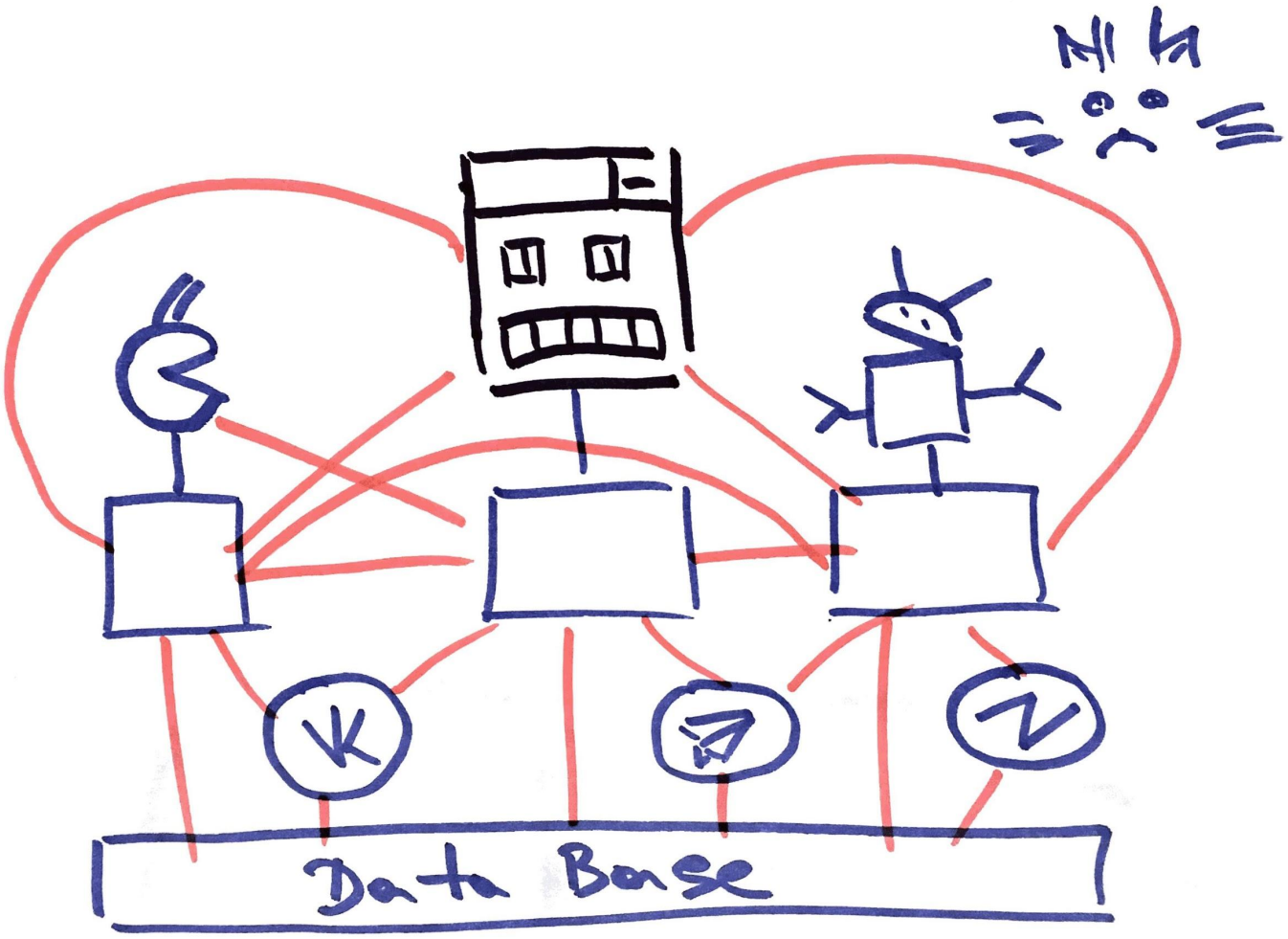
Database Delivery: The big problem

Роман Гордеев

Что будет?

1. Рассмотрим основные проблемы доставки БД
2. Изучим основные принципы и подходы управления схемой БД а также практики доставки БД
3. Продемонстрируем практический пример с использованием liquibase

Как и любая подобная история, эта история начинается с приложения...


Data Base

Какие вызовы?

1/ Множество различных окружений, различающихся по конфигурации, требованиям и данным

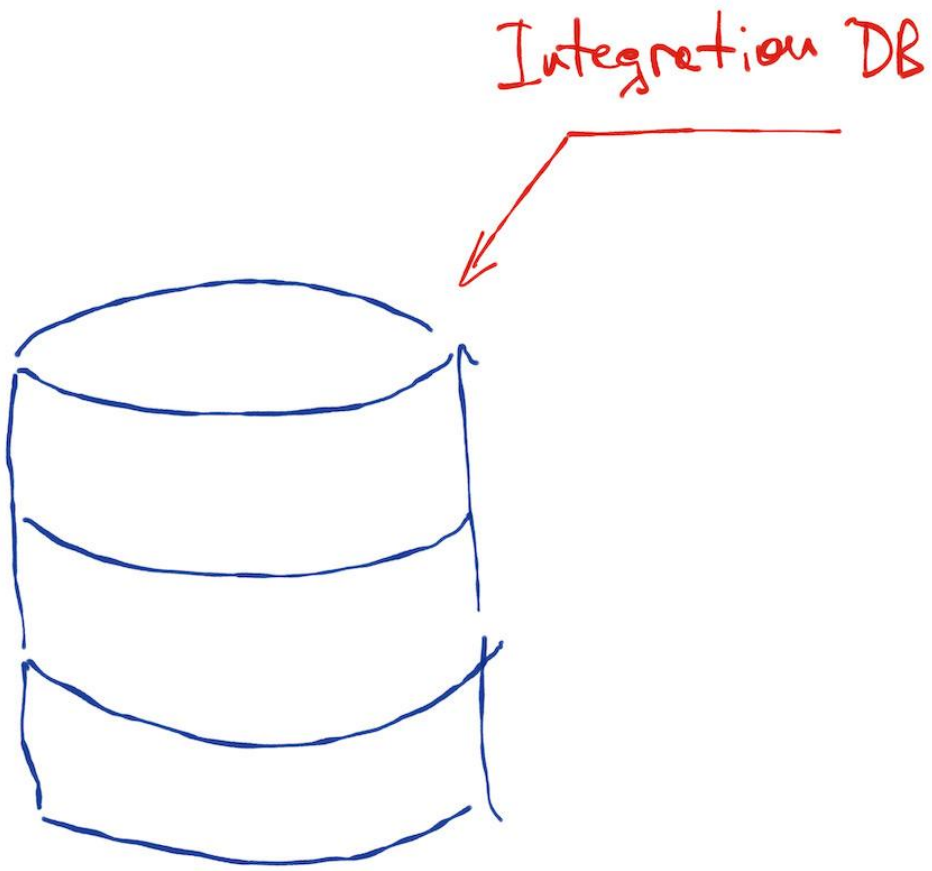
2/ Много участников, одновременно модифицирующих приложение и нижележащие данные и структуры

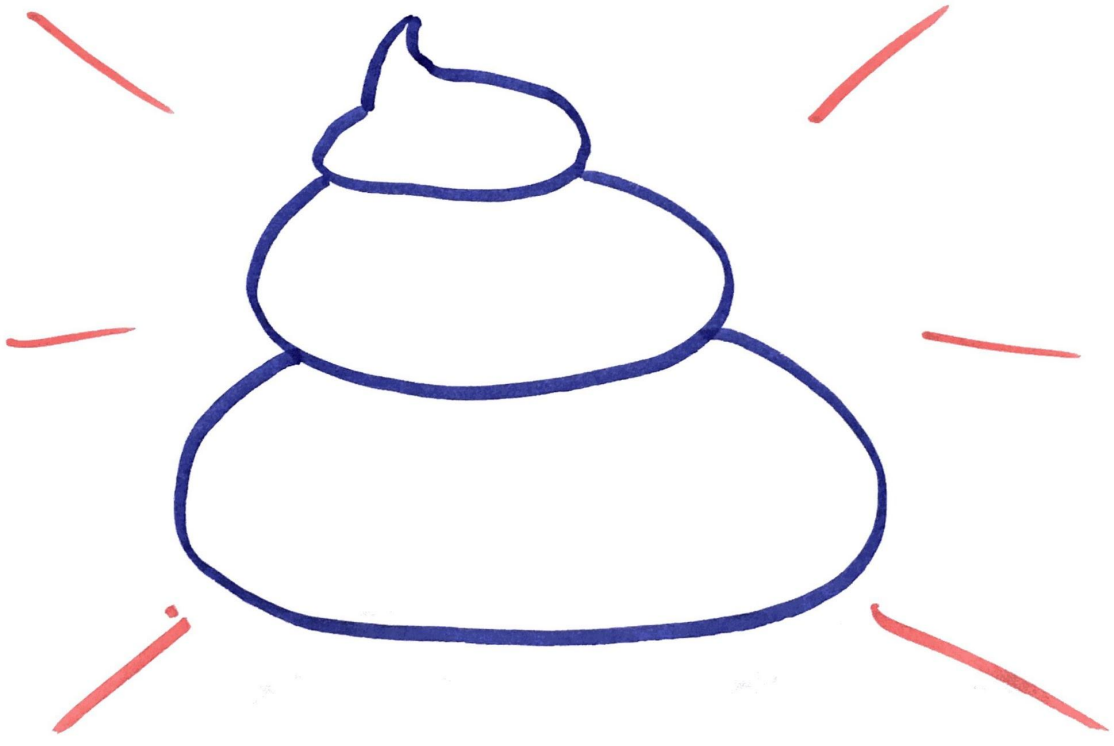
3/ Наличие большого (или не очень) количества продуктовых БД

Что мешает?

1/ Данные в БД

2/ Интеграция на уровне БД


Integration DB



3/ Несогласованность различных окружений

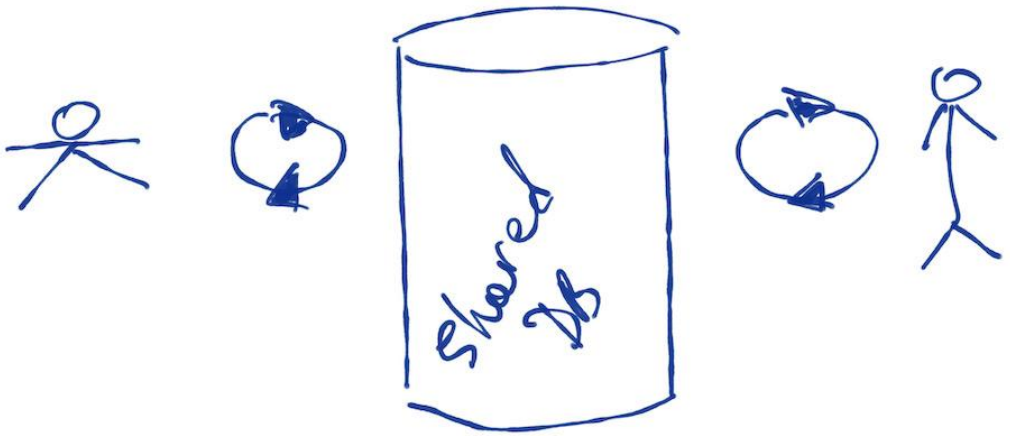
4/ Конфликты при взаимных модификациях схемы БД

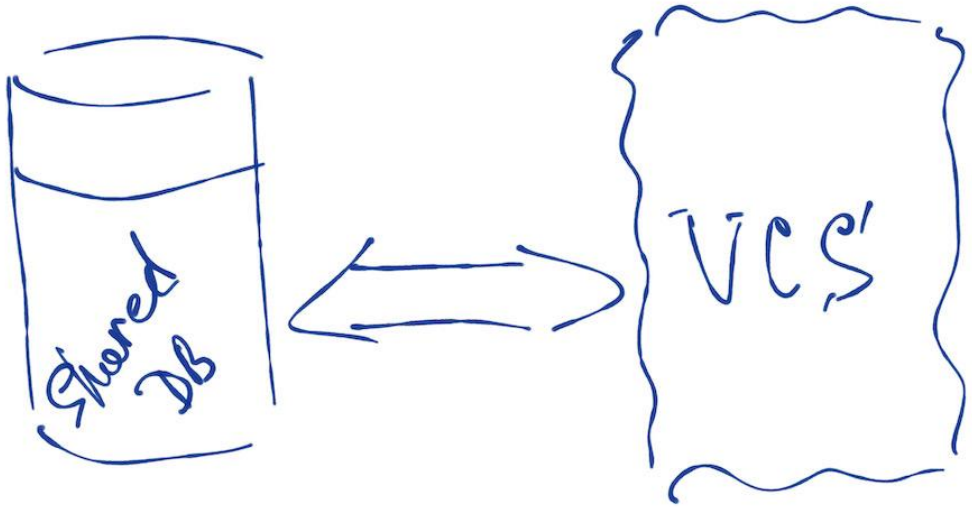
Что можно сделать?

Database as a code (схема, словари, тестовые данные)

Доставка изменений БД должна быть включена в процесс интеграционного тестирования

Каждому разработчику своя БД

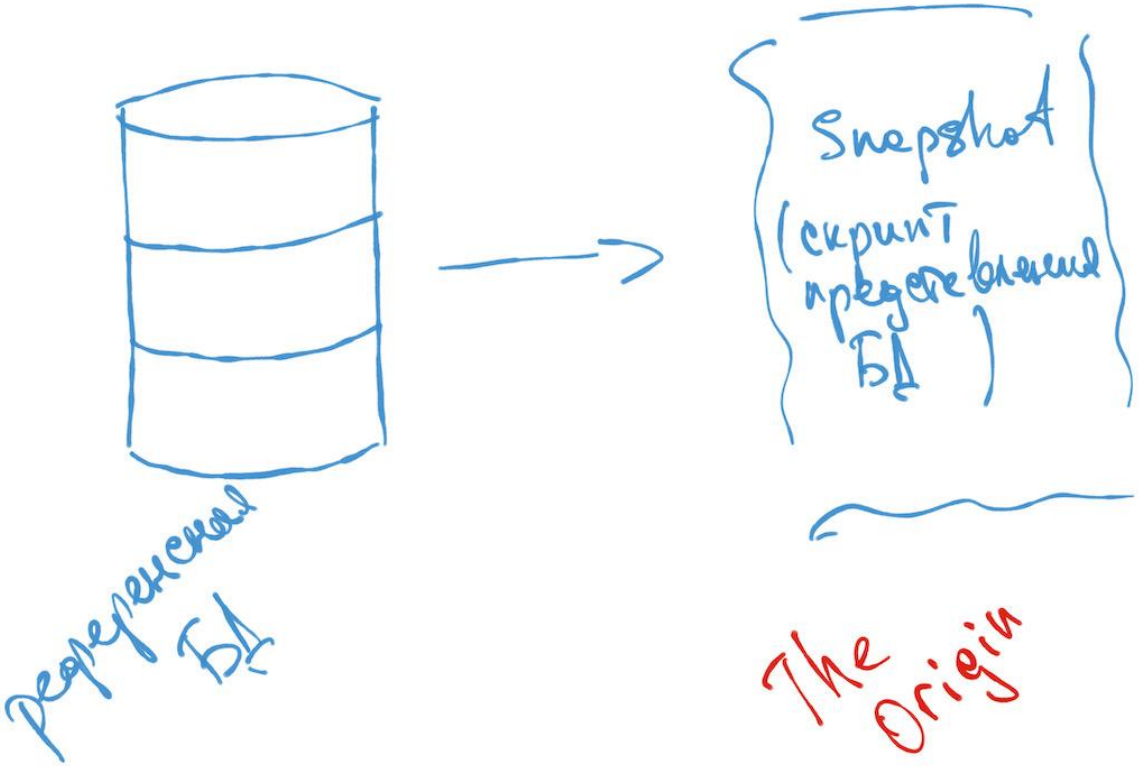

Shared
DB

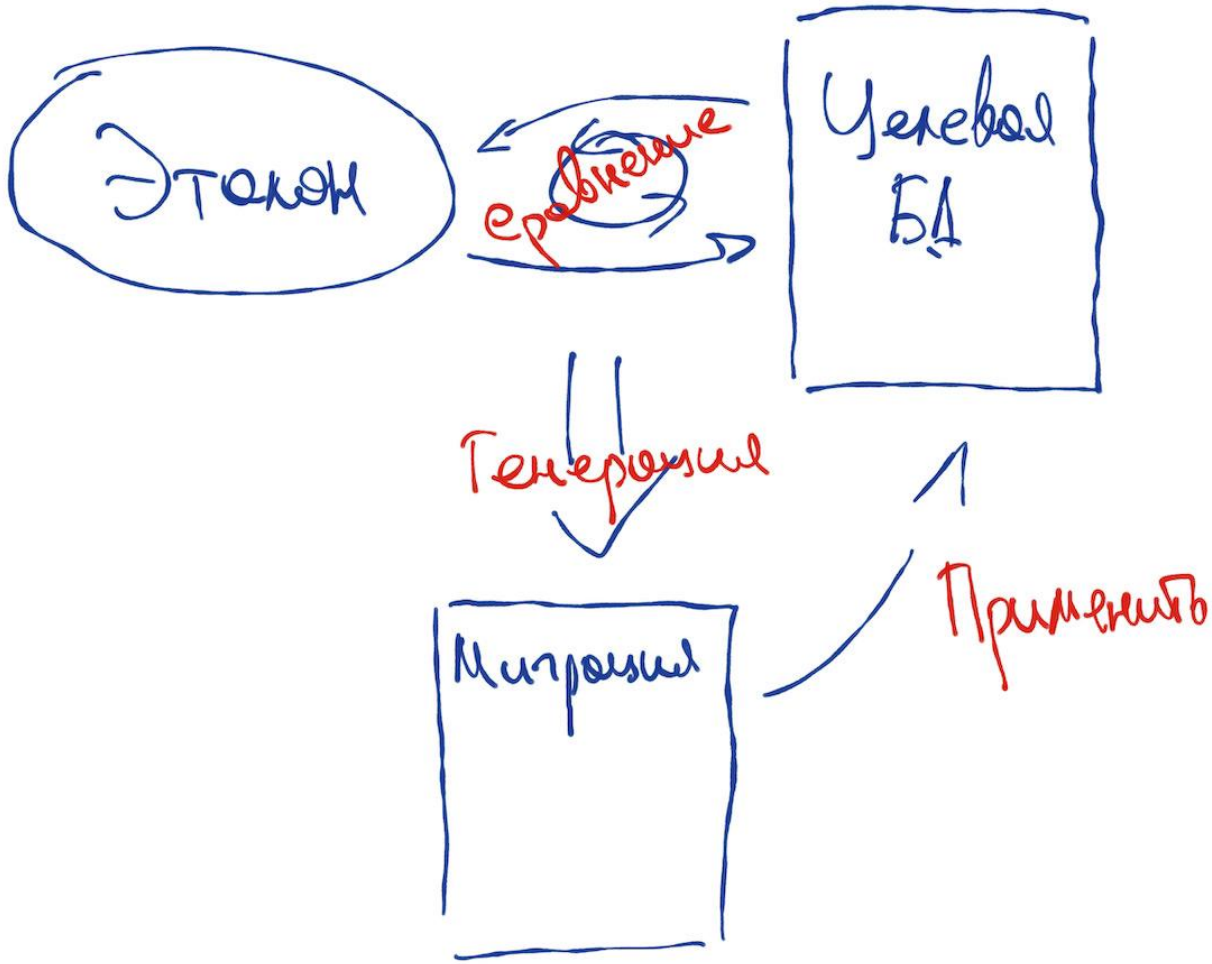

Stored
DB
VCS

Перманентно поддерживать согласованность модели приложения и схемы БД

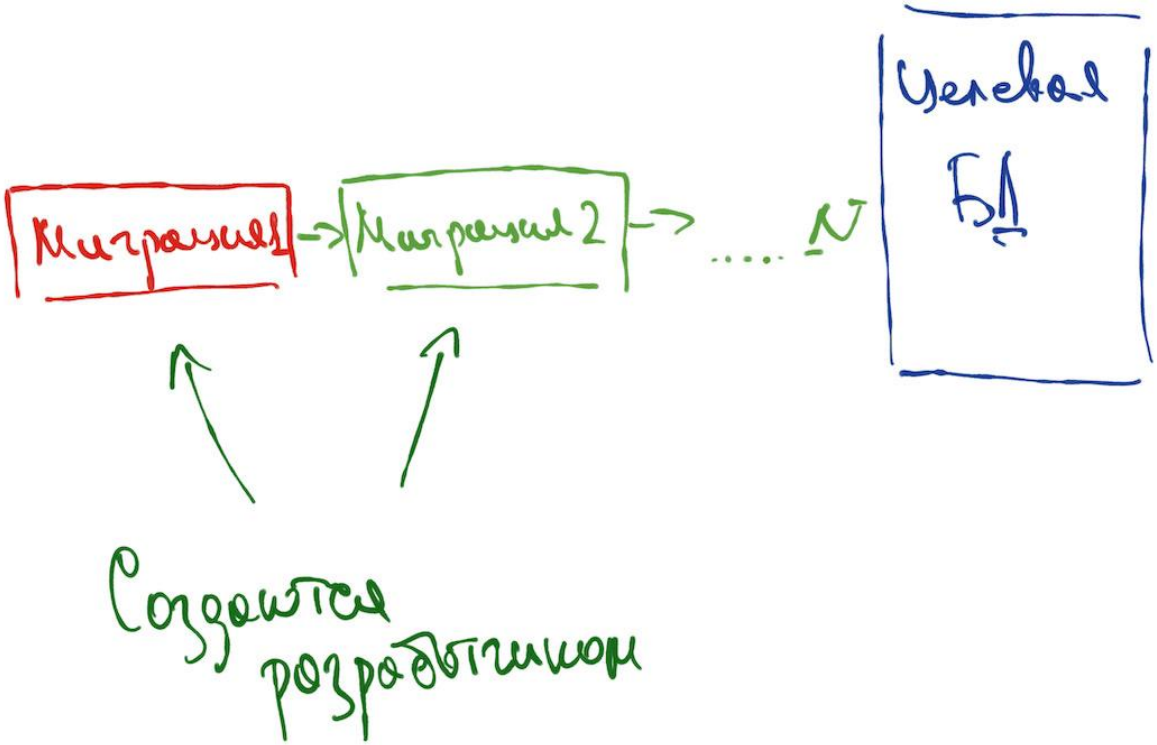
Подходы к управлению доставкой изменений БД

На основе эталона (state-based)


Snapshot
(скрипт предоставления БД)
реферансная БД
The Origin


Эталон
сравнение
Целевая БД
Генерация
Миграции
Применить

На основе последовательного применения инкрементальных изменений (Migration-Based)


Миграция 1
Миграция 2
N
Целевая БД
Создаются разработчиком

Практический пример

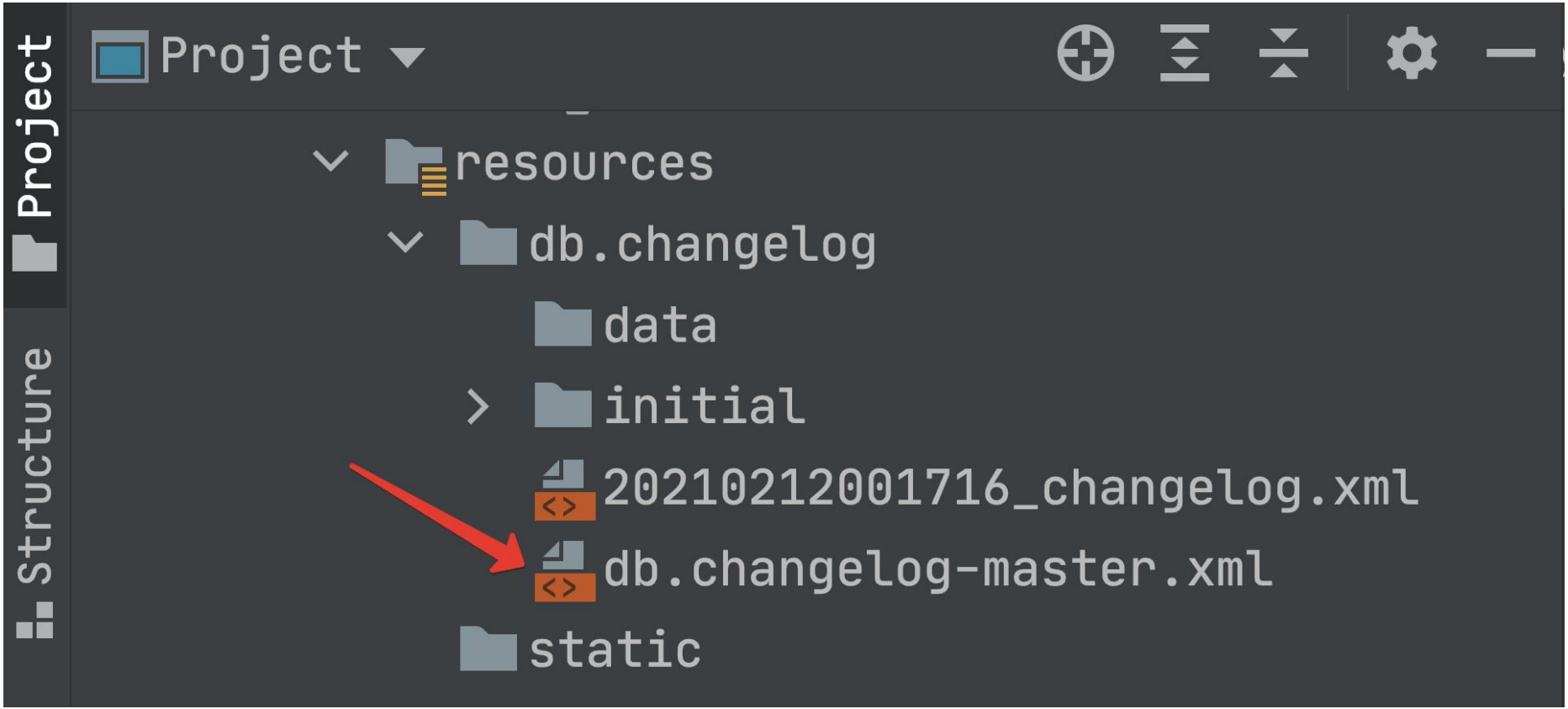
Легенда

1. Переписываем легаси приложение с БД
2. У нового приложения есть изменения в схеме БД
3. В новом приложении хотим реализовать database delivery на основе миграций

Шаг 1. Синхронизируем схему легаси БД и снепшота нового приложения

Шаг 2. Формируем начальную миграцию, полностью описывающую актуальную схему БД

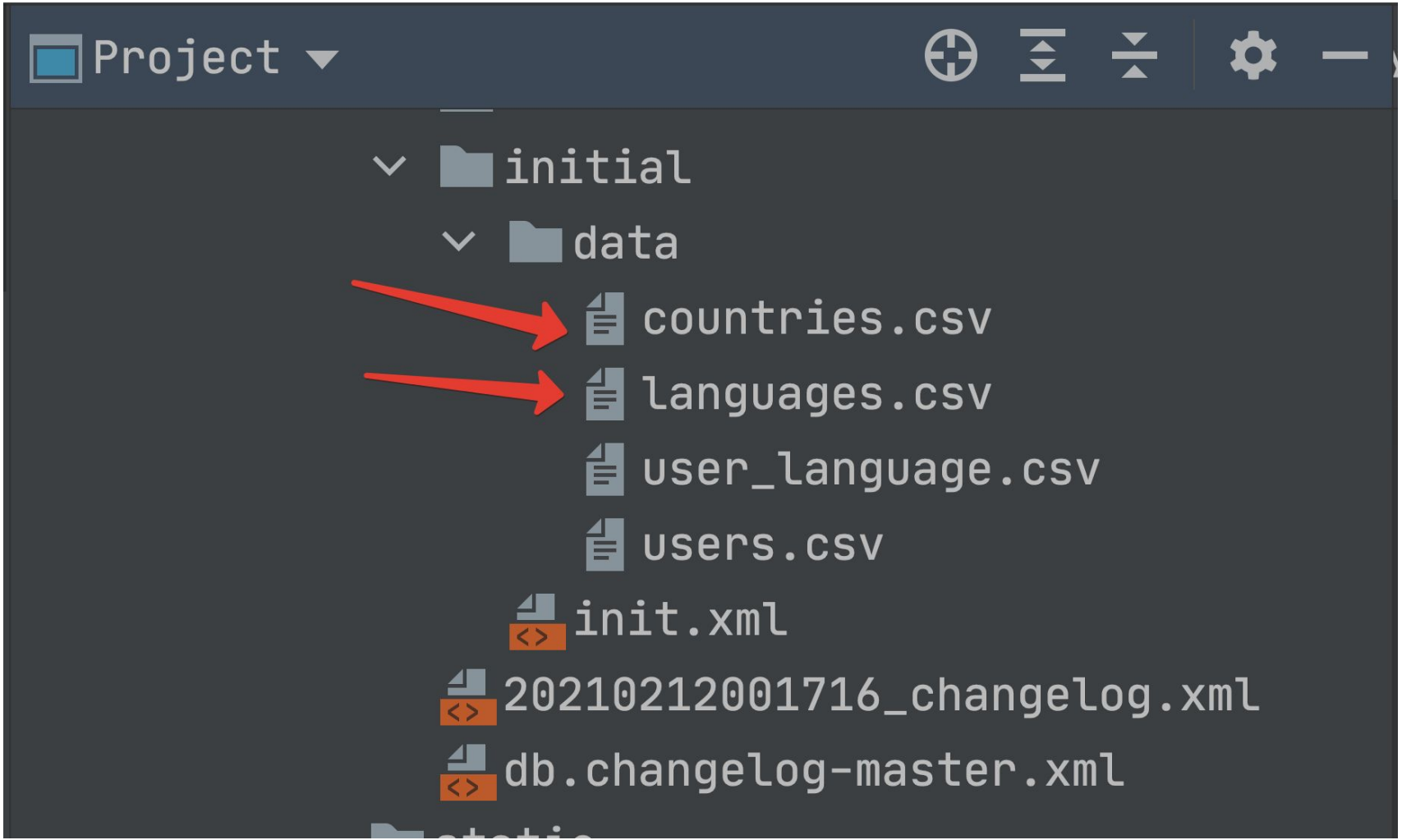
```
./gradlew liquibaseGenerateChangelog
```


Project
Project
Structure
resources
db.changelog
data
initial
20210212001716_changelog.xml
db.changelog-master.xml
static

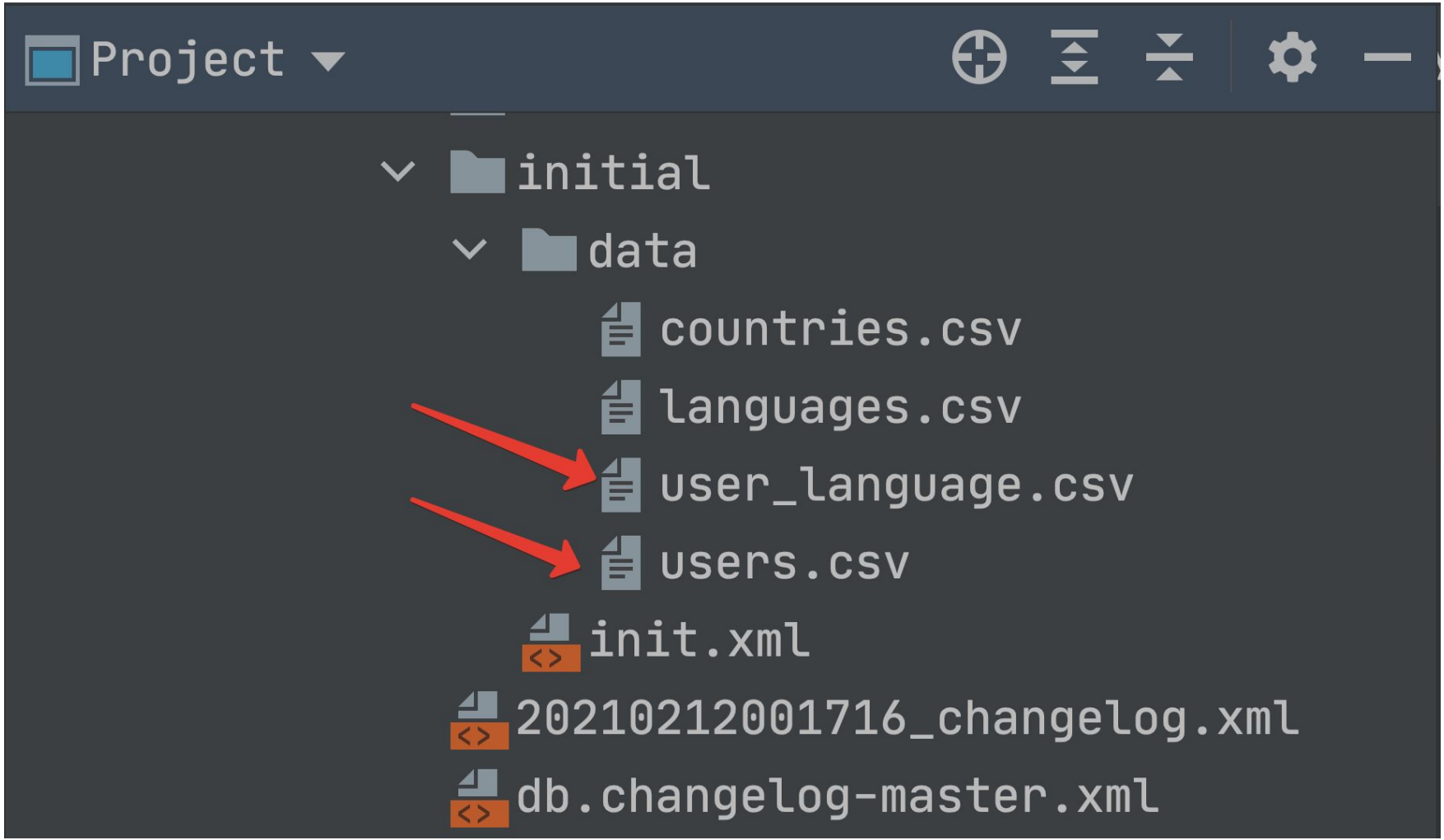
Шаг 3. Инициализируем схему нового снепшота и проверяем отсутствие расхождений с целевой БД

Шаг 4. Настраиваем интеграционные тесты

Шаг 5. Создаем миграции словарей


Project
initial
data
countries.csv
languages.csv
user_language.csv
users.csv
init.xml
20210212001716_changelog.xml
db.changelog-master.xml

Шаг 6. Создаем миграции данных для тестовых и локальных сред


Project
initial
data
countries.csv
languages.csv
user_language.csv
users.csv
init.xml
20210212001716_changelog.xml
db.changelog-master.xml

Шаг 7. Рефакторинг

```
./gradlew liquibaseDiffChangeLog -PrunList=diffLog
```

References

- Vladimir Khorikov. Database versioning best practices https://enterprisecraftsmanship.com/posts/database-versioning-best-practices/
- Scott W. Ambler, Pramod J. Sadalage. Refactoring Databases: Evolutionary Database Design https://www.ozon.ru/product/refaktoring-baz-dannyh-evolyutsionnoe-proektirovanie-162984061
- https://docs.liquibase.com/home.html

That all folks!

https://github.com/rtk-it/migrations