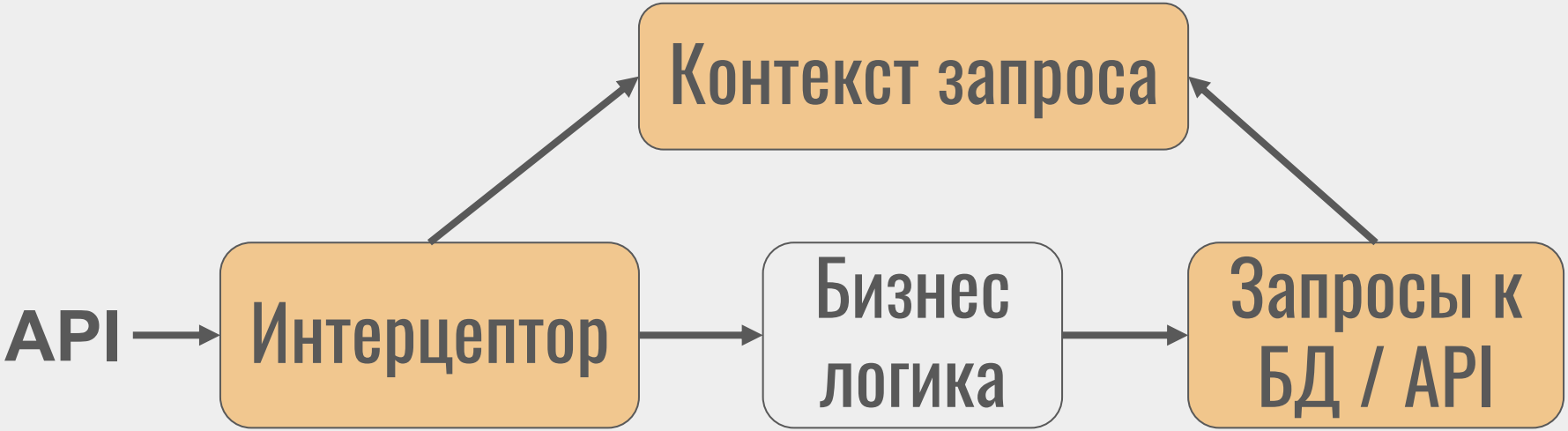
GET https://company1.mdaudit.ru/api/items

```
fun getItems(tenant: String) {
  …
  db.query(“select * from items where tenant = $tenant”)
  …
}
```

Проблемы

- Нужно доработать **ВСЕ** методы
- Высокая вероятность **ошибки**

Как реализовать логику внутри приложения?



До

GET https://company1.mdaudit.ru/api/items

```
fun getItems(tenant: String) {
  …
  db.query(“select * from items where tenant = $tenant”)
  …
}
```

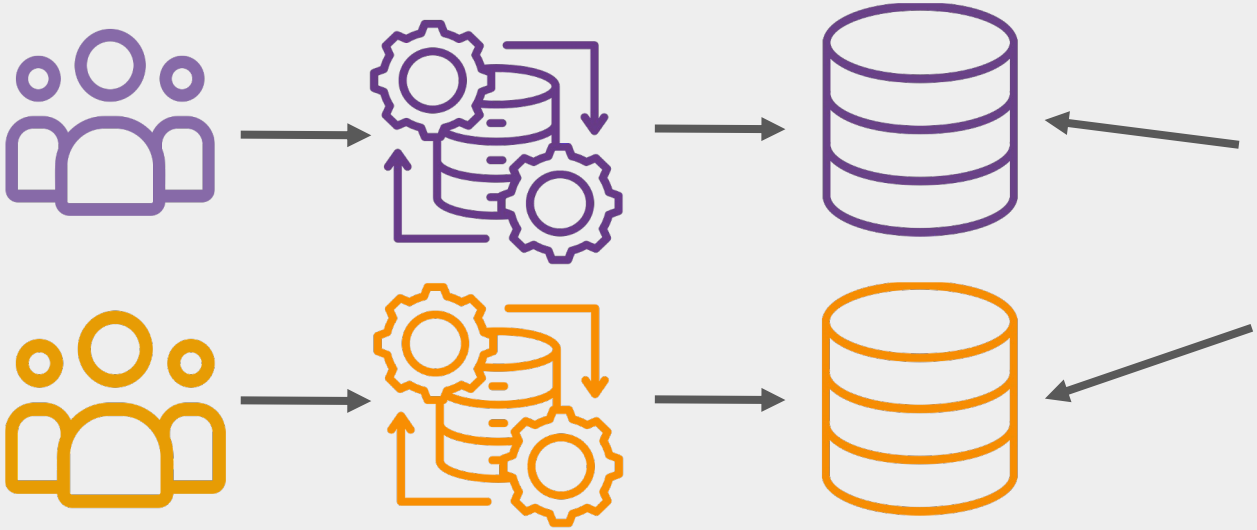
ПОСЛЕ

GET https://company1.mdaudit.ru/api/items

```
fun getItems() {
  …
  db.query(“select * from items”)
  …
}
```

Как хранить данные клиентов?

