ОСНОВЫ WEB-РАЗРАБОТКИ

Лекция 11. REST API

Курс читают:

Шульман В.Д. @ShtuzerVD
Пелевина Т.В. @anivelat
Шабанов В.В. @ZeroHug
Шумилин В.В @Nodthar1107

REST & API & RESTFUL. ОПРЕДЕЛЕНИЯ

RESTful-сервис – это сервер, взаимодействие с которым реализовано по архитектурному паттерну **REST.**

REST — REpresentational State Transfer (передача состояния представления). Это архитектурный подход для разработки **API**.

API — (**Application programming interface**). Это контракт, который предоставляется программой (в нашем случае это серверная часть приложения, написанная на Golang).

API


CONTRACT
AGREEMENT made this
day of
20
(First Party)
(Second Party)

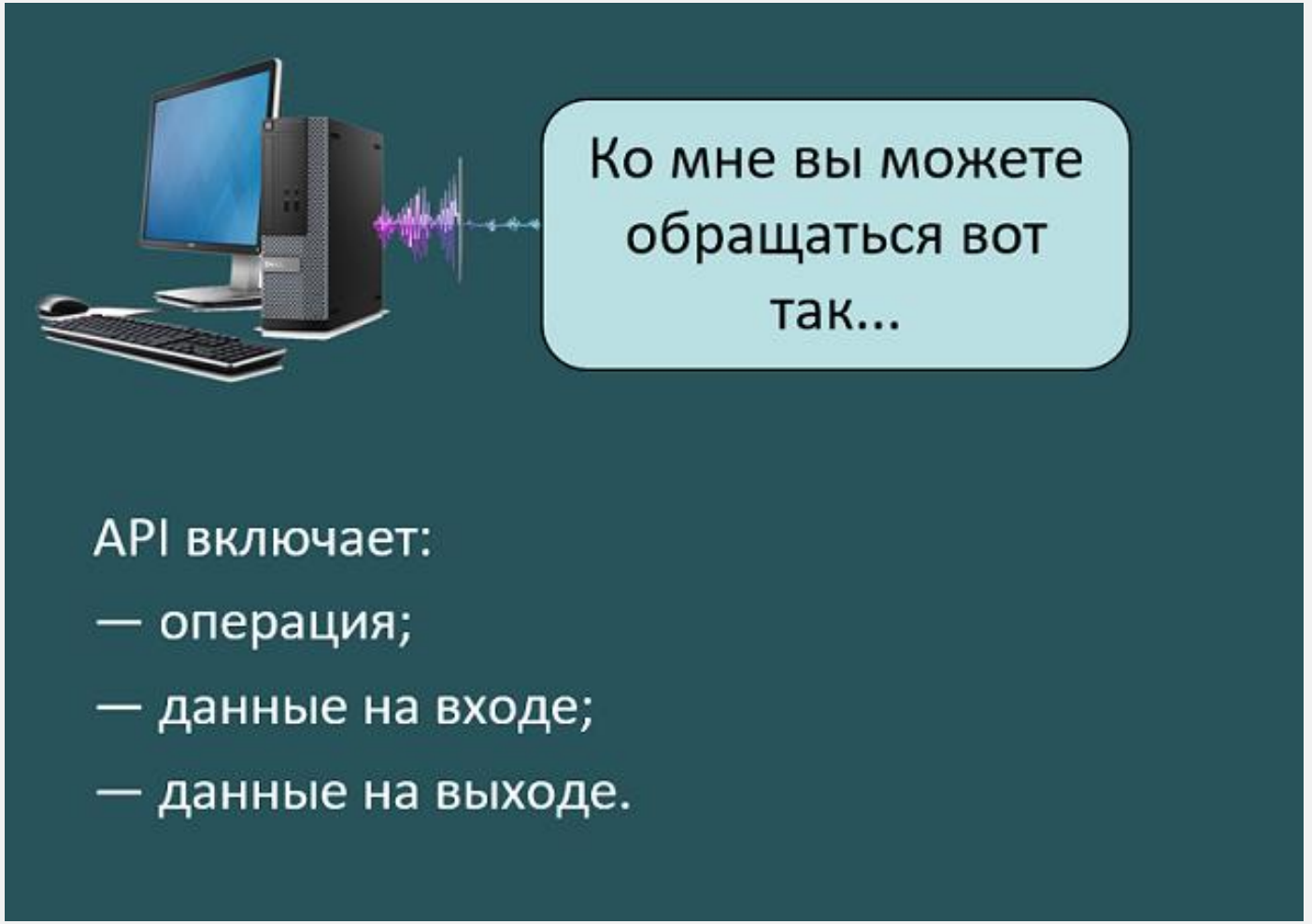
API

API (Application programming interface) — это контракт, который предоставляет программа. «Ко мне можно обращаться так и так, я обязуюсь делать то и это».

