БАНК ЦЕНТР-ИНВЕСТ

Flow 2022

Дружба – это чудо

Contract First для аналитиков и разработчиков

Flow 2022

01

ПРОБЛЕМЫ

Какие проблемы возникают при выполнении задач интеграции

02

РЕШЕНИЕ

Применение подхода Contract First для решения обозначенных проблем

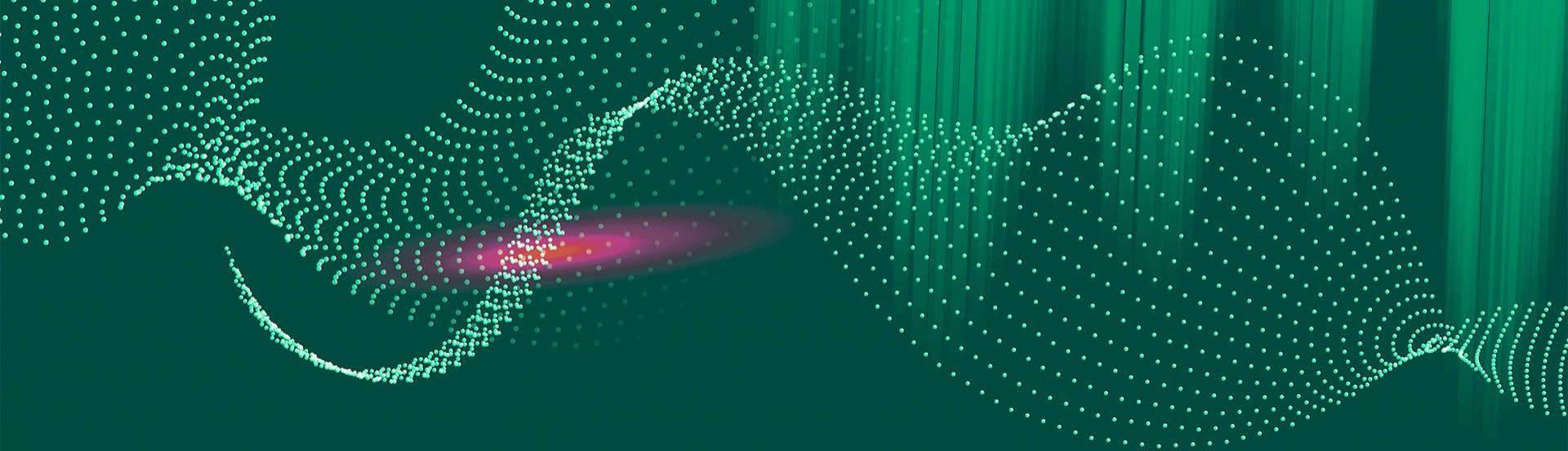
03

ПРАКТИКА

Практическая реализация озвученных решений

РОМАН ЛАПТЕВ

- ПАО КБ «Центр-инвест»
- Команда DevOps и интеграций
- Техлид
- Автоматизирую всю «ручную работу»



01 ПРОБЛЕМЫ

Никогда такого не было –
и вот опять

ПОРОЧНЫЙ КРУГ БЫТИЯ

ПОЛУЧЕНИЕ ЗАДАЧИ

Аналитики долго и кропотливо составляют ТЗ в текством виде с описанием интеграции приложений

ПОЛУЧЕНИЕ ТЗ

Разработчики читают ТЗ и уточняют недостающую информацию в части интеграции приложений

СОГЛАСОВАНИЕ ТЗ

Разработчики задают эти вопросы аналитикам, получают обновлённое ТЗ и по новой

НУ ХУЖЕ УЖЕ НЕ БУДЕТ ™

БУДЕТ!
ПРОБЛЕМ ХВАТИТ ВСЕМ!

“Щас, ТЗ на 100 страниц проверим”

— Аналитики

“Щас, с другим приложением отладимся”

— Разработчики

“Щас, корректность всех полей проверю”

— Тестировщики

“Ай, обратную совместимость поломали!”

— Команда

РАБОТА С ТЗ В ТЕКСТОВОМ ВИДЕ

ТЗ составляется в текстовом формате, который никак не валидирует описываемую интеграцию

- Необходимо вручную описать все взаимодействия и представления данных
- Проверка корректности производится вручную
- Долго пишется, сложно, долго и неохотно пересогласовывается
- Отсутствуют вменяеме подсказки, анализ написанного и прочие полезные вещей из мира разработчиков

НЕОБХОДИМОСТЬ ОТЛАДКИ

Интегрируемые приложения часто реализуются разными командами и им приходиться отлаживать взаимодействие

- Может затянуться при большом количестве вызовов
- Необходимо вручную сверять с ТЗ реализацию вызовов и представления данных в коде
- Если интеграция с приложением другой команды – приходиться много коммуницировать касательно корректности приёма/отправки запросов

НЕОБХОДИМОСТЬ ПРОВЕРКИ РЕАЛИЗАЦИИ

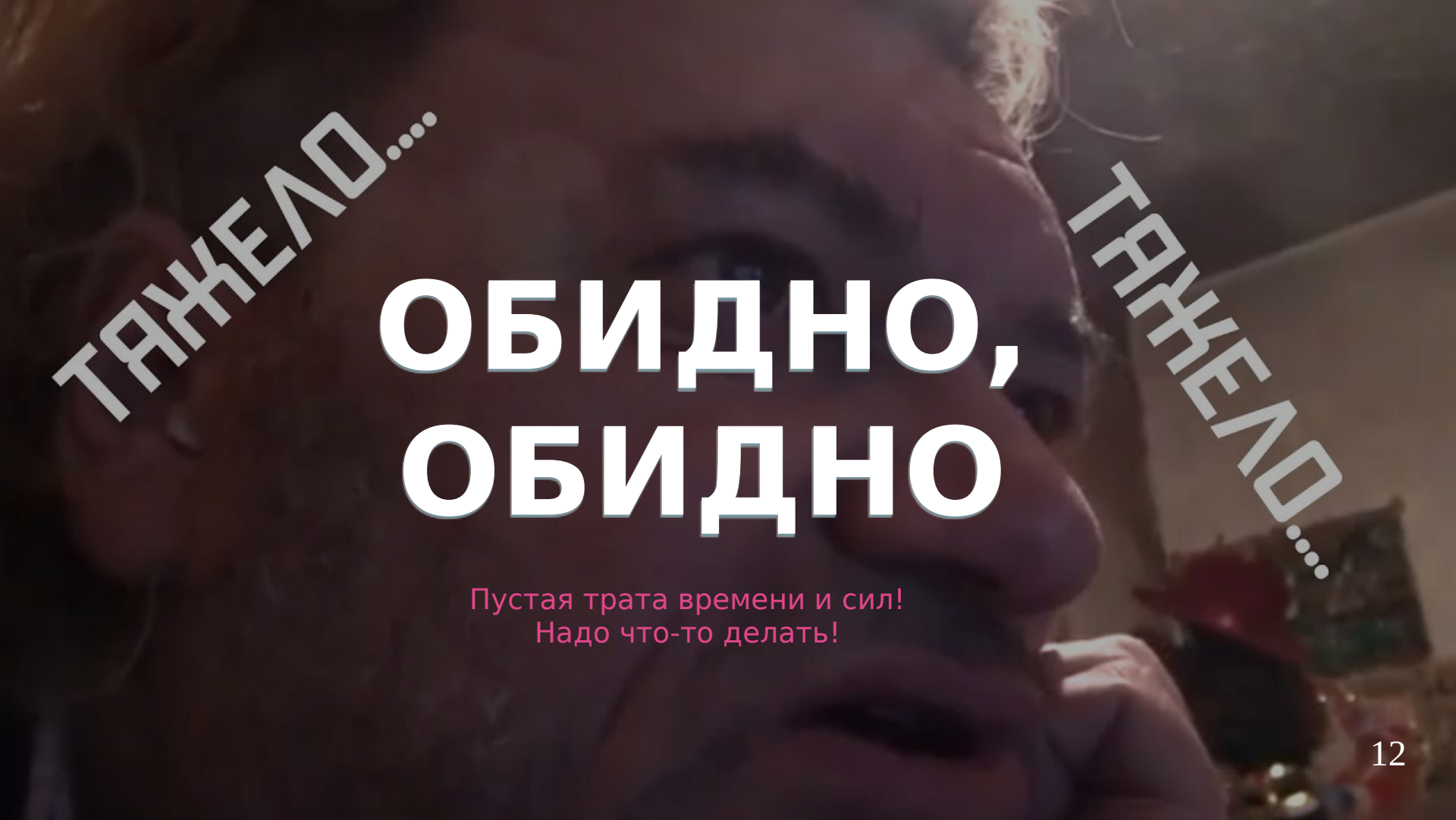
Даже отлаженное взаимодействие не гарантирует наличия всех необходимых вызовов, полей и ответов

- Такие проблемы могут всплыть не сразу
- Разработчики могут забыть реализовать редкое nullable поле или редкий запрос
- Используемые типы данных могут различаться в интегрируемых приложениях, что может привести к неожиданным ошибкам

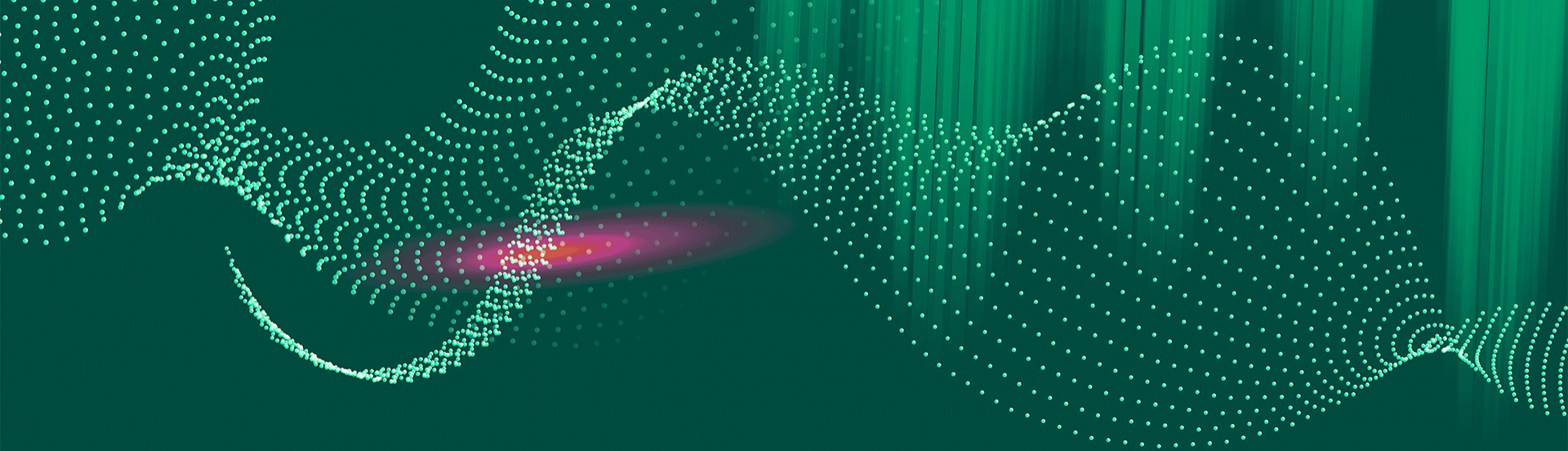
МОЖНО ПОЛОМАТЬ ОБРАТНУЮ СОВМЕСТИМОСТЬ

Может возникнуть ситуация, когда изменения в интеграции идут вразрез с предыдущими версиями и уже реализованные вызовы завершаются ошибками

- Критическая ошибка, которую можно заметить поздно
- Чтобы не возникала – нужно очень долго валидировать ТЗ вручную и проверять предыдущие версии
- Может возникнуть даже не из-за ТЗ, а из-за особенностей реализации интегрируемых приложений



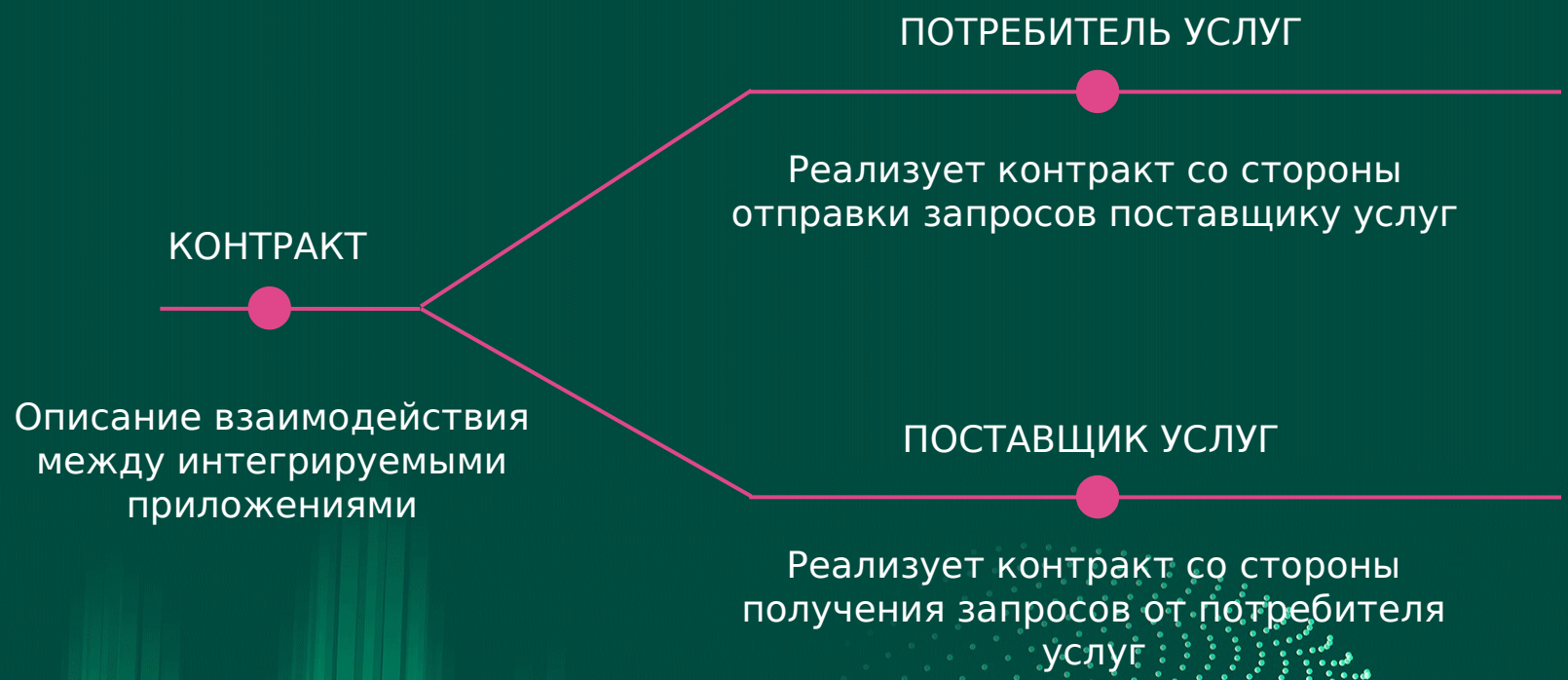
Пустая трата времени и сил!
Надо что-то делать!



02 РЕШЕНИЕ

Contract First – спаситель наш

ДА КТО ТАКОЙ ЭТОТ ВАШ...



“Так получается контракт – это ТЗ наше! Так и где обещанное решение всех проблем?!”

— average недослушавший меня собеседник

ОДНО МАЛЕНЬКОЕ ИЗМЕНЕНИЕ

Замена технической части ТЗ на описание с помощью языка контракта

ТЕКСТОВОЕ ТЗ

Слабо регламентированное описание системы, предназначенное для чтения человеком

ЯЗЫК КОНТРАКТА

Строго регламентированное описание системы, с которым может работать человек и компьютер

УДОБСТВО ОПИСАНИЯ ИНТЕГРАЦИИ

Описание интеграции на языке контракта позволяет воспользоваться преимуществами языков программирования

- Помощь в написании – возможность по нажатию кнопки создать заготовку под запрос или данные
- Синтаксический анализ и валидация – ещё на этапе написания описывается что с контрактом не так
- Переиспользование данных – можно единожды описать схему данных и переиспользовать её
- Можно сгенерировать красивый документ, который не надо форматировать

ГЕНЕРАЦИЯ КОДА ИТЕГРАЦИИ ПРИЛОЖЕНИЙ

За счёт строго регламентированной структуры, по языку контракта можно автоматически генерировать код для интегрируемых приложений

- Интеграцию не нужно реализовывать – код для неё уже сгенерирован
- Сгенерированный по одной схеме код для разных приложений не нужно согласовывать между собой
- Генератор ничего не забывает реализовать и строго следует контракту

ОТЛАДКА ДО СТАРТА РАЗРАБОТКИ

К процессу генерации кода можно подвязать проверки критичных для интеграции вещей – обратную совместимость, соответствие типов данных и т.д.

- Позволяет переместить проверку контракта на более ранний этап – перед стартом разработки
- Позволет аналитикам сразу увидеть недочёты в описании интеграции, из-за чего правки вносятся быстрее

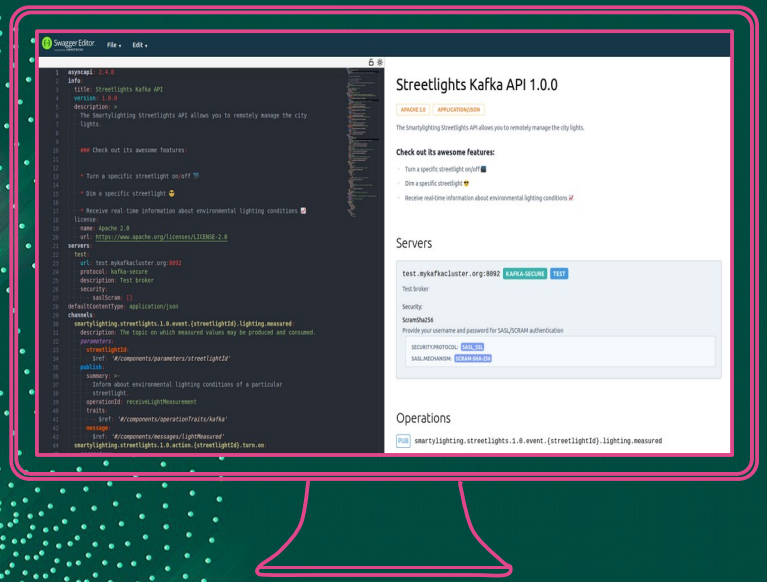
03 ПРАКТИКА

Мы не бездельничаем – оно автоматизировано

ЯЗЫКИ КОНТРАКТА

	ГДЕ ПРИМЕНЯЕТСЯ	ОСНОВНОЕ НАЗНАЧЕНИЕ	КОММЕНТАРИЙ (Если надо выбирать)
OpenAPI	RESTful (req-res)	сервис-сервис, клиент-сервис	Любимец публики и популярное решение
protobuf	GRPC (req-res)	сервис-сервис, клиент-сервис (мобилки)	Когда HTTP/1.1 мало. Поддерживает только ContractFirst
WSDL	SOAP (req-res)	сервис-сервис	Можете найти его в легаси и при работе с госухой
GraphQL Schema	GraphQL (req-res)	клиент-сервис	Популярная альтернатива REST для фронта/мобилок
AsyncAPI	AMQP, Kafka, WebSocket (pub-sub)	сервис-сервис	Молодое решение, превносящее ContractFirst в мир брокеров сообщений и pub-sub модели

СОЗДАНИЕ ФАЙЛА КОНТРАКТА



- OpenAPI – Swagger Editor
- WSDL – Altova, Oxygen
- GraphQL Schema - GraphQL Voyager
- gRPC – IDE (а кому щас легко?)

ГЕНЕРАТОРЫ КОДА

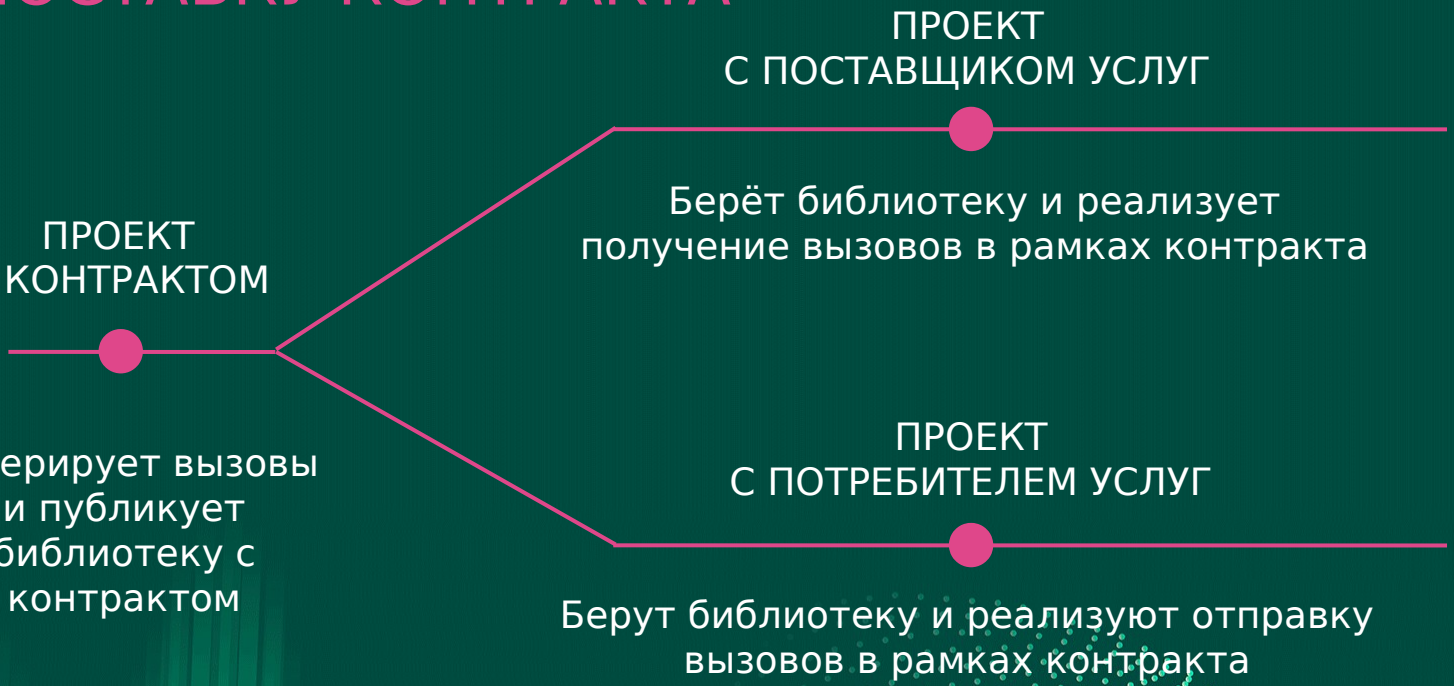
	ГЕНЕРАТОР	ЯЗЫКИ
OpenAPI	openapi-generator-gradle-plugin	Java, Kotlin и более 20 других
protobuf	protobuf-gradle-plugin	Java, Kotlin и другие, где доступен gRPC
WSDL	wsdl2java	Java
GraphQL Schema	graphql-java-codegen	Java

ТЕСТИРУЕМ КОНТРАКТ

Для тестирования контракта необходимо задействовать сгенерированный в сэмулированном межсервисном взаимодействии и проверить корректность всех сценариев взаимодействия приложений

- Сгенерировали код
- Замокали клиент с ответами
- Замокали сервер с вызовами
- Прогнали вызовы
- Проверяем корректность результата всех вызовов
- Если тесты прошли – обратная совместимость контракта не нарушена

АВТОМАТИЗИРУЕМ ПОСТАВКУ КОНТРАКТА



ПОЧЕМУ НЕ МОНОРЕПОЗИТОРИЙ

Монорепозиторий позволяет запутать код приложений между собой, чего не возникает при разделении проектов

- Отделение контракта позволяет продумывать его не завязываясь на реализацию приложений
- При отделении контракта аналитикам и тестировщикам можно предоставить доступ к проекту с контрактом, не предоставляя доступ к коду
- Все проблемы можно решить, но их решение сложнее, чем просто разделение проектов

ПРИМЕРЧИКИИИ

На словах хорошо – теперь в коде

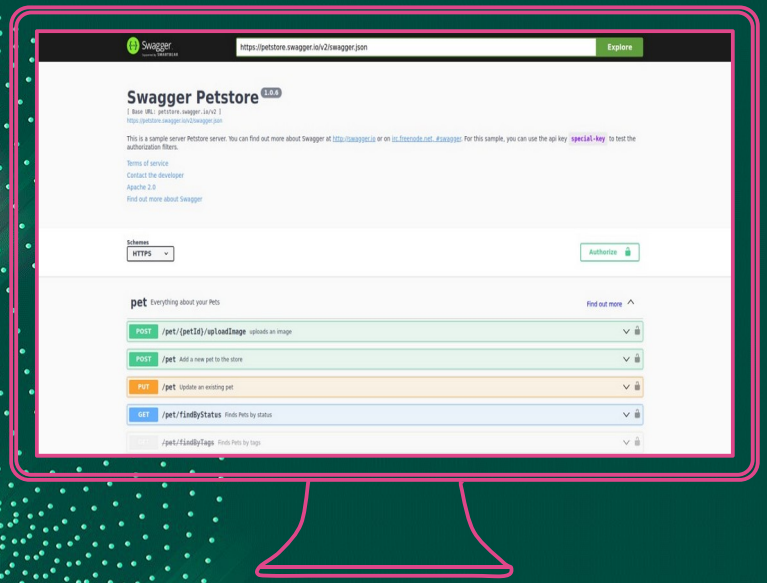
БЫЛО

- Тяжело вручную всё составлять и проверять
- Постоянные подводные камни, нарушающие процесс
- Трата времени на согласование и решение ошибок

СТАЛО

- С подходящими инструментами всё делается легче
- Процесс идёт куда прямолинейнее и с меньшим числом ошибок
- Всё согласовано автоматически

И ЭТО НЕ ВСЁ!



- Автоматическая документация
- Инструментарий для тестировщиков
- Моки клиентов и серверов
- Генерация схемы для уже существующих приложений

ВЫВОДЫ

ВЫИГРЫШ

- Упрощение составления контракта
- Увеличение скорости разработки
- Уменьшение числа ошибок
- Уменьшение объёма работ

НЮАНСЫ

- Генераторы кода имеют свои ограничения
- Аналитикам нужно уметь писать на языке контракта
- Нужно настраивать генерацию кода и CI/CD

МАТЕРИАЛЫ ДОКЛАДА

1. Описание gRPC, которе меня вдохновило - https://habr.com/ru/company/yandex/blog/484068/
2. Прочёл перед внедрением - https://habr.com/ru/post/483206/
3. Демо приложения - https://github.com/CunningBird/contract-first-applications
   - Генератор для OpenAPI - https://github.com/CunningBird/contract-first-openapi
   - Генератор для Protobuf - https://github.com/CunningBird/contract-first-grpc
   - Генератор для SOAP - https://github.com/CunningBird/contract-first-wsdl
   - Генератор для GraphQL - https://github.com/CunningBird/contract-first-graphqls

ДЕЛАЙТЕ ХОРОШО – ПЛОХО НЕ ДЕЛАЙТЕ

А если делаете плохо –
перестаньте и делайте хорошо