Как было раньше?



Данные **разных** клиентов хранятся **отдельно**

Вариант 1: Объединяем все данные в общие таблицы

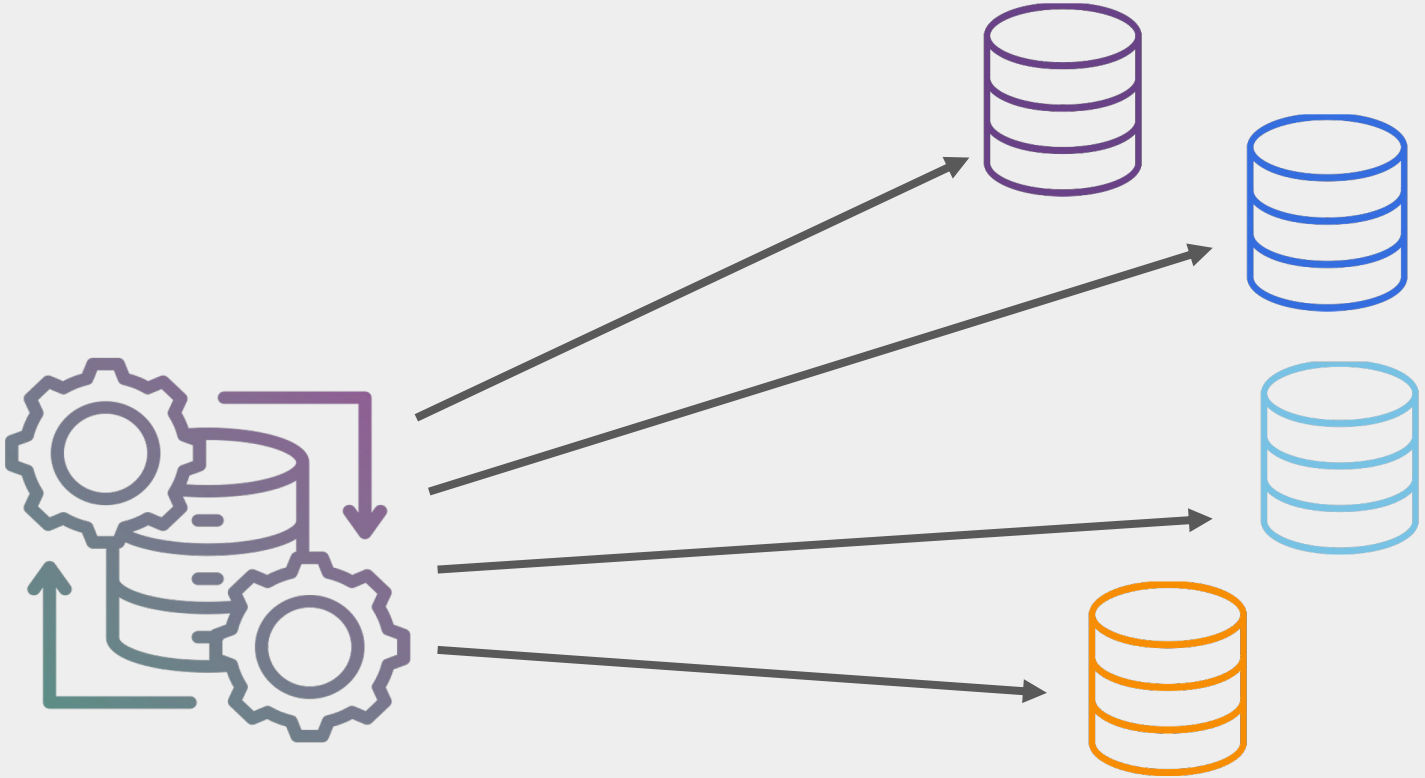
ID	Name	Tenant
1	Заявка 1	A
2	Заявка 2	A
3	Заявка 1	B

select * from items where tenant = ‘A’;