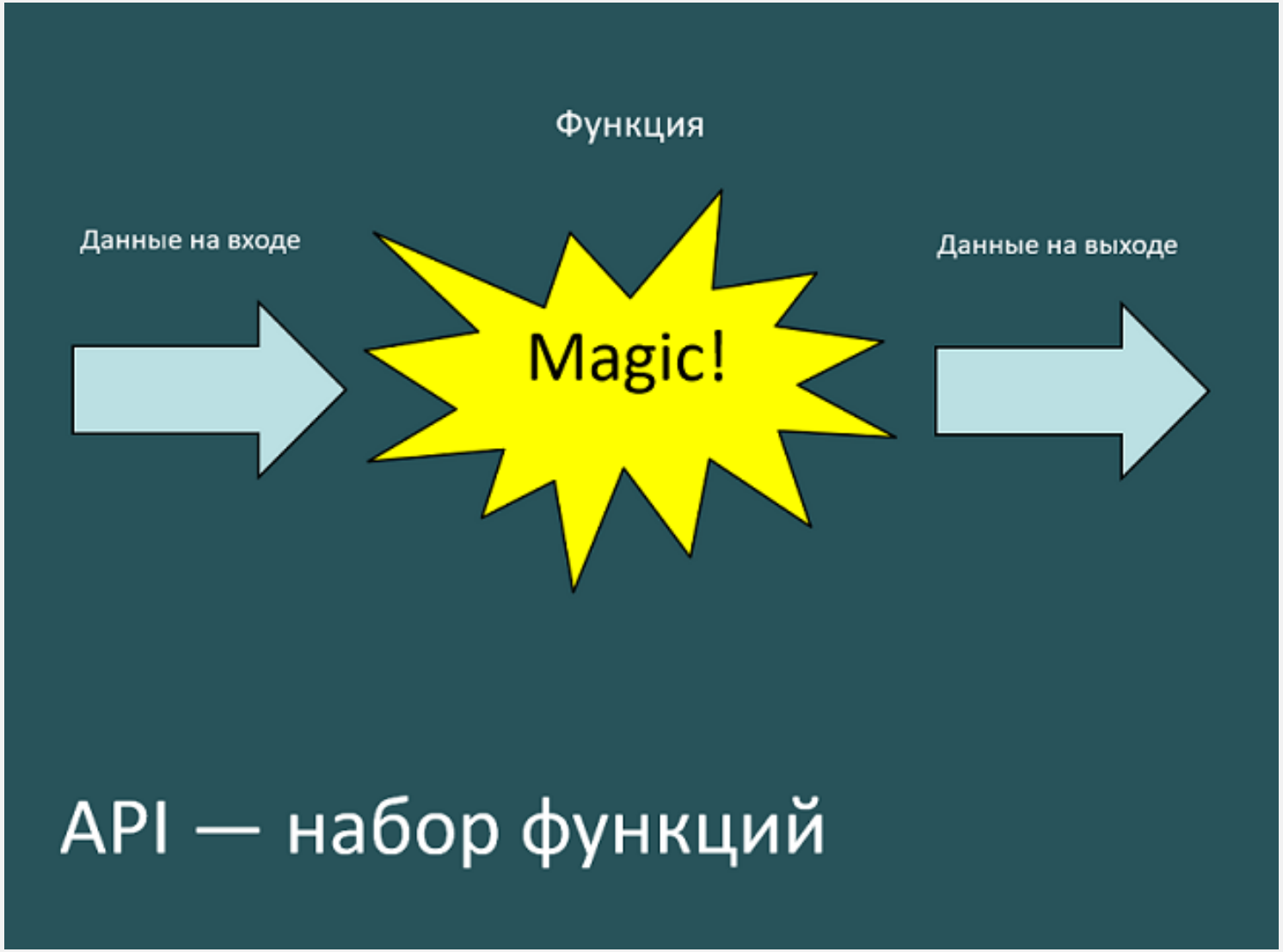
Обычно, выделяют 2 подхода:

- Code first — сначала пишем код, потом по нему генерируем контракт
- Contract first — сначала создаем контракт, потом по нему пишем или генерируем код

