# Серебряная пуля для интеграций

Или Опять этот "it depends..."

# Чернухин Максим

- 10+ лет в IT
  - 3 линия поддержки
  - Back, Front, ML и тд
  - Архитектура, инфраструктура
  - CTO
- Люблю ТРИЗ и сложные задачки
- Нравится восходить на горы и экстрим
- Есть 30 и 40-летний бонсай

facebook.com/chernukhin.maksim

TG @MaksCher

# О чём сегодня поговорим

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# Выделение сервисов

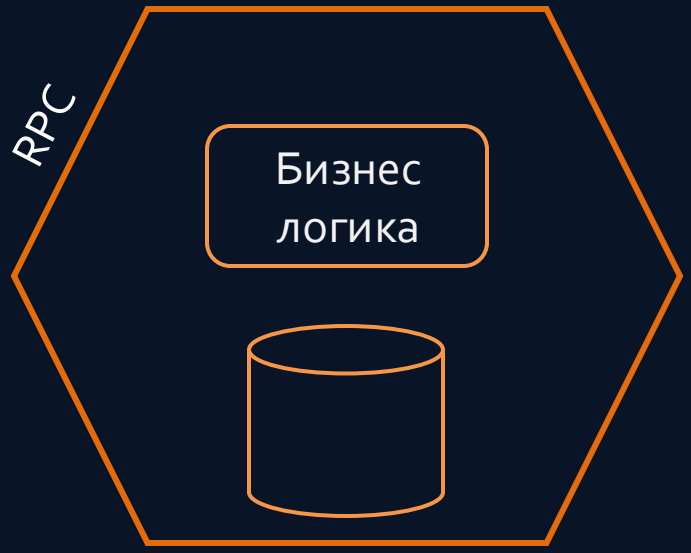
Domain / домен – изолированная область отвечающая за определенные бизнес сущности

Разработка API это всегда немного архитектура

# Выделение сервисов

Domain / домен – изолированная область отвечающая за определенные бизнес сущности

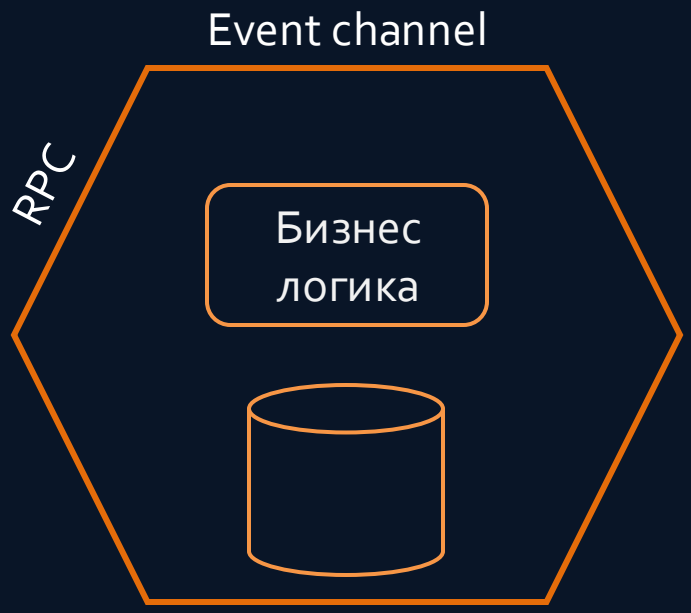
Разработка API это всегда немного архитектуры



# Выделение сервисов

Domain / домен – изолированная область отвечающая за определенные бизнес сущности

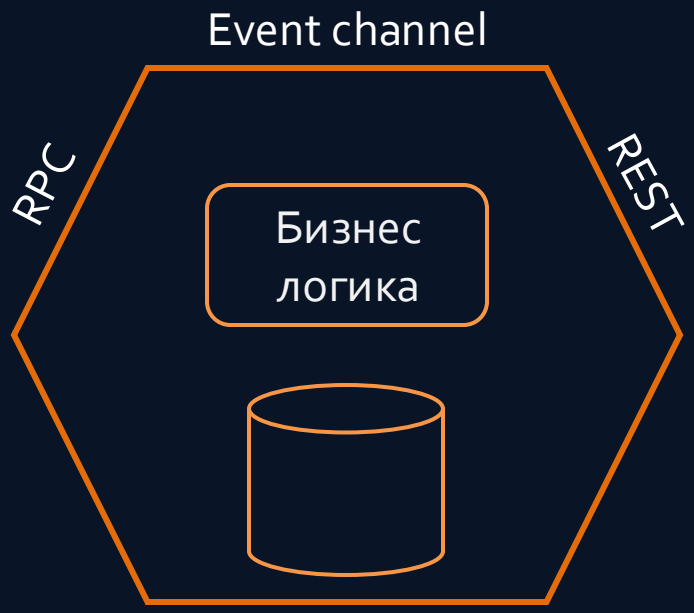
Разработка API это всегда немного архитектуры



# Выделение сервисов

Domain / домен – изолированная область отвечающая за определенные бизнес сущности

Разработка API это всегда немного архитектуры



# Проектирование сервиса

- Принцип единственной ответственности

# Проектирование сервиса

- Принцип единственной ответственности
- Изолирование внутренних функций

# Проектирование сервиса

- Принцип единственной ответственности
- Изолирование внутренних функций
- Четкие границы бизнес логики

# Проектирование сервиса

- Принцип единственной ответственности
- Изолирование внутренних функций
- Четкие границы бизнес логики
- Никакой скрытой логики в API

# Проектирование сервиса

- Принцип единственной ответственности
- Изолирование внутренних функций
- Четкие границы бизнес логики
- Никакой скрытой логики в API
- Предсказуемость поведения

# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# Интеграции

- Синхронная
  - Время ответа важно
  - Процесс на стороне клиента останавливается
  - Простота

# Интеграции

- Синхронная
  - Время ответа важно
  - Процесс на стороне клиента останавливается
  - Простота

- Асинхронная
  - Время ответа не важно
  - Клиент не ждёт
  - Сложность разработки отладки

# Немного практики

Сервис который хранит знания о:

- Городах
- Банкоматах

Отвечает за их обновление и удаление и актуальное состояние

# Мы тут

- Интеграции про что стоит не забыть
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# REST

REST - Representational State Transfer

# REST

REST - Representational State Transfer

Что надо помнить:

- HTTP Методы (GET, POST, DELETE, PUT, PATCH)
- Кеширование на балансировщиках
- Stateless
- HTTP codes
- Управление ресурсами

# Когда подойдет

- Есть выделенные ресурсы
- Нет запуска процессов
- Необходимость кеширования
- Вы не хотите усложнений

# Когда подойдет

- Есть выделенные ресурсы
- Нет запуска процессов
- Необходимость кеширования
- Вы не хотите усложнений
- Вы любите JSON =)

# Немного практики

Система которая хранит знания о:

- Городах
- Банкоматах

GET host.com/cities/id

# Немного практики

Система которая хранит знания о:

- Городах
- Банкоматах

GET host.com/cities/id

{“ID”: “123”,
“Name”: “Москва”
“FederalDistrict”: “Центральный”}

# Фильтрация

GET host.com/cities?FederalDistrict=Центральный


```
[ {“ID”: “123”,
  “Name”: “Москва”
  “FederalDistrict”: “Центральный”},
{“ID”: “124”,
  “Name”: “Мытищи”
  “FederalDistrict”: “Центральный”} ]
```

# Фильтрация

GET host.com/cities?FederalDistrict=Центральный


```
"Data" : [
        {“ID”: “123”,
         “Name”: “Москва”
         “FederalDistrict”: “Центральный”},
        {“ID”: “123”,
         “Name”: “Москва”
         “FederalDistrict”: “Центральный”} ],
Pagination {
        "limit": 10,
        "offset": 45,
        "sort": " ID :desc"}
```

# Вложенность и иерархия

GET host.com/cities/atms

GET host.com/atms

# Вложенность и иерархия

1. GET host.com/cities/{cityID}/atms
2. GET host.com/atms?cityID={id}

# Вложенность и иерархия

POST host.com/cities/{cityID}/atms

```
 {
  “Name”: “Смоленская плаза”
  “Kind”: [“USD”,”EURO”]
}
```

# Вложенность и иерархия

GET host.com/cities/{cityID}/atms/{atmID}

# Вложенность и иерархия

GET host.com/cities/{cityID}/atms/{atmID}

404 not found

# Вложенность и иерархия

GET host.com/cities/{cityID}/atms/{atmID}

404 not found

cityID – Нет

atmID – Есть

# Вложенность и иерархия

Только один ID в path

GET host.com/cities/atms/{atmID}

# Ошибки

## Не забывайте использовать HTTP code

Response code : 500

```
{
  "status": INTERNAL_SERVER_ERROR,
  "message": "Внутренняя ошибка сервиса"
}
```

# Ошибки

## Используем HTTP code

Response code : 400


```
{
"status":"VALIDATION_ERROR”
"message":"Некорректные данные",
"details":{
                "violations":[
                                {
                                  "parameter":"surveyId",
                                  "type": "PATH_VARIABLE",
                                  "message":"Некорректный формат значения "102325""
                                },
                                {
                                   "parameter":"page",
                                   "type": "FIELD",
                                   "message":"Некорректный тип данных"
                                }]
                }
 }
```

# Итоги по REST

- Иерархия ресурсов
- Единообразие контрактов и ошибок
- HTTP codes

# Добавление логики

Добавим метро, офисы и услуги

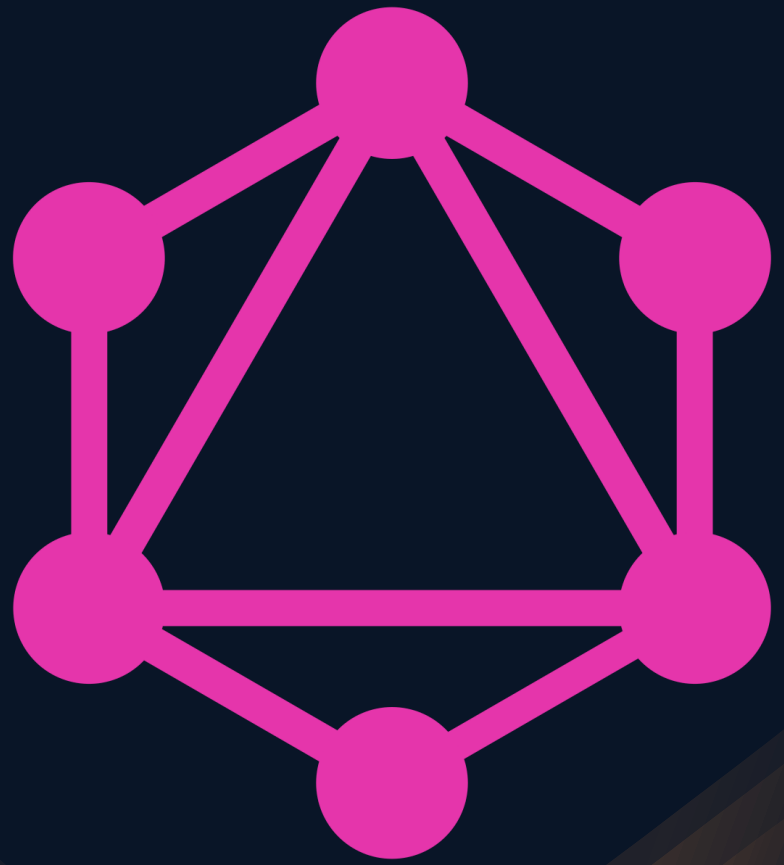
host.com/cities/metro/offices/atms

# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# GraphQL

GraphQL – язык запросов к API разработанный в FB



# GraphQL

Основные отличия:

- Получение только нужных данные
- Все ресурсы – один запрос
- Многоуровневая фильтрация

# GraphQL

Запрос:

```
query {
  getOffices {
    offices {
      distance
      title
      locations {
        lat
        lon
        address {
          house
          street
        }
      }
      kinds
      metro {
        cityName
        distance
        lat
        lon
        lineName
        name
      }
    }
  }
}
```

# GraphQL

Ответ:


```
"data": {
    "getOffices": {
        "offices": [
            {
                "distance": 416,
                "title": "ДО «СМОЛЕНСКАЯ»",
                "locations": {
                        "lat": 55.744747,
                        "lon": 37.582718,
                        "address": {
                            "house": "27",
                            "street": "пл Смоленская-Сенная"
                        }
                    },
                "kinds": [
                        "retailStandart",
                        "retailVip",
                        "vip"
                        ],
                "metro": [
                    {
                        "cityName": "Москва",
                        "distance": 336,
                        "lat": 55.747713,
                        "lon": 37.583802,
                        "lineName": "Арбатско-Покровская",
                        "name": "Смоленская"
                    },
                    {
```

# GraphQL

Запрос:

```
query {
  getOffices(filters: { metro: "Смоленская", kinds: "vip" }) {
    offices {
      distance
      title
      locations {
        lat
        lon
        address {
          house
          street
        }
      }
      kinds
      metro {
        cityName
        distance
        lat
        lon
        lineName
        name
      }
    }
  }
}
```

# GraphQL изменения данных

Мутация:

```
mutation {
  createOffice(input: {
    title: "ДО «СМОЛЕНСКАЯ»"
    distance: 416
    locations: {
      lat: 55.744747
      lon: 37.582718
      address: {
        house: "27"
        street: "пл Смоленская-Сенная"
      }
    }
    kinds: ["retailStandart", "retailVip", "vip"]
    metro: [
      {
        cityName: "Москва"
        distance: 336
        lat: 55.747713
        lon: 37.583802
        lineName: "Арбатско-Покровская"
        name: "Смоленская"
      }
    ]
  }) {
    office {
      id
      title
      distance
      locations {
        lat
        lon
        address {
          house
          street
        }
      }
      kinds
      metro {
        cityName
        distance
        lat
        lon
        lineName
        name
      }
    }
    success
    message
  }
}
```

# GraphQL итоги

- Гибкость API
- Схемы контрактов
- Сложность разработки
- Рекурсия в фильтрах

# Запуск процессов

Обновить данные банкомата, запустив внутренний процесс

# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# RPC

Remote Procedure Call – вызов процедуры (функции или метода) на удаленном сервере

- SOAP
- JSON RPC
- gRPC

# RPC

Обновление данных банкомата

```
{
 "jsonrpc": "2.0",
 "method": ”updateATM",
 "params": {
          ”firstParam": ”231”
         ”kind": ”USD”
          },
 "id": 1
}
```

# RPC

Обновление данных банкомата

```
{
 "jsonrpc": "2.0",
 "result": {
          ”status": ”done”
            },
 "id": 1
}
```

# RPC

gRPC

- Важно уменьшить трафик
- Есть требование к схеме .proto
- Готовы работать с HTTP2.0

# RPC

gRPC

- Важно уменьшить трафик
- Есть требование к схеме .proto
- Готовы работать с HTTP2.0

SOAP

- Необходим ГОСТ
- Очень любите XML =)
- Понимаете XML External Entities (XXE)

# RPC итоги

- Необходимо запустить какой-то процесс

# RPC итоги

- Необходимо запустить какой-то процесс
- Вызвать метод или функцию

# RPC итоги

- Необходимо запустить какой-то процесс
- Вызвать метод или функцию
- JSON RPC и gRPC

# RPC итоги

- Необходимо запустить какой-то процесс
- Вызвать метод или функцию
- JSON RPC и gRPC и НЕ SOAP

# Обновились данные

Необходимо получать уведомления, о обновление данных от банкомата

# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# Очереди и события

Интеграции с помощью очередей на базе:

- Kafka
- RabbitMQ
- IBM MQ
- ActiveMQ

# Очереди и события

Интеграции с помощью очередей на базе:

- Kafka
- RabbitMQ
- IBM MQ
- ActiveMQ

# Очереди и события

- Асинхронное взаимодействие

# Очереди и события

- Асинхронное взаимодействие
- Буфер для нагрузки

# Очереди и события

- Асинхронное взаимодействие
- Буфер для нагрузки
- Устойчивость при падении систем

# Очереди и события

- Асинхронное взаимодействие
- Буфер для нагрузки
- Устойчивость при падении систем
- Сложность

# Очереди и события

- Асинхронное взаимодействие
- Буфер для нагрузки
- Устойчивость при падении систем
- Сложность
- DLQ


# Очереди и события

- Асинхронное взаимодействие
- Буфер для нагрузки
- Устойчивость при падении систем
- Сложность
- DLQ


# Очереди и события

- Асинхронное взаимодействие
- Буфер для нагрузки
- Устойчивость при падении систем
- Сложность
- DLQ


# Типы событий

- Event Notification
- Event-Carried State Transfer
- Event-Sourcing
- CQRS

# Типы событий

- Event Notification
- Event-Carried State Transfer
- Event-Sourcing
- CQRS


**Martin Fowler**

# Event Notification

- Сообщение о новом стоянии ресурса
- Низкая связанность
- Простота реализации

# Event-Carried State Transfer

- Передаем в сообщении все данные
- Клиенты сохраняют данные к себе
- Данные начинают хранится в разных системах

# Event-Sourcing

- Лог изменений системы или ресурса (инкримент)
- Сложность разработки
- Чаще всего используют для хранения данных в системе

# CQRS

- Два потока
  - Команды (изменение)
  - Чтение
- Сложность разработки и CRUD

# Изменения состояния банкомата

Event Notification

Очередь

ATMStoSubscribers

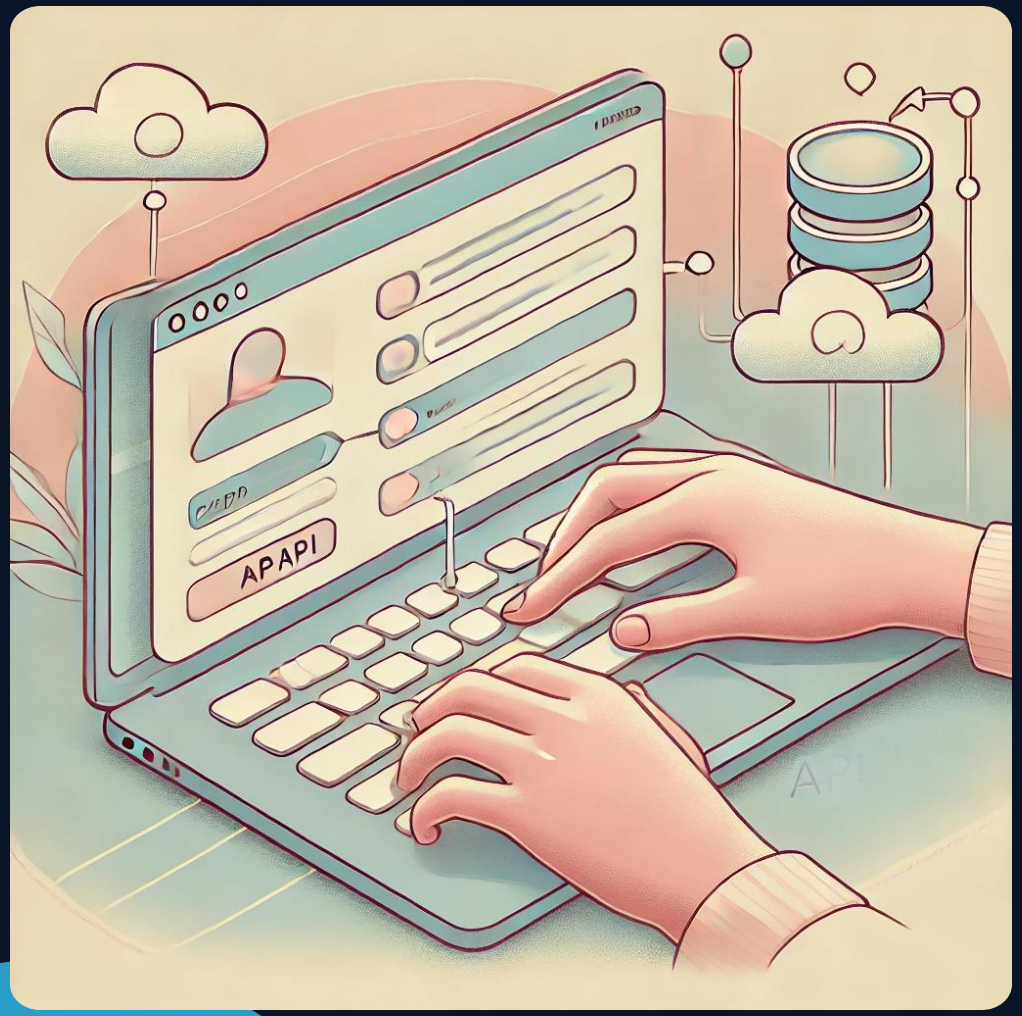
```
{
  ”USD": ”10.000",
  ”Euro": ”10.000",
  ”adress": {
       "house": "27"
       "street": "пл Смоленская-Сенная"
     }
}
```

# Очереди и события итоги

- Есть события
- Необходим буфер для нагрузки
- Отказоустойчивость
- Не забываем DLQ
- Сложность разработки

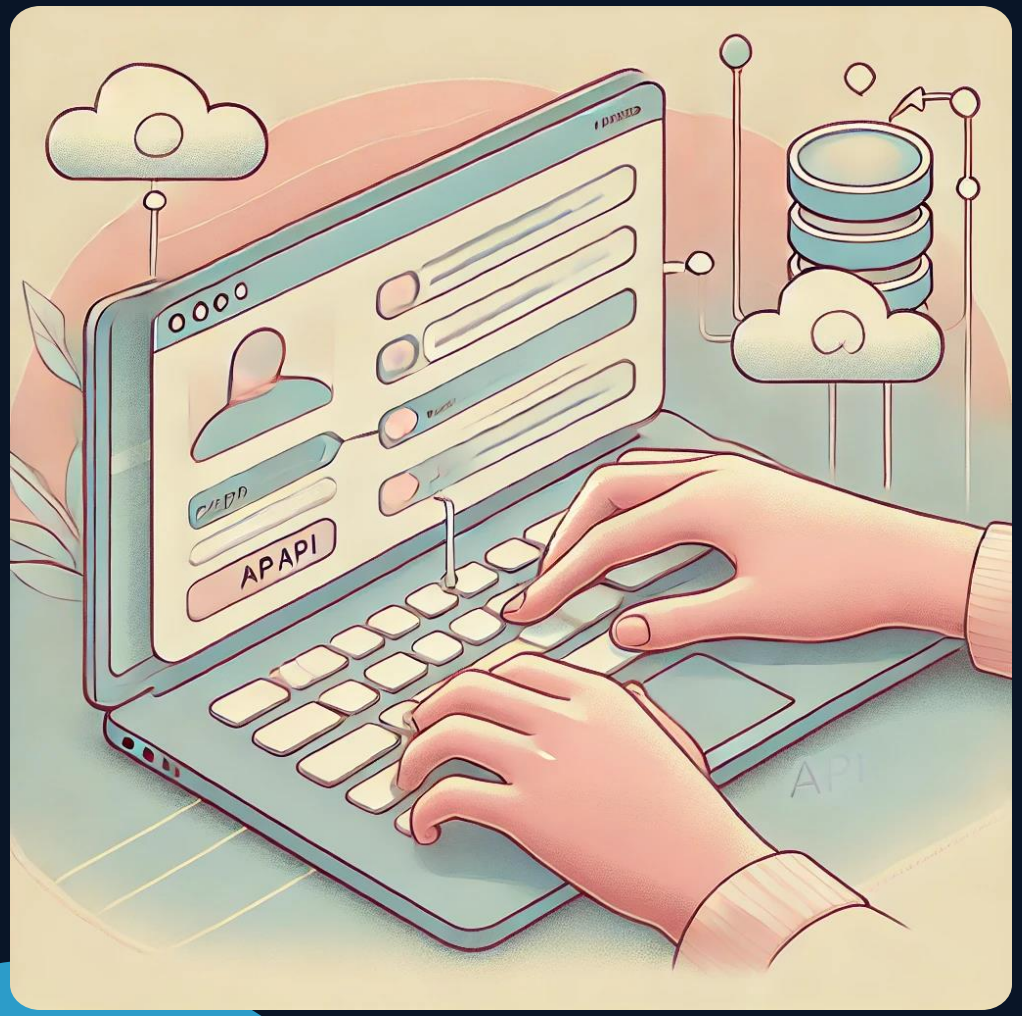
# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы



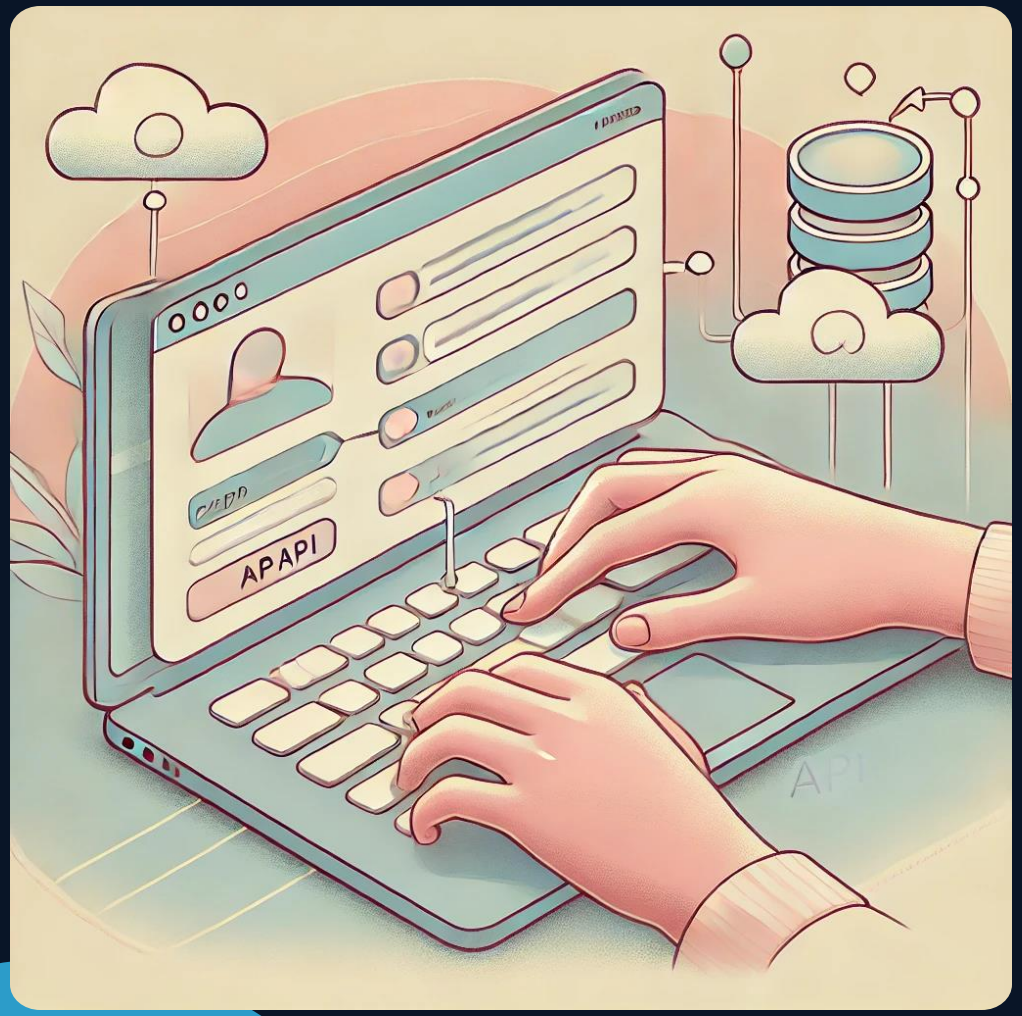

# BPM и управление процессами

Необходимо разработать систему, которая будет управлять процессом заполнения заявки на кредитную карту




# BPM и управление процессами

Необходимо разработать систему, которая будет управлять процессом заполнения заявки на кредитную карту




# BPM и управление процессами

Необходимо разработать систему
которая будет управлять проце
заполнения заявки на кредитну

# BPM

GET host.com/aplications/{aplicationD}

```
 {
 “FIO”: “FIO”,
 “City”: “Moscow”,
 “Product”: “Product”,
 “Status”: “In progress”,
 “Steps”: {
            “current”: “checking-whitelist”,
            “previous”: “Init”
            “next”:[ “step3”, “step5”]
            },
}
```

# BPM

POST host.com/aplications/{aplicationD}/init

Body

{“FIO”: “FIO”}

Response:


```
{
 “FIO”: “FIO”,
 “Steps”: {
            “current”: “checking-whitelist”,
            “previous”: “init”
             “next”:[ “step3”, “step5”]
}
}
```

# BPM

POST host.com/aplications/{aplicationD}/checking-whitelist

Body

{“City”: “Moscow”,
“Product”: “Product”}

Response:


```
{
 “City”: “Moscow”,
  “Product”: “Product”,
 “Steps”: {
              “current”: “step3”,
              “previous”: “checking-whitelist”
               “next”:[ “step6”]
}}
```

# BPM итоги

- А надо ли вам это
- Статусная модель
- Движение по шагам
- Единообразие подхода

# Мы тут

- С чего стоит начать
- Виды интеграций
- REST и немного DDD
- GraphQL или когда REST’а не хватило
- RPC
- Очереди и события
- BPM и управление процессами
- Выводы

# Выводы

- It depends

# Выводы

- It depends
- Использовать разные протоколы в одной системе это нормально

# Выводы

- It depends
- Использовать разные протоколы в одной системе это нормально
- Не усложнять

# Выводы

- It depends
- Использовать разные протоколы в одной системе это нормально
- Не усложнять
- Предсказуемость

# Выводы

- It depends
- Использовать разные протоколы в одной системе это нормально
- Не усложнять
- Предсказуемость
- Best practices

# Вопросы

- facebook.com/chernukhin.maksim
- TG @MaksCher