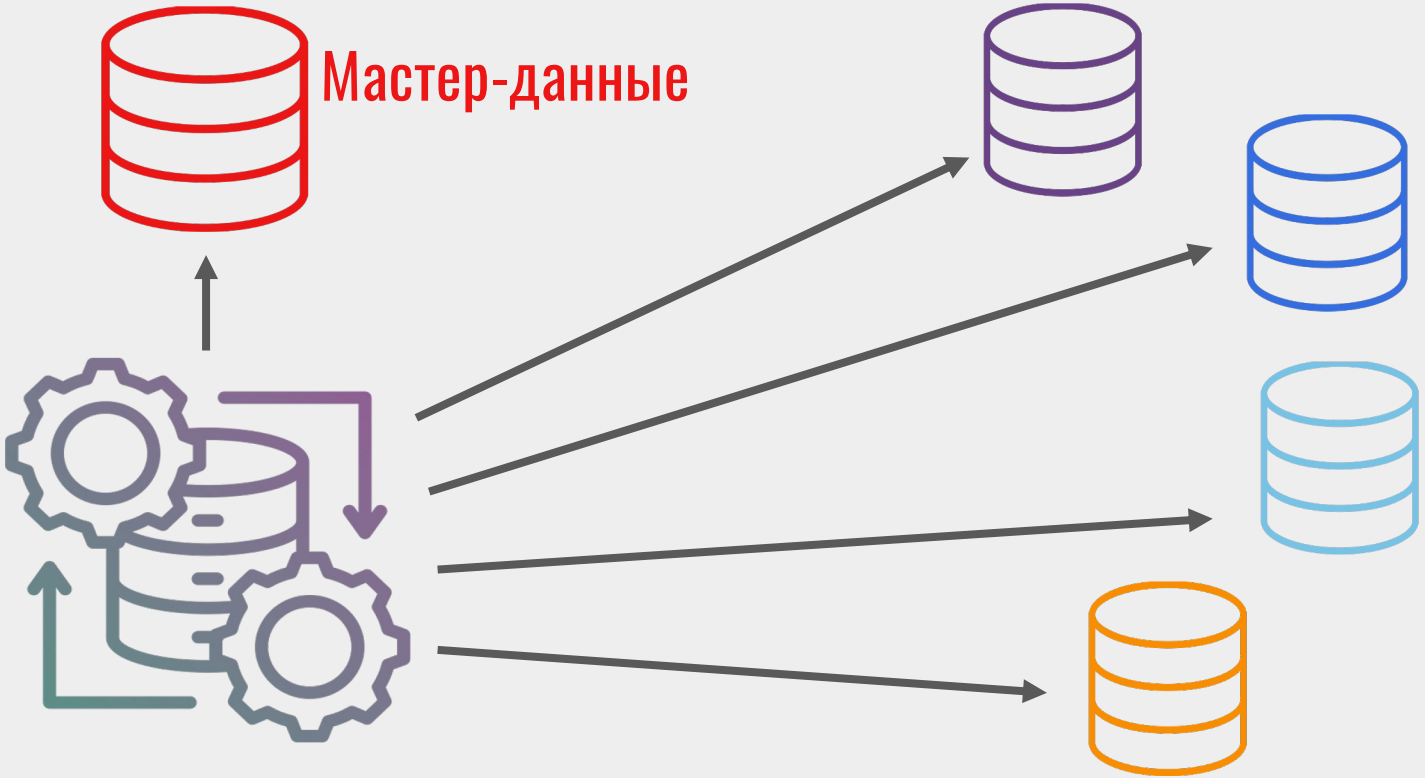
Проблемы

- Реализация требует значительных доработок
- Риск получить “чужие” данные при ошибке разработчика
- Большой объем данных в таблице

Вариант 2: Оставляем отдельные БД



Вариант 2: Оставляем отдельные БД



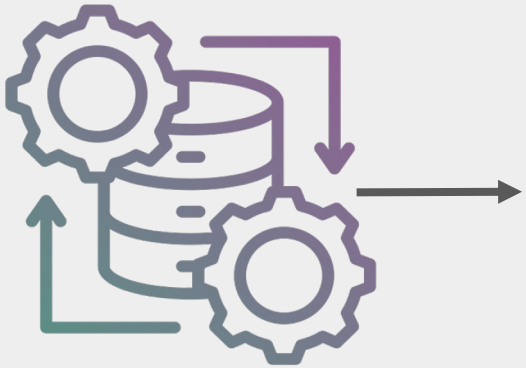
Преимущества

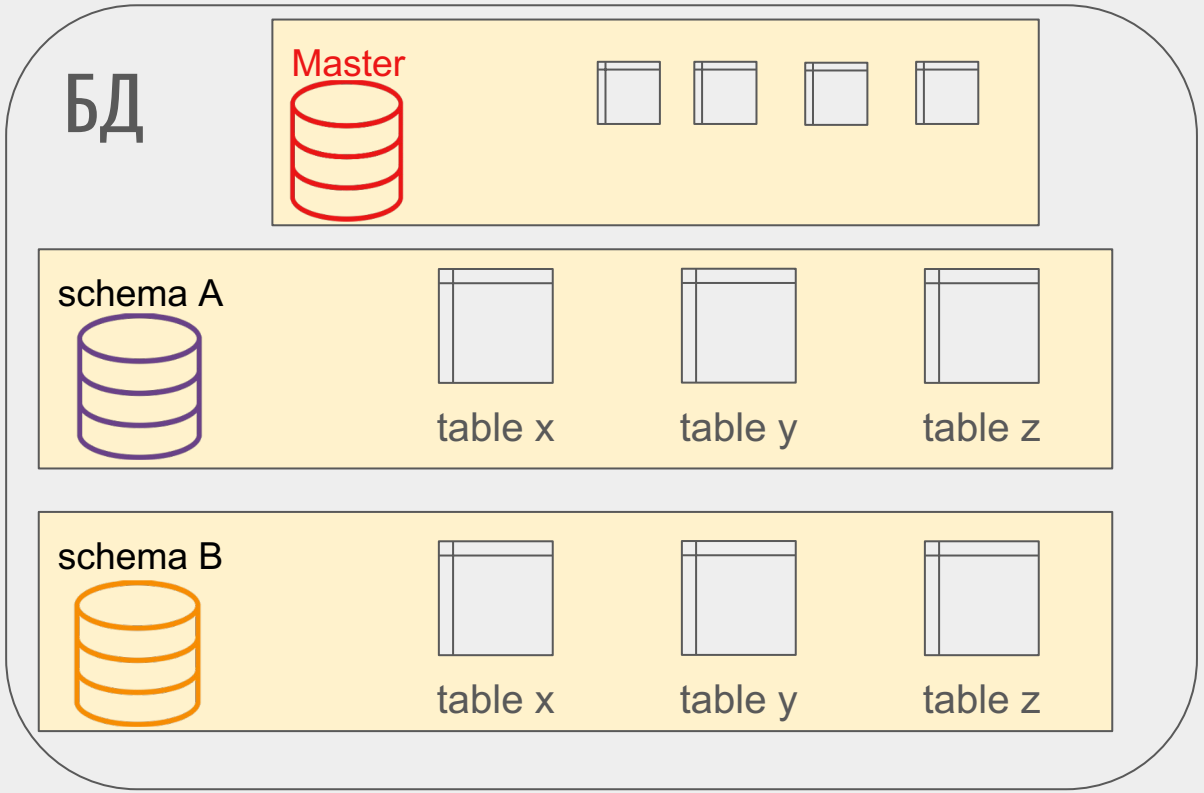
- Относительно просто реализовать
- Просто поддерживать - реализация скрыта за абстракцией
- Партиционирование по клиентам “из коробки”

Недостатки

- Требуется “мастер” БД
- Отсутствует возможность выполнения JOIN-ов и создания внешних ключей между БД
- Добавление нового клиента требует создания схемы

Вариант 3: Одна БД- много схем





Вариант 3: Одна БД- много схем

```
set search_path = schema_A;

select * from items;
```

Преимущества

- Относительно просто реализовать
- Просто поддерживать - реализация скрыта за абстракцией
- Партицирование по клиентам “из коробки”
- **Можно выполнять JOIN-ы и использовать внешние ключи между таблицами разных клиентов**

Создание схемы клиента

1. Создаем схему в БД
2. Запускаем скрипт миграции в новой схеме
3. Заполняем данными клиента
4. Создаем запись в мастер таблице

Итоги 5 лет в проде

Итоги 5 лет в проде

- Клиенты регистрируются сами

Итоги 5 лет в проде

- Клиенты регистрируются сами
- "Бесплатный" пробный период

Итоги 5 лет в проде

- Клиенты регистрируются сами
- "Бесплатный" пробный период
- **В 20 раз снизились затраты на инфраструктуру**

Итоги 5 лет в проде

- Клиенты регистрируются сами
- "Бесплатный" пробный период
- В 20 раз снизились затраты на инфраструктуру
- **Поддерживает 1 девопс**

Выводы

Выводы

- **Серебряной пули не существует**

Выводы

- Серебряной пули не существует
- **Лучшее - враг хорошего**

Выводы

- Серебряной пули не существует
- Лучшее - враг хорошего
- **Невозможно всё учесть заранее**

Спасибо за внимание

@TYURIN23