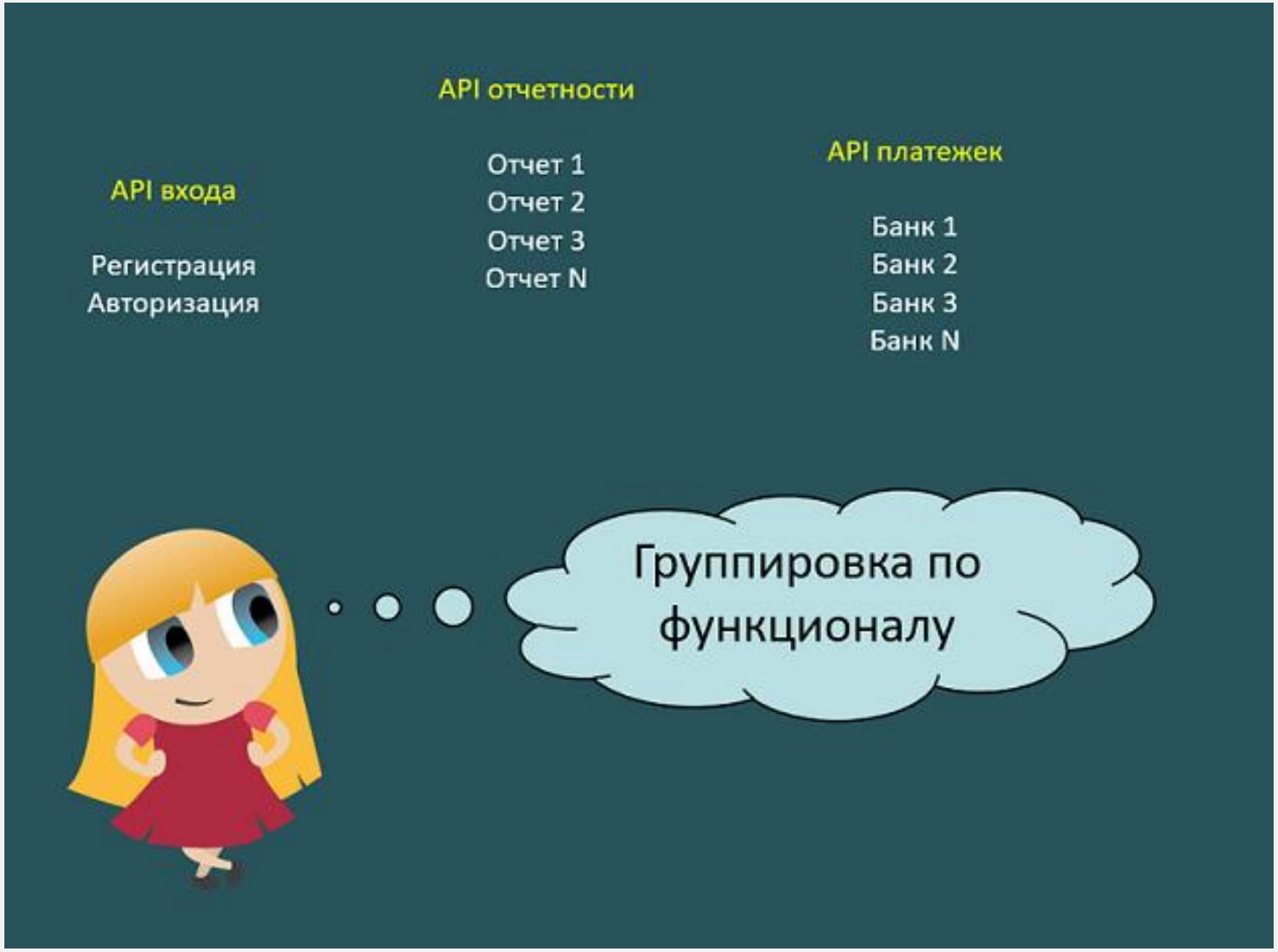
API


Ко мне вы можете обращаться вот так...
API включает:
— операция;
— данные на входе;
— данные на выходе.

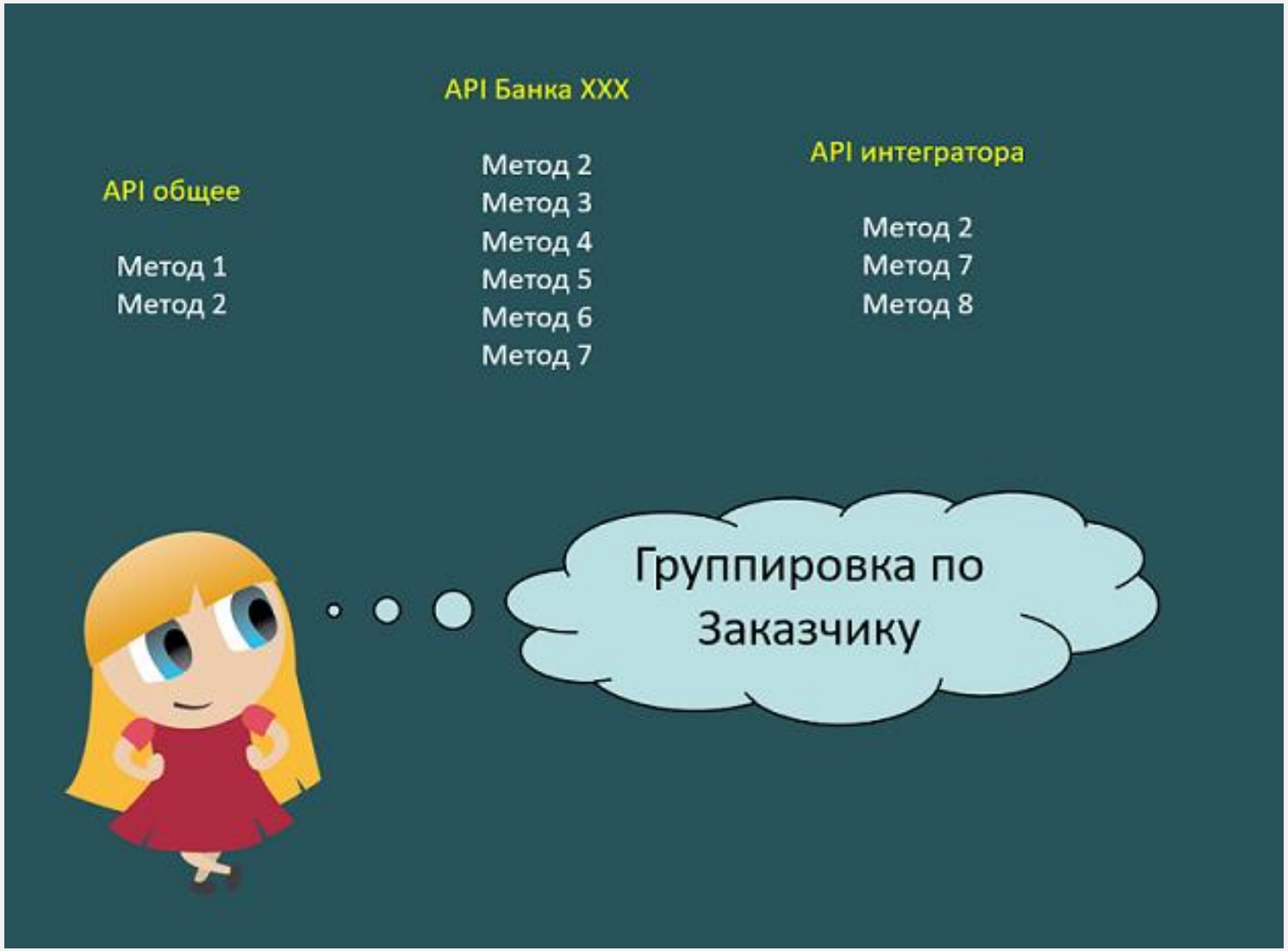
API


Функция
Данные на входе
Данные на выходе
Magic!
API — набор функций

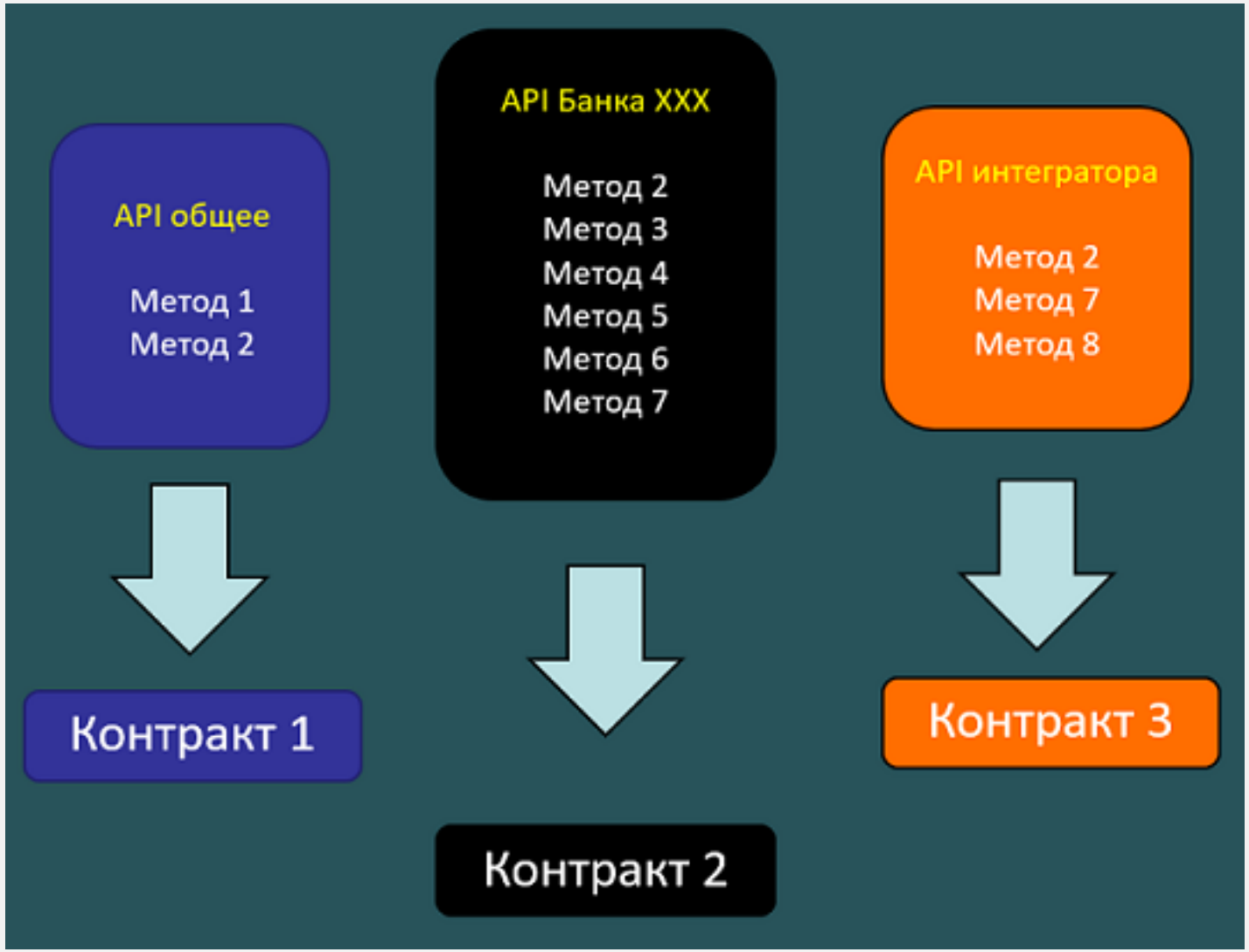
API


API отчетности
API входа
Регистрация
Авторизация
Отчет 1
Отчет 2
Отчет 3
Отчет N
API платежек
Банк 1
Банк 2
Банк 3
Банк N
Группировка по функционалу

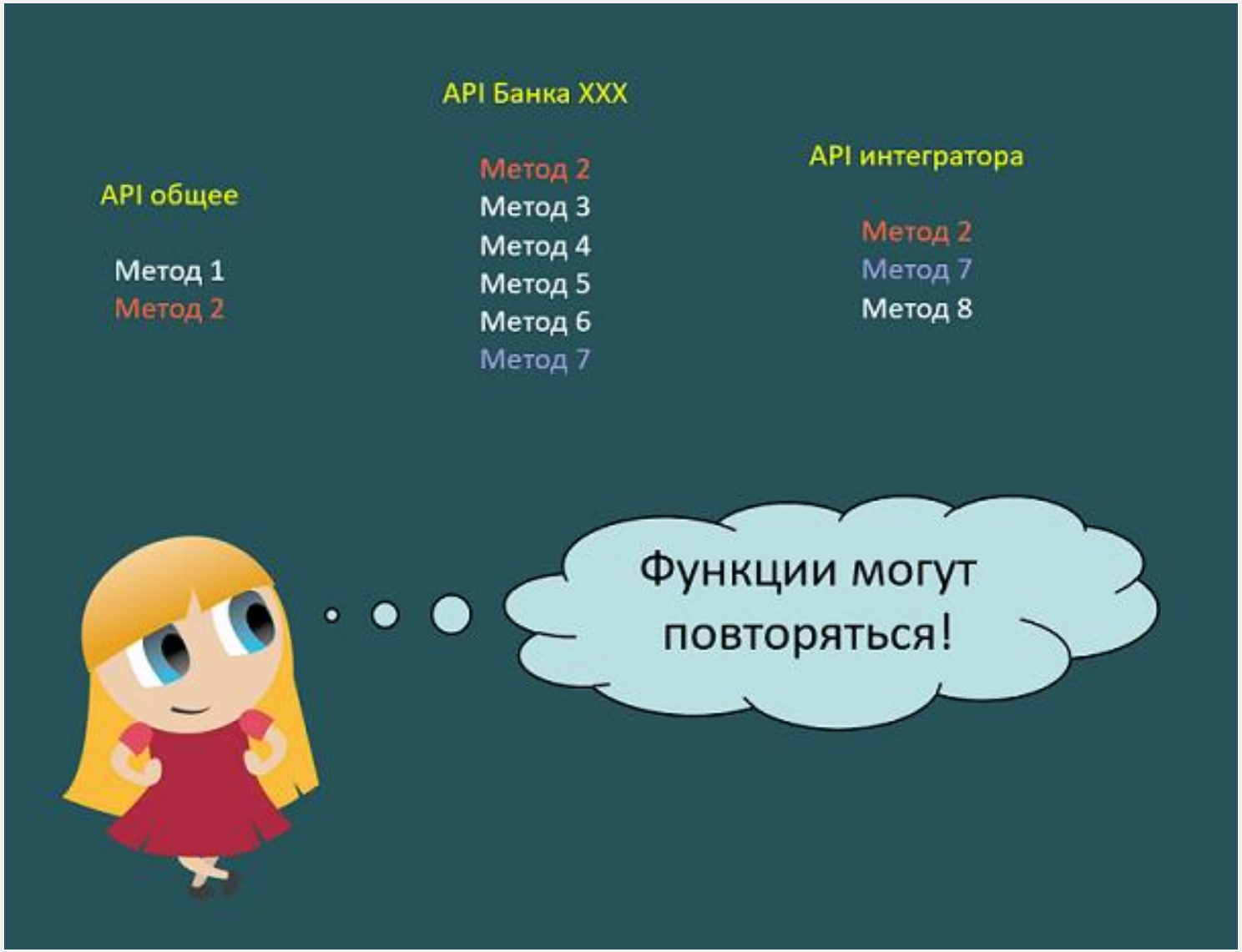
API


API общее
Метод 1
Метод 2
API Банка XXX
Метод 2
Метод 3
Метод 4
Метод 5
Метод 6
Метод 7
API интегратора
Метод 2
Метод 7
Метод 8
Группировка по Заказчику

API


API общее
Метод 1
Метод 2
API Банка XXX
Метод 2
Метод 3
Метод 4
Метод 5
Метод 6
Метод 7
API интегратора
Метод 2
Метод 7
Метод 8
Контракт 1
Контракт 2
Контракт 3

API


API общее
Метод 1
Метод 2
API Банка XXX
Метод 2
Метод 3
Метод 4
Метод 5
Метод 6
Метод 7
API интегратора
Метод 2
Метод 7
Метод 8
Функции могут повторяться!

API

Вызвать апи можно как **напрямую**, так и **косвенно**.

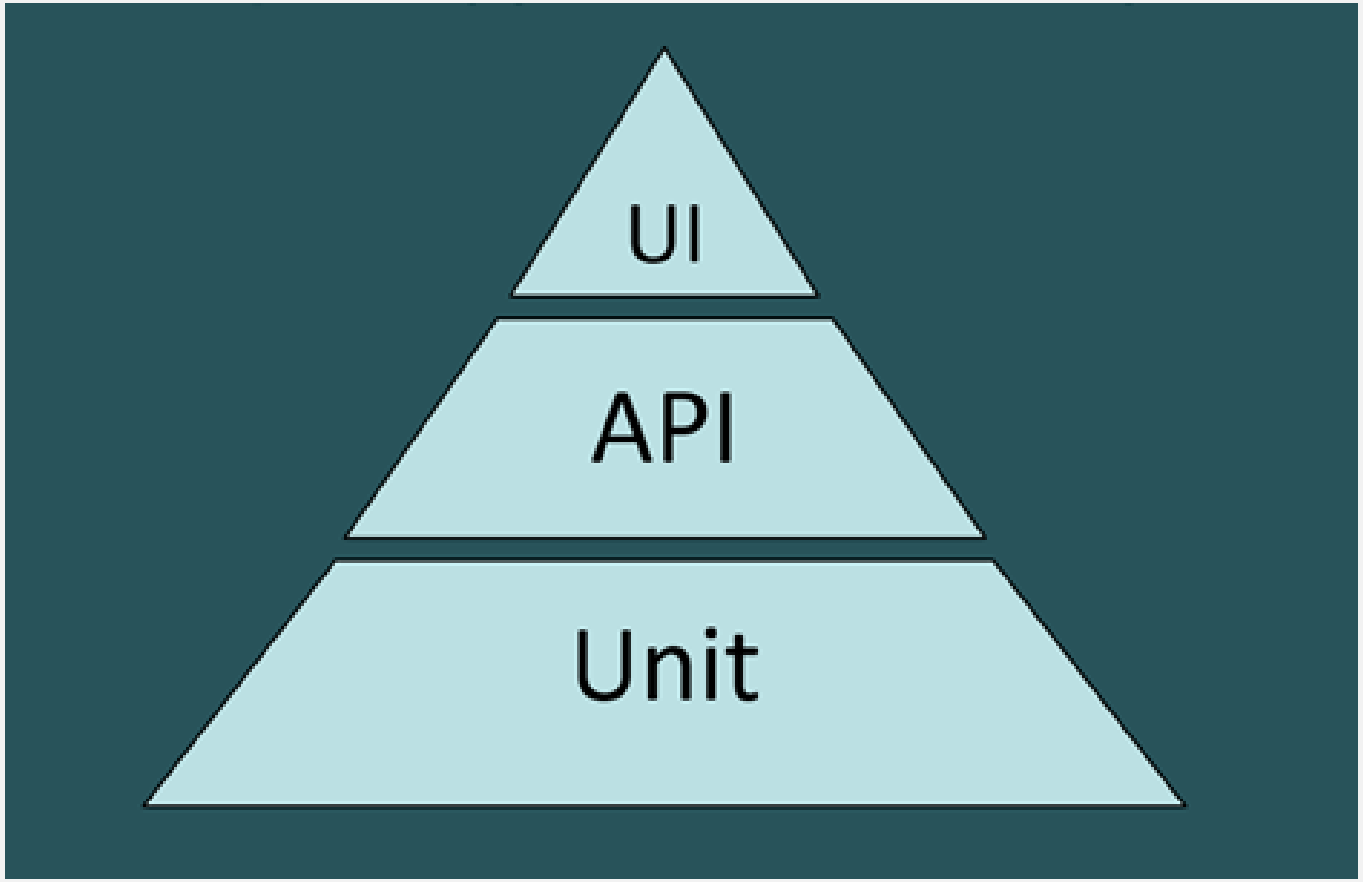
Напрямую:

1. Система вызывает функции внутри себя
2. Система вызывает метод другой системы
3. Человек вызывает метод
4. Автотесты дергают методы

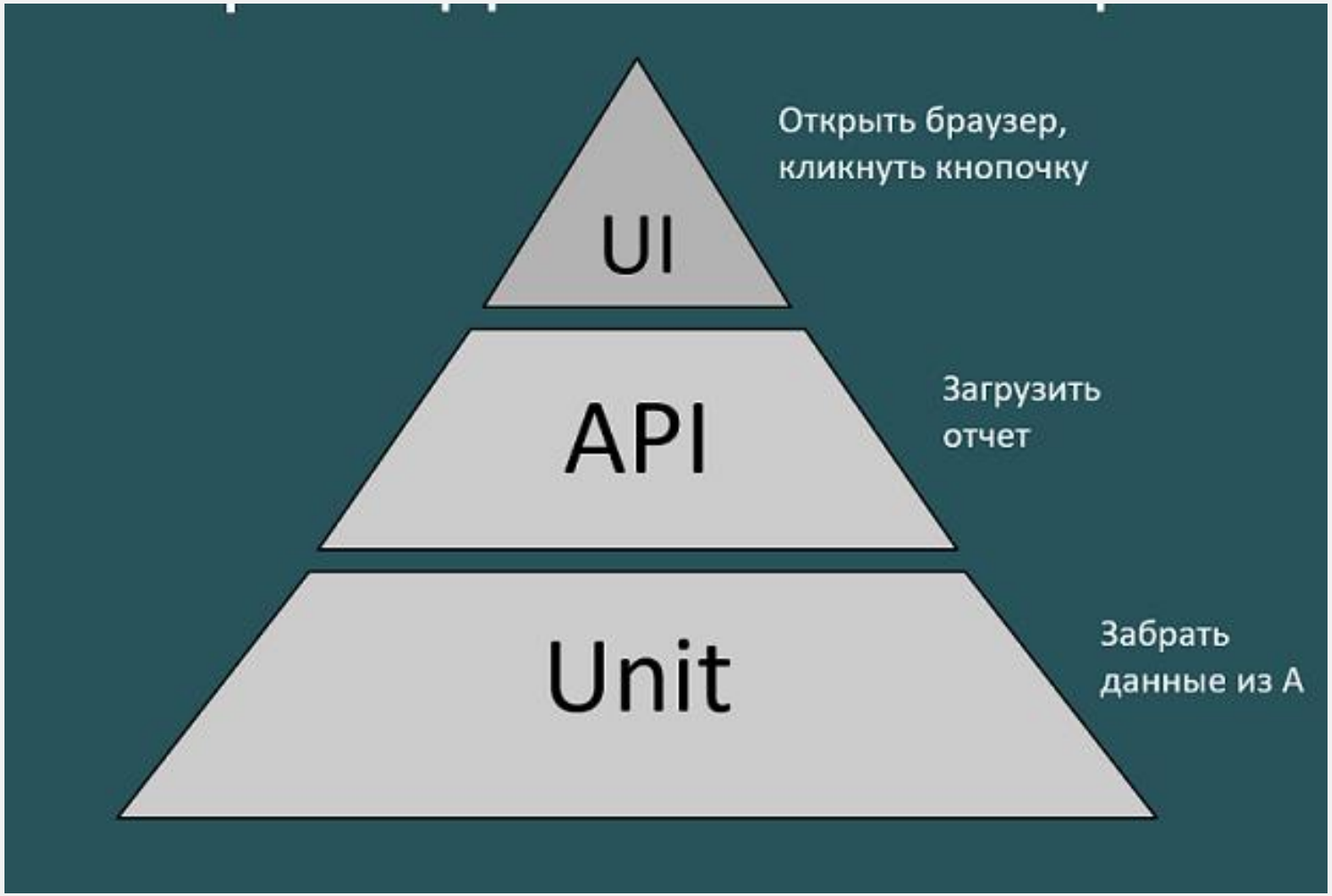
Косвенно:

1. Пользователь работает с GUI

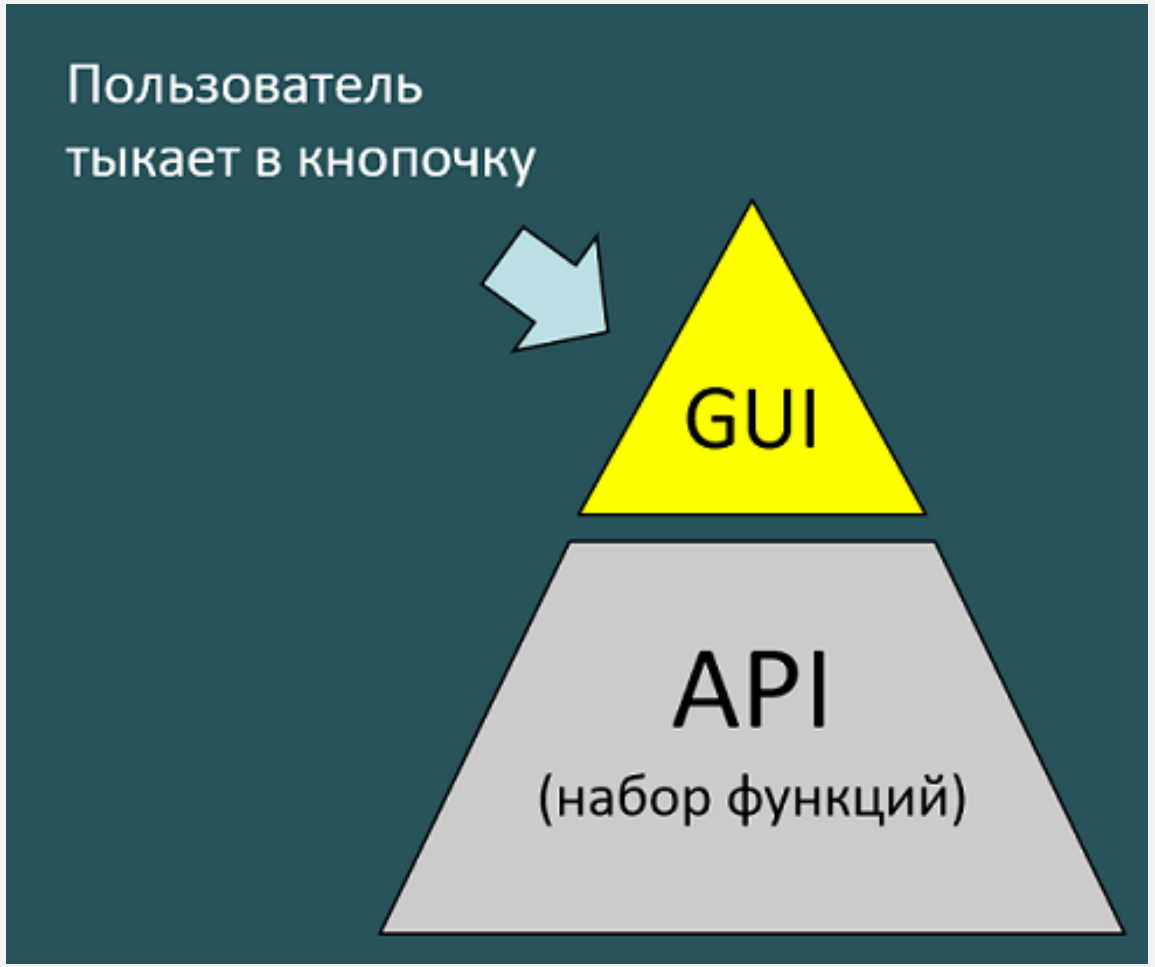
API. ТЕСТИРОВАНИЕ

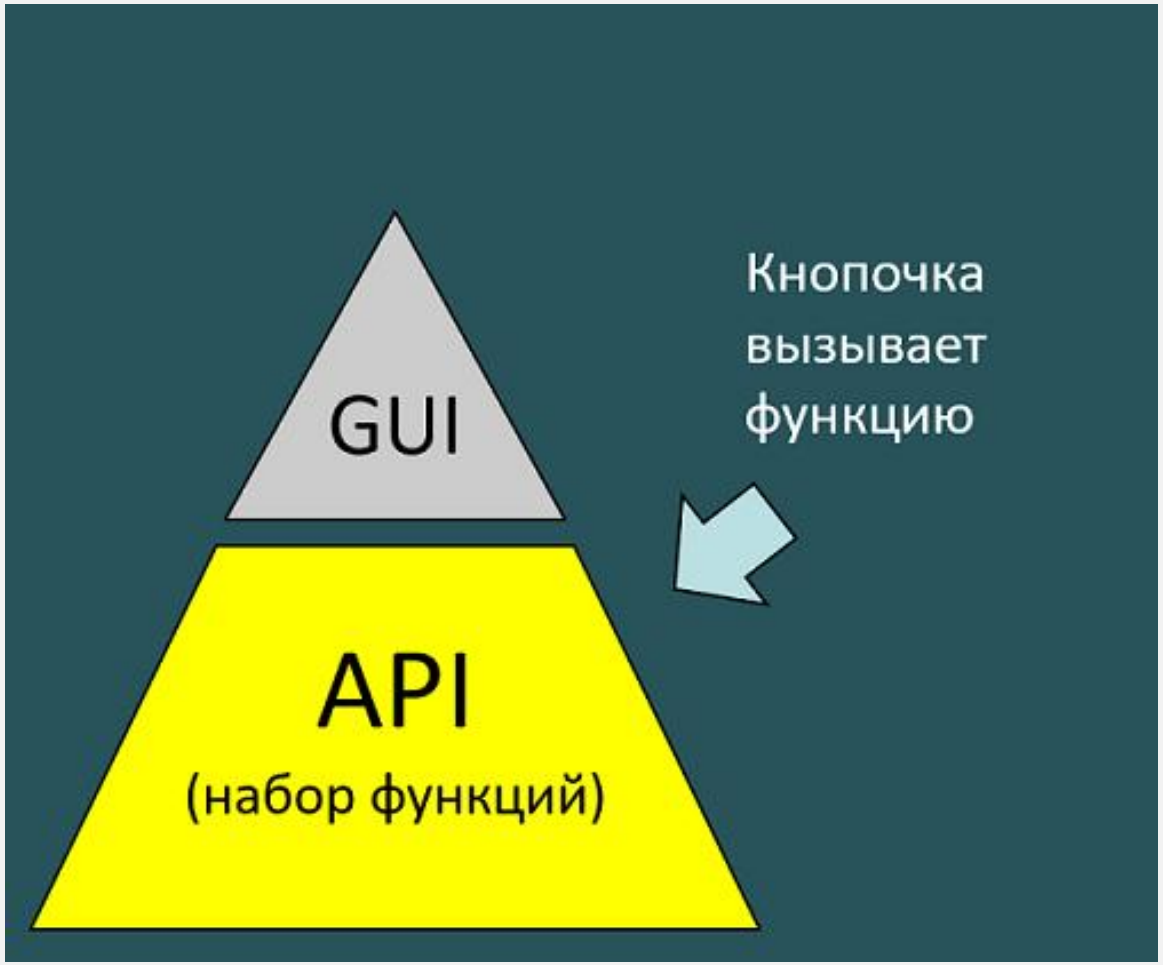


API. ТЕСТИРОВАНИЕ



API. ТЕСТИРОВАНИЕ





REST

Отличительной особенностью сервисов **REST** является то, что они позволяют наилучшим образом использовать протокол **HTTP**.

При выстраивании API в соответствии с принципами REST, мы оперируем с ресурсами, например:

Создать пользователя: **POST /users**
Удалить пользователя по ID: **DELETE /users/1**
Получить всех пользователей: **GET /users**
Получить пользователя по ID: **GET /users/1**

REST

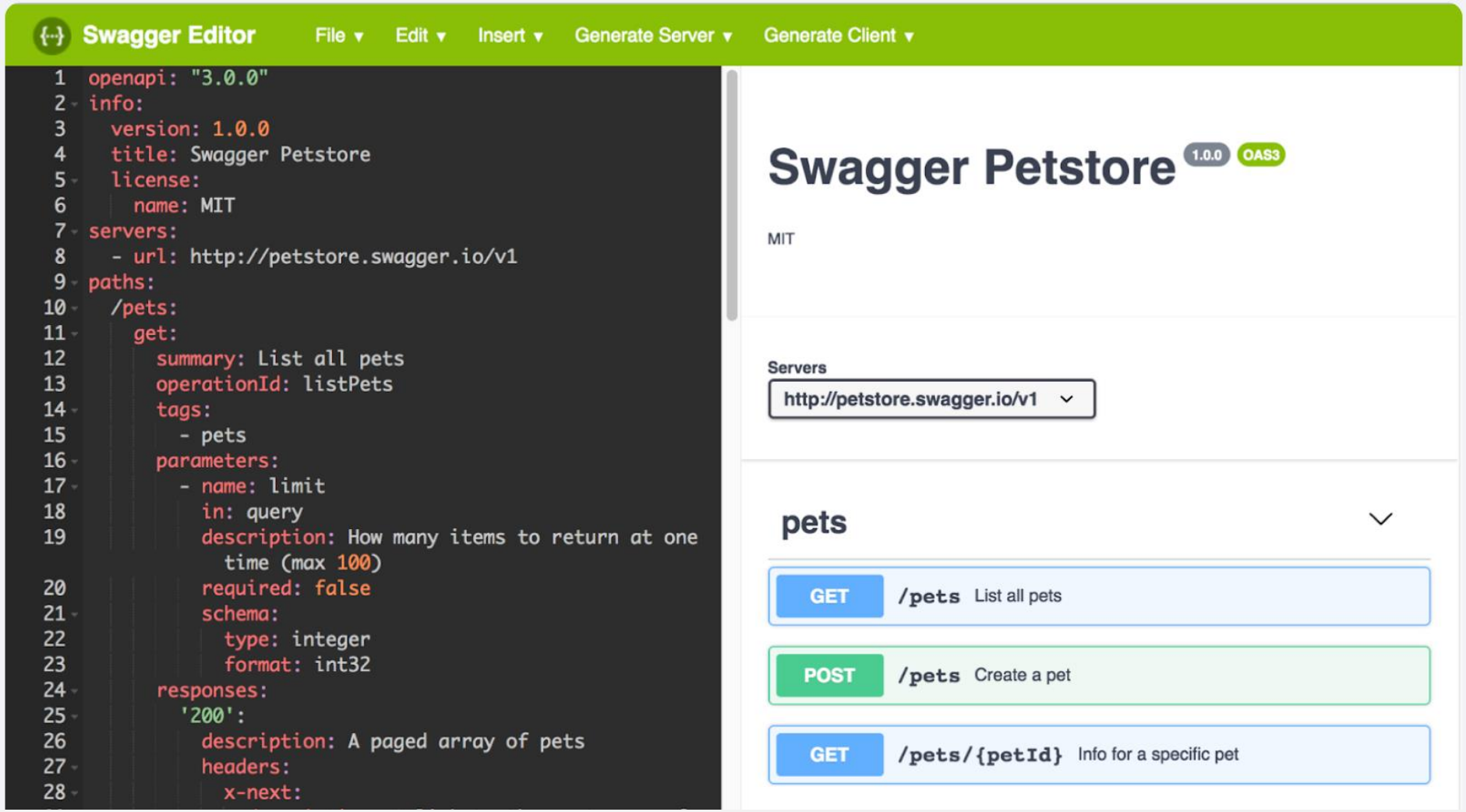
Вот как обычно реализуется служба REST:

Формат обмена данными: JSON — наиболее популярный формат. Данные передаются в телах http-запросов и http-ответов

Транспорт: всегда HTTP. REST полностью построен на основе HTTP, протоколе прикладного уровня.

Определение сервиса: не существует стандарта для этого, а REST является гибким. Это может быть недостатком в некоторых сценