Flow

2025 Autumn

Карта технологий интеграции

Юрий Куприянов
ТГ-канал «Системный сдвиг»

Немного обо мне

Аналитик, преподаватель, консультант

>25 лет опыта:
от программиста до
архитектора CTO/CPO

>10 предметных областей

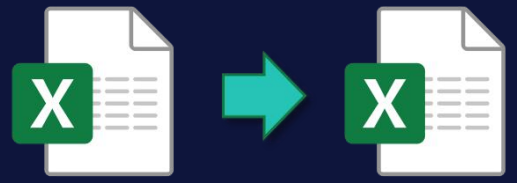
>10 лет учу аналитиков

Веду канал «Системный сдвиг»: t.me/systemswing

Веду курсы по разработке требований и проектированию интеграций в школе Systems.Education

Что я знаю про интеграции

2001: Преобразование файлов Excel



2006-2022: Общая БД, long polling, ETL, ESB, внутренние и открытые API, EDA, микросервисы и т.д. и т.п.

2022: Курс по основам проектирования интеграций

Хочу всё знать об интеграциях!

2025: Карта технологий интеграции

Проблема с классификациями

Виды интеграций:

1) Горизонтальные
2) Вертикальные
3) Точка-точка
4) Внешние API
5) Интеграция данных
6) Интеграция бизнес-процессов
7) Событийная интеграция
8) Корпоративная шина
9) Интеграция в стиле RPC
10) Webhooks

Животные делятся на:

а) принадлежащих Императору

б) набальзамированных

в) приручённых

г) молочных поросят

д) сирен

е) сказочных

ж) отдельных собак

з) включённых в эту классификацию

и) бегающих как сумасшедшие

к) бесчисленных

л) нарисованных тончайшей кистью из верблюжьей шерсти

м) прочих

н) разбивших цветочную вазу

о) похожих издали на мух

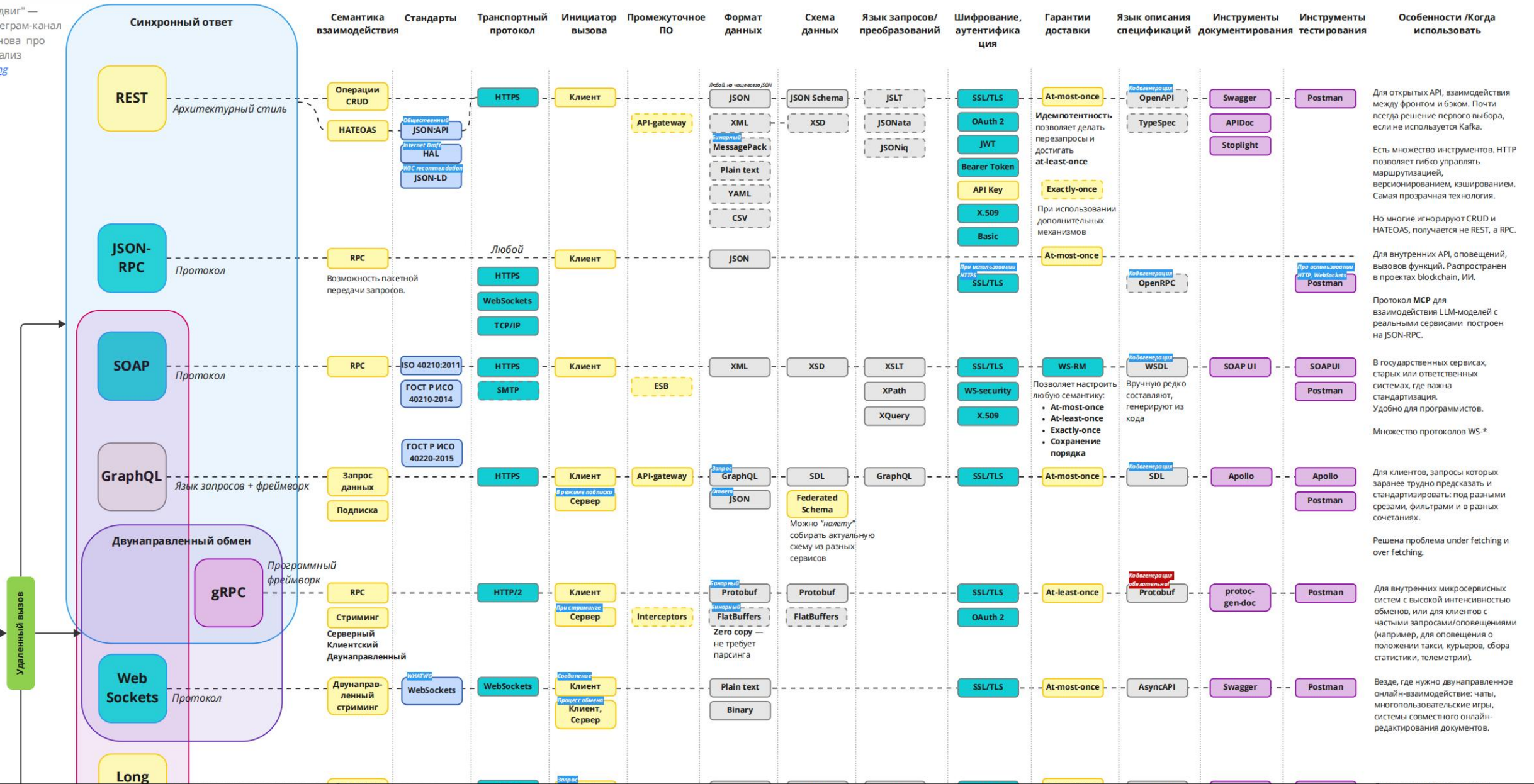
Хорхе Луис Борхес. «Аналитический язык Джона Уилкинсона»

Паттерны и технологии интеграции

Версия 1.1.0

System Swing

"Системный Сдвиг" — авторский телеграм-канал Юрия Куприянова про системный анализ
t.me/systemswing



Аспекты интеграций

- Верхнеуровневый шаблон
- Синхронность / асинхронность
- Семантика запроса
- Протоколы
- Форматы и схемы данных
- Инструменты и языки преобразования данных
- Оповещение клиентов об ошибках
- Безопасность
- Производительность, надежность, гарантии доставки
- Языки и инструменты спецификации

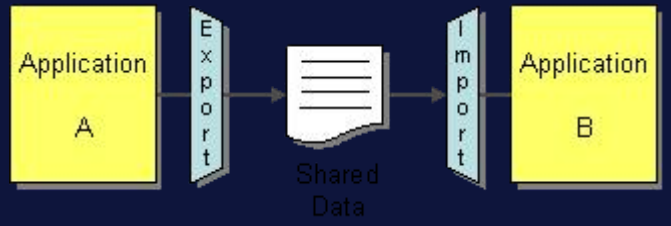
Зачем нужна классификация?

- Что нужно знать про интеграции? (оглавление)
- Что не забыть при выборе и внедрении технологии?
- Как выбрать технологию под задачу и ограничения?
- Как сравнить две технологии?
- Какие технологии используются в нашем проекте / компании в каких случаях, и что на радаре?

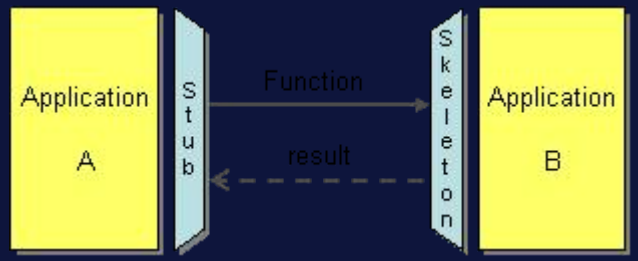
Основные технологии интеграции

SOAP — протокол обмена сообщениями
REST — архитектурный стиль
GraphQL — язык запросов
gRPC — программный фреймворк
WebSockets — коммуникационный протокол
Web Hook, Long Polling — приемы применения HTTP
SSE — часть стандарта HTML
MQTT — коммуникационный протокол
RabbitMQ — один из продуктов, реализующих протокол AMQP
Kafka — программный продукт
ESB — архитектурный паттерн и класс программных продуктов
ETL — коммуникационный паттерн и класс программных продуктов

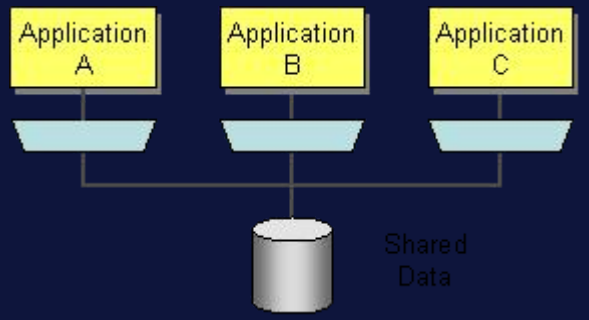
Верхнеуровневый шаблон



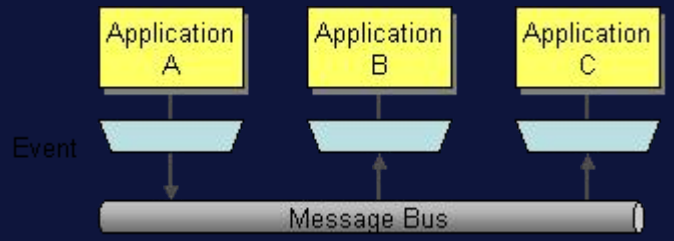
Файловый обмен



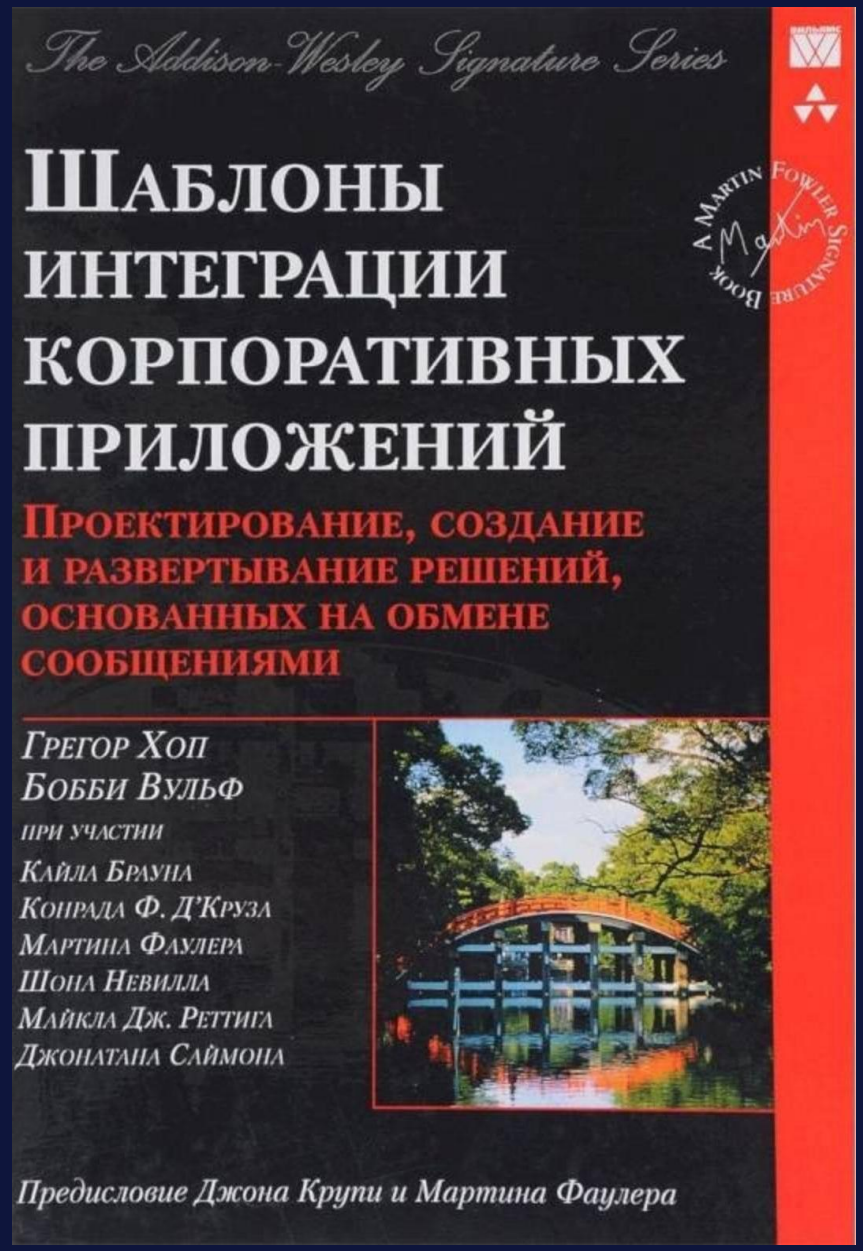
Удаленный вызов



Общая база данных



Асинхронная передача сообщений



Изображения Enterprise Integration Patterns, https://www.enterpriseintegrationpatterns.com

Обложка: Издательство «Диалектика-Вильямс»

Синхронность и асинхронность

Синхронность: блокировка процесса до получения ответа.

Что важнее:

минимизация задержек

latency

Синхронный

- проще*
- не требует дополнительных систем
- возможны потери
- труднее параллелить

или

увеличение пропускной способности?

throughput

Асинхронный

- сложнее в обработке
- нужно промежуточное ПО или постоянное соединение
- легко параллелить
- можно обрабатывать большие объемы

Синхронность и асинхронность

Синхронные

- SOAP
- GraphQL
- REST
- gRPC

Асинхронные

- SOAP
- MQTT
- GraphQL
- Kafka
- gRPC
- RabbitMQ
- WebSockets
- WebHook
- Long Polling
- SSE

Кадр из мультипликационного фильма «История игрушек 2», Walt Disney Company

Интеграционный стиль

Семантика запроса: что мы имеем в виду?

Шаблон | Протокол | Продукт | Язык

Синхронные (запрос-ответ):

- Вызов процедуры RPC (SOAP, gRPC)
- Операции CRUD над ресурсом (REST API Level 2)
- Гипермедиа (REST API Level 3: HATEOAS)
- Запрос данных по спецификации (GraphQL, общая БД / SQL)

Асинхронные:

- Публикация/подписка (WebHook, SSE, MQTT, GraphQL)
- Поток данных (Websockets, gRPC, HTTP Streaming)
- Очередь (AMQP, RabbitMQ, Redis)
- Журнал (Kafka, Redis)
- Передача файла с записями (файловый обмен, ETL)

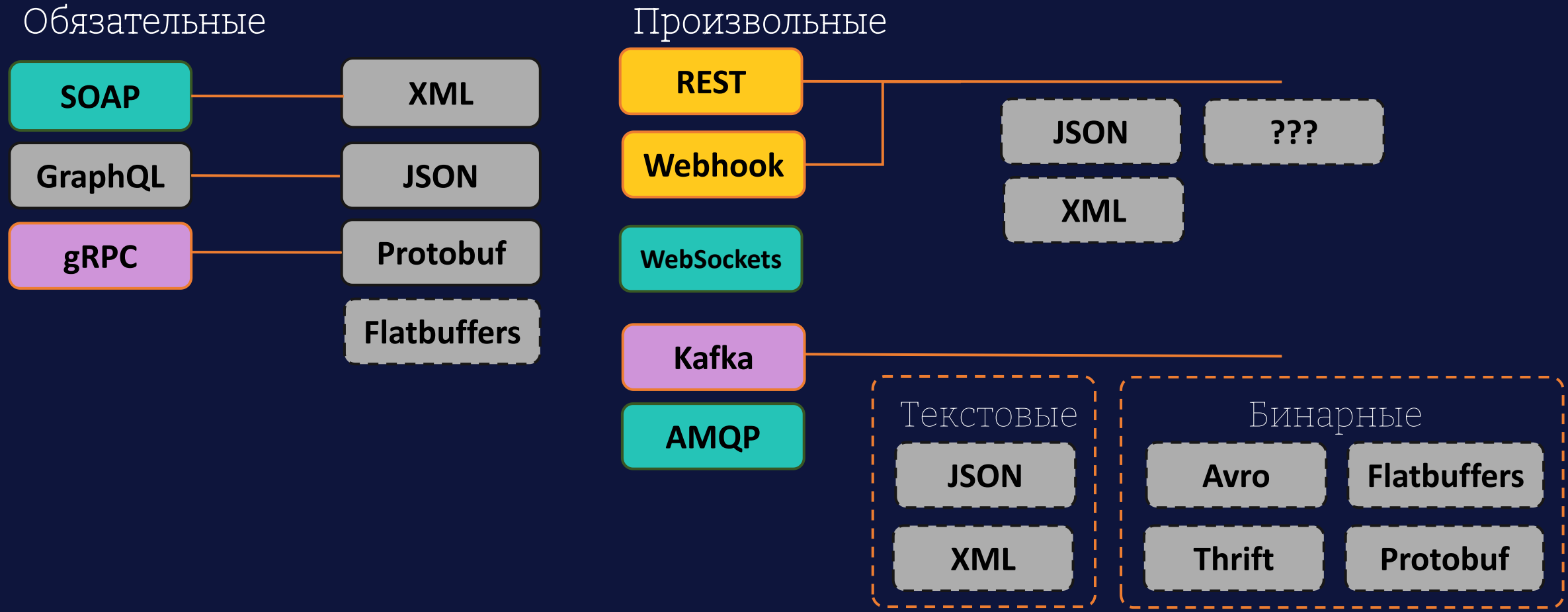
Протоколы интеграций

Главный вопрос: HTTP или низкоуровневый протокол?
(TCP/IP быстрее)

А если HTTP — это просто транспорт, или мы используем все его возможности? (Роутинг, кэширование, медиатипы...)

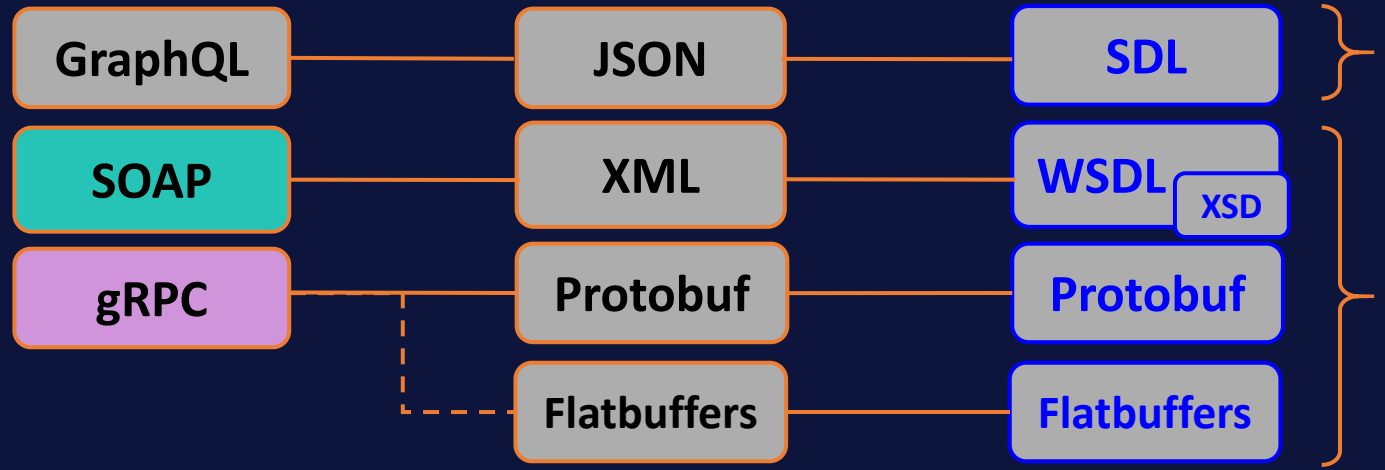
Форматы полезной нагрузки

Диктует ли технология формат? Он текстовый или бинарный?



Язык описания схемы данных

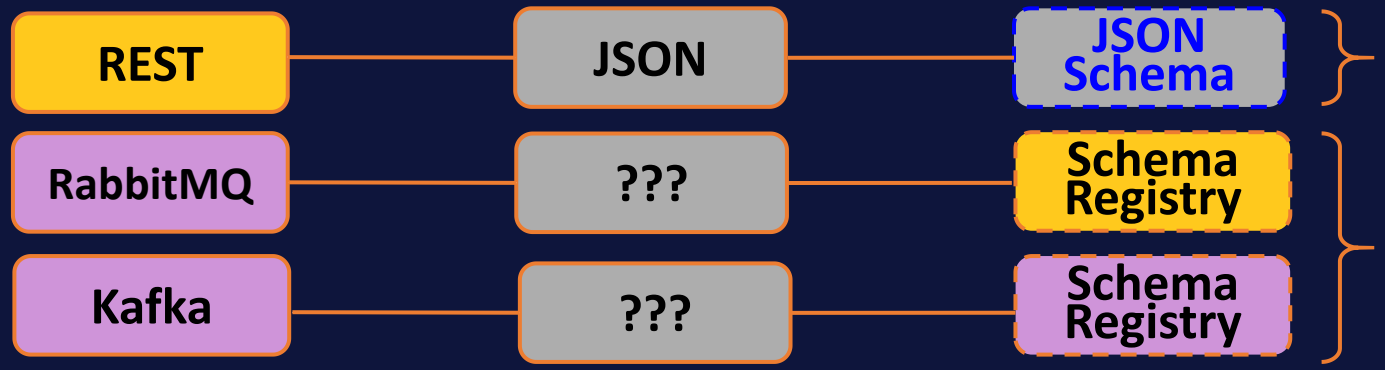
Обязательно использование схемы + кодогенерация клиентов / сервера



Определяет схему данных, фильтры и операции

Определяют и схему данных, и интерфейсы

Использование схемы опционально



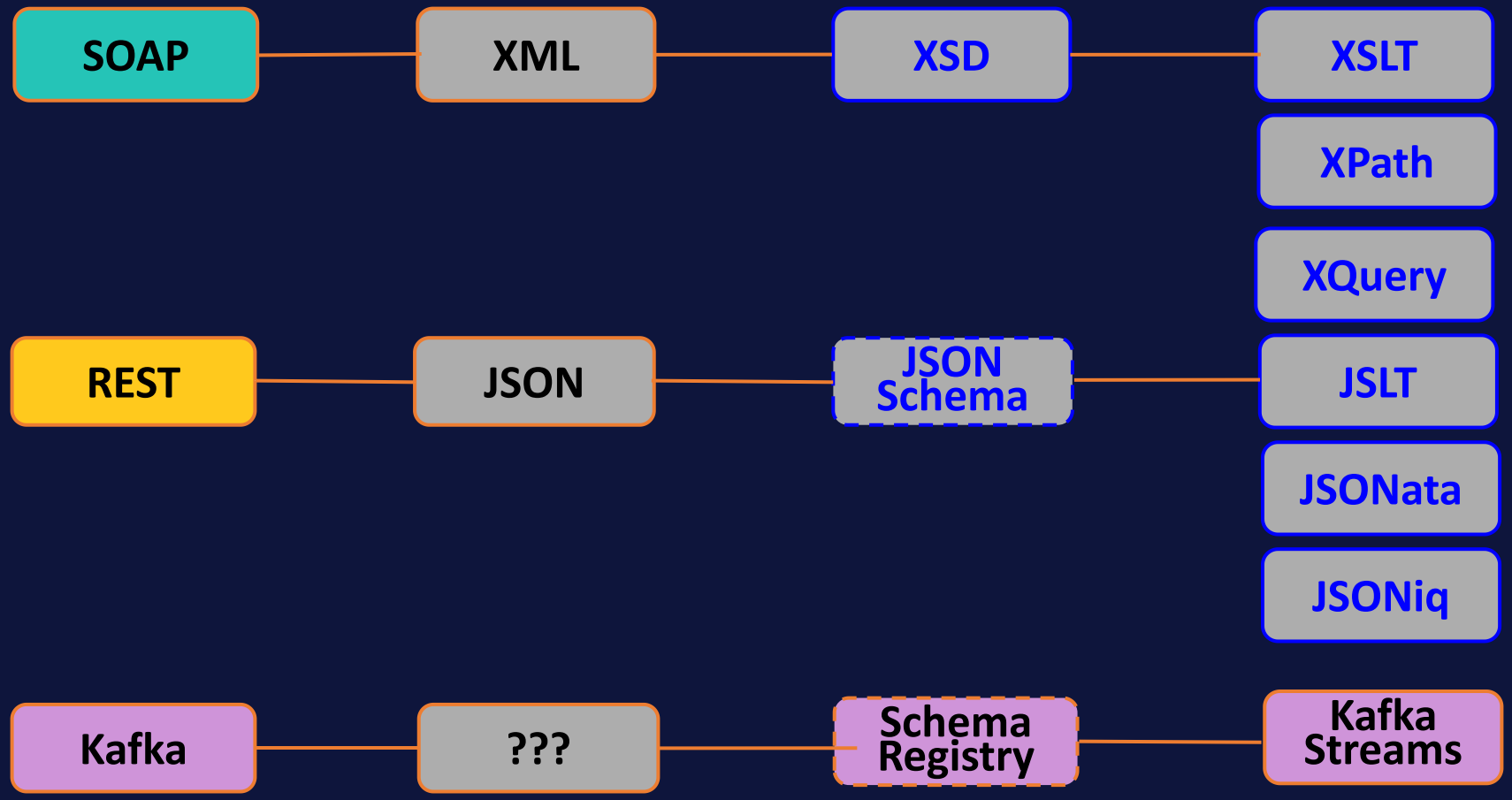
Кодогенерация при использовании OpenAPI

Кодогенерация при использовании AsyncAPI

Сравнение разных форматов и схем

	Обяз. Схема	Формат	Сложные структуры	Типизация	Размер и скорость парсинга
CSV	Нет	Текст	Нет, плоские таблицы	Динамическая	Самый маленький из текстовых, очень высокая скорость
XML	Нет	Текст / Бинарный EXI	Вложенные объекты, метаданные, пространства имен	Динамическая	Самый избыточный, средняя скорость
JSON	Нет	Текст	Вложенные объекты, массивы, перечисления	Слабая	Избыточный, быстрее XML
Protobuf	Да	Бинарный	Вложенные объекты, массивы, словари (maps), перечисления	Статическая, сильная	Зависит от данных. Меньше CSV, высокая скорость
Flatbuffers	Да / нет	Бинарный	Вложенные объекты, массивы, перечисления, прстранства имен	Статическая, сильная	Быстрее и меньше Protobuf, парсинг не нужен
Avro	Да / нет	Бинарный	Вложенные объекты, массивы, словари (maps), перечисления, прстранства имен	Динамическая	Меньше Protobuf, но медленнее

Что если нам нужны преобразования данных?



Стандартизованные языки запросов и трансформаций

Спецификации от частных разработчиков и сообществ

Обработку нужно написать самостоятельно

Обработка ошибок

Можно ли вводить свои кастомные ошибки?

SOAP
GraphQL
gRPC

Можно задавать собственные коды ошибок и их детализацию

REST

Стандартные коды ошибок HTTP, без возможности расширения. Но можно передать дополнительные поля — RFC 9457.

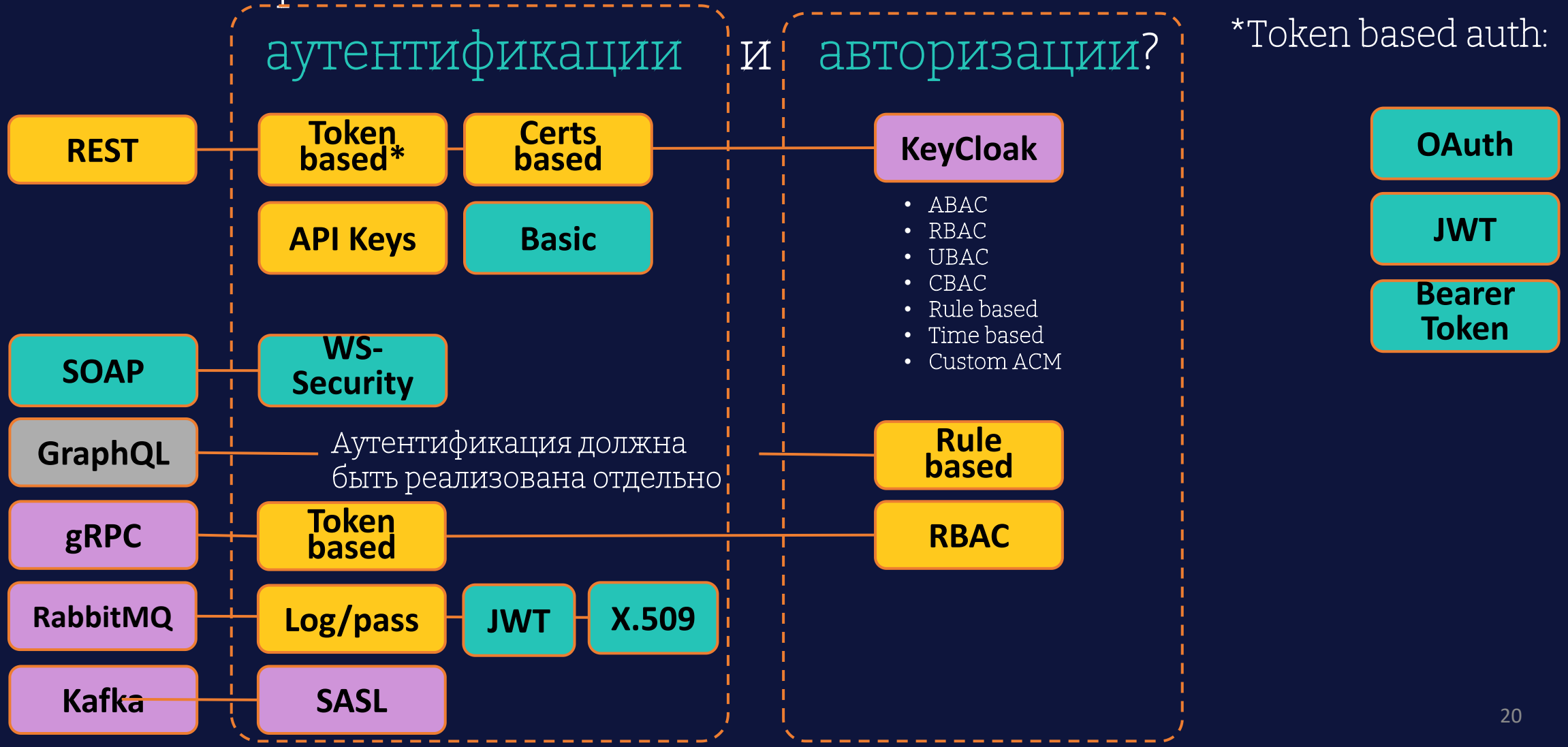
WebSockets
RabbitMQ
MQTT
Kafka

Нет механизмов обработки ошибок, реализовать самим на уровне консьюмеров

Безопасность

SSL/TLS — для шифрования есть везде

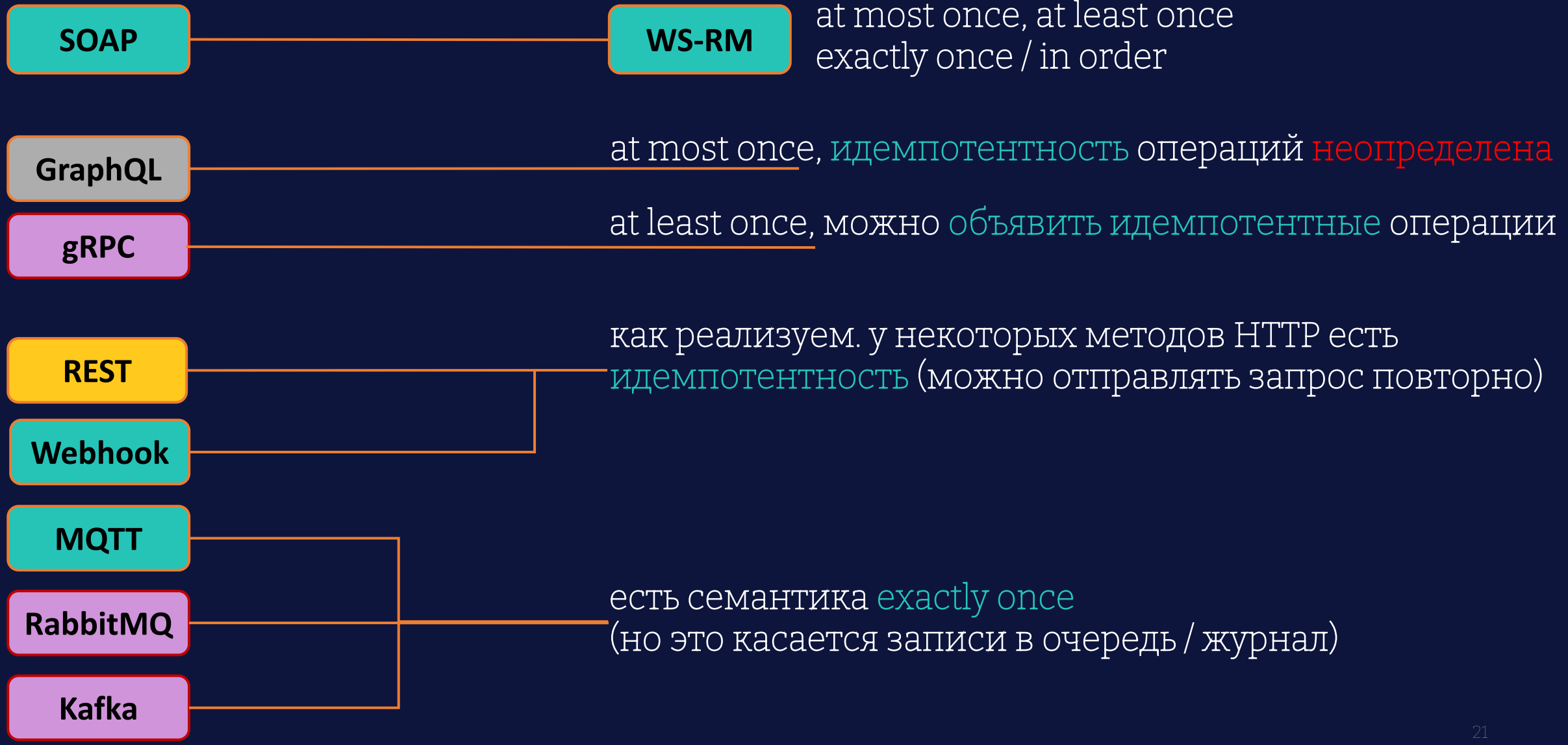
Есть ли встроенные механизмы аутентификации и авторизации?



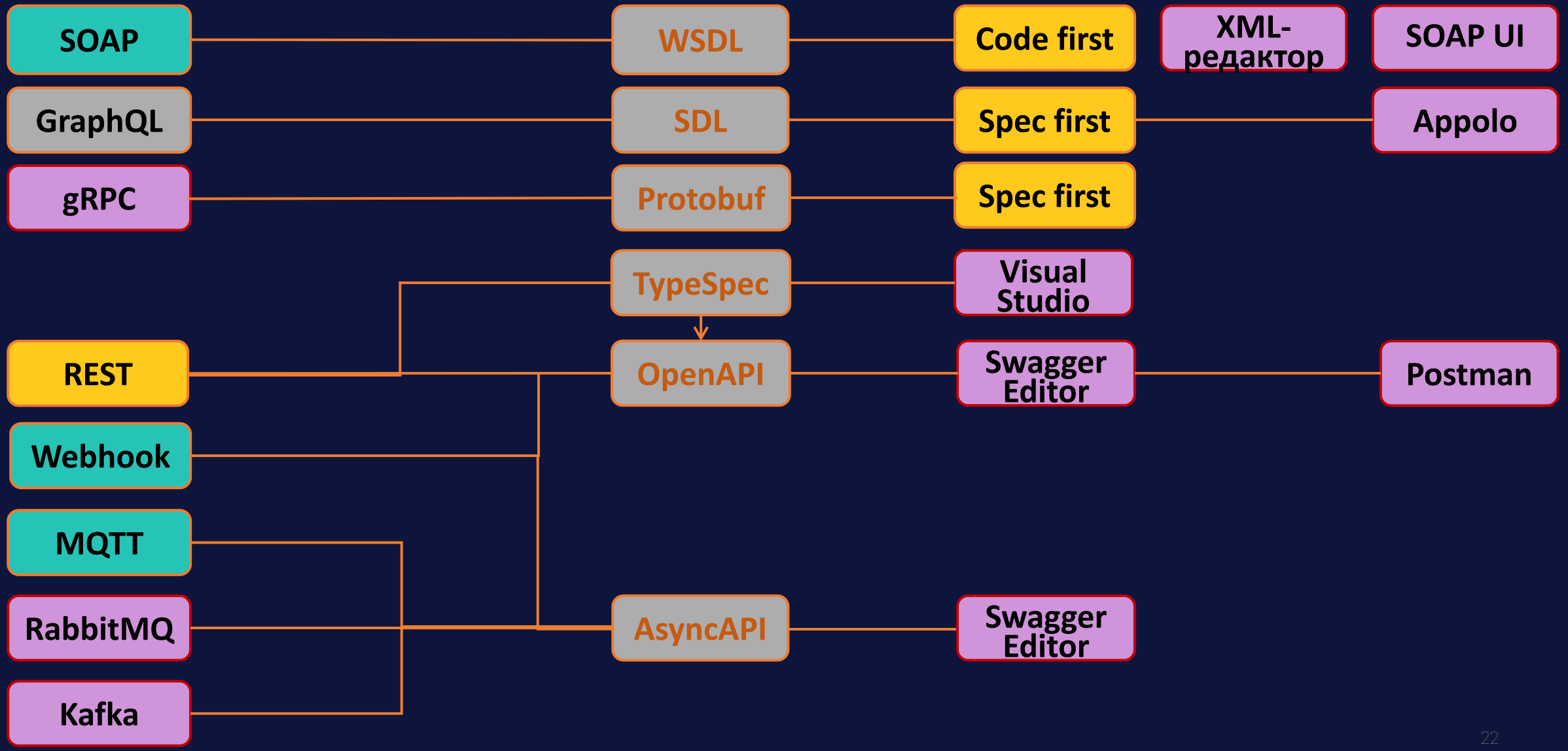
*Token based auth:

- OAuth
- JWT
- Bearer Token

Гарантии доставки



Языки спецификаций и инструменты



«Фишки» разных технологий

SOAP — высокая степень стандартизации

REST — гибкость, множество инструментов, наблюдаемость

GraphQL — возможность развития, объединение схем из разных сервисов, борьба с under-fetching / over-fetching

gRPC — скорость, двунаправленность, одно соединение HTTP/2

Kafka — высокая надежность хранения, распараллеливание, высокая пропускная способность

RabbitMQ — возможность гибкого роутинга сообщений

Кейсы использования

SOAP — legacy, гос. сервисы, ответственные системы

REST — весь web, внешние API

GraphQL — внешние API, разнородные клиенты, быстро развивающиеся микросервисы (федеративная схема)

gRPC — микросервисы, телеметрия, оповещения / события

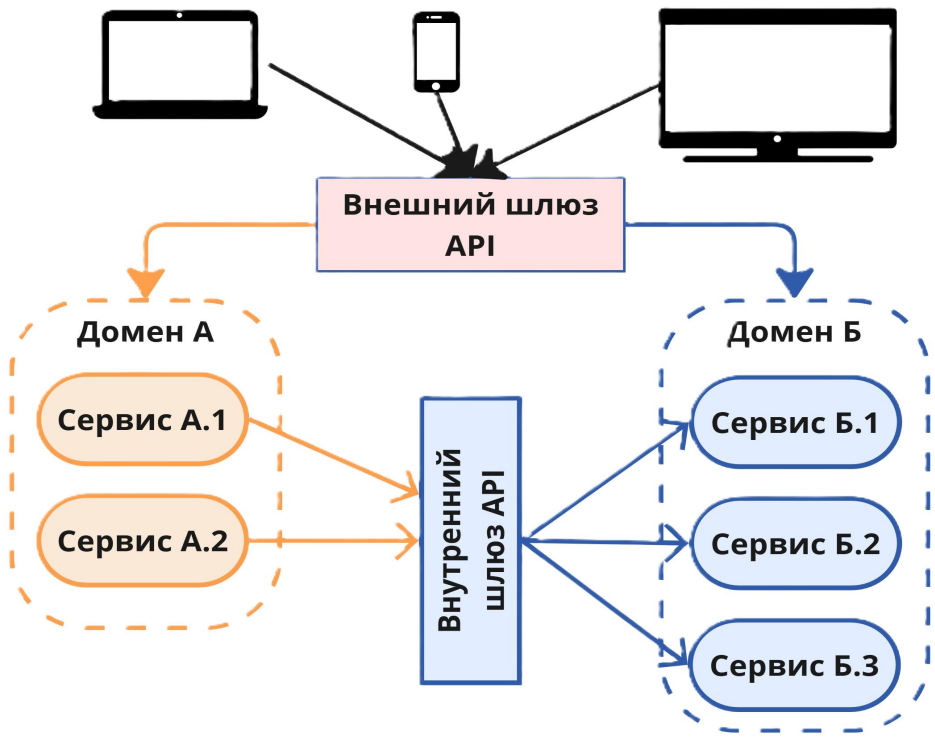
MQTT — умные дома, IoT, телеметрия

Kafka — организация потоков данных для аналитики, EDA

RabbitMQ — запуск длительных фоновых задач, микросервисы, EDA

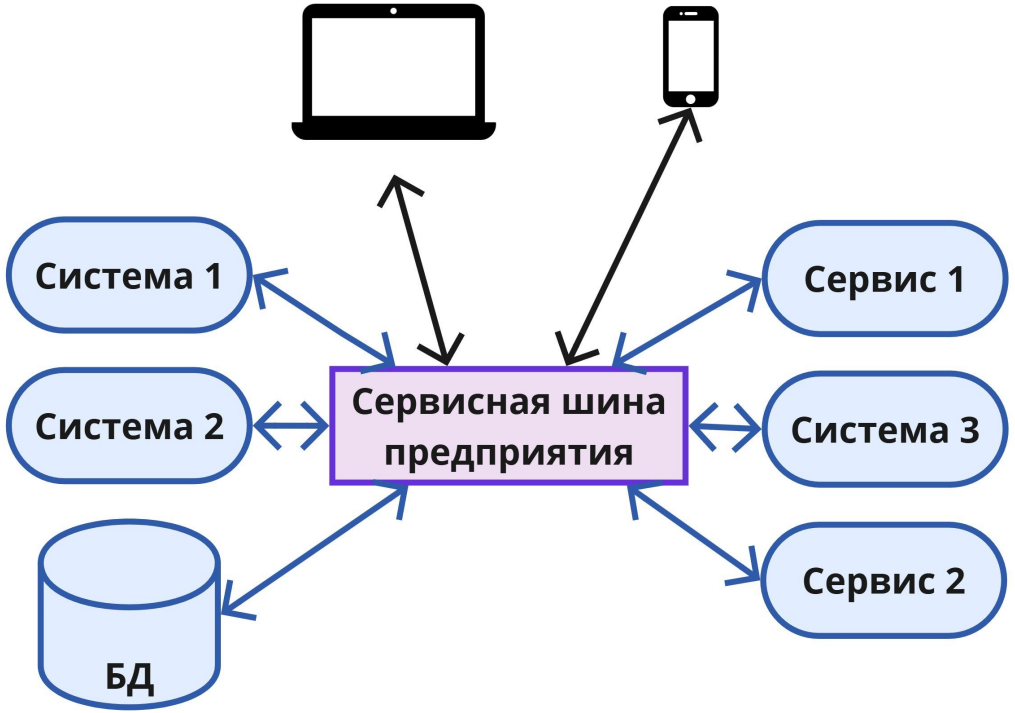
Паттерны «промежуточного ПО»

API gateway, Reverse Proxy



Яляется единой точкой входа для внешних запросов. Обеспечивает контроль доступа, аутентификацию, маршрутизацию, ограничение частоты запросов, балансировку, сбор аналитики.

ESB, Integration framework



Централизованная система обеспечения интеграций. Поддерживает множество протоколов, обеспечивает безопасность, масштабируемость, преобразование данных, контроль ошибок, гарантии доставки, обеспечение транзакций, мониторинг и т.п.

Оценка новой (для нас) технологии

JSON-RPC:

Паттерн:

Режим взаимодействия:

Семантика запроса:

Протокол:

Формат:

Ошибки:

Инструменты преобразования:

Безопасность:

Семантика доставки:

Язык спецификаций:

Оценка новой (для нас) технологии

JSON-RPC: протокол

Паттерн: удаленный вызов

Режим взаимодействия: синхронный

Семантика запроса: RPC

Фишки: легковесный, пакетные запросы

Протокол: не зависит от протокола. Реализации: HTTP, Websockets, TCP

Формат: текстовый, JSON, свой формат вызовов и ответов. Схемы нет.

Ошибки: встроенные + кастомные

Инструменты преобразования: нет

Безопасность: встроенная в HTTP (SSL)

Семантика доставки: at-most-once

Язык спецификаций: OpenRPC

Проблемы: затруднен роутинг, кэширование, мониторинг, передача бинарных данных

Оценка новой (для нас) технологии

RMI:

Паттерн:

Режим взаимодействия:

Семантика запроса:

Протокол:

Формат:

Ошибки:

Инструменты преобразования:

Безопасность:

Семантика доставки:

Язык спецификаций:

Оценка новой (для нас) технологии

RMI: протокол удаленного вызова методов

Паттерн: удаленный вызов

Режим взаимодействия: синхронный

Семантика запроса: вызов методов удаленного объекта

Протокол: TCP

Формат: бинарный. Типизация как в Java

Ошибки: встроенные (Java Exceptions)

Инструменты преобразования: нет

Безопасность: нет

Семантика доставки: at-most-once

Фишки: нативный для Java

Проблемы: нет поддержки вне Java

Язык спецификаций: Java (Interface). Есть общий реестр: RMI Registry

Оценка новой (для нас) технологии

Apache Thrift:

Паттерн:

Режим взаимодействия:

Семантика запроса:

Протокол:

Формат:

Схема:

Ошибки:

Инструменты преобразования:

Безопасность:

Семантика доставки:

Язык спецификаций:

Оценка новой (для нас) технологии

Apache Thrift: язык IDL + фреймворк кодогенерации

Паттерн: удаленный вызов

Режим взаимодействия: синхронный, асинхронный

Семантика запроса: RPC

Протокол: TCP-сокеты, файл, память, IPC-сокеты, HTTP

Формат: бинарный, компактный, JSON.

Схема: обязательна. Строгая типизация. Сложные структуры.

Ошибки: специальный тип exception

Инструменты преобразования: нет

Безопасность: TLS, SASL

Семантика доставки: at-most-once

Язык спецификаций: Thrift

Фишки: вариативность форматов и протоколов

Проблемы: плохая документация, разные функции в разных языках. Используют в основном для сериализации данных.

Выбор технологии интеграций

- Возможности систем, API, протоколы
- Объем данных (одного сообщения, всех данных)
- Скорость или пропускная способность? (Синхр. /асинхр.)
- Смысл (семантика) запроса
- Форматы и схемы данных (сложность, типобезопасность, размер, скорость сериализации / десериализации, эволюция)
- Преобразование данных — нужно ли?
- Кодогенерация
- Передача бизнес-ошибок
- Безопасность, аутентфикация, управление доступом
- Уровень гарантии доставки
- Наличие инструментов спецификации / тестирования / наблюдения

Как использовать карту?

- Изучить всё, что на ней нанесено
- Отметить технологии и инструменты, которые используются в компании, и кейсы применения, чтобы выбирать технологию по назначению
- Как радар технологий: используем, пробуем, присматриваемся

Спасибо за внимание

Вопросы?

Последняя версия карты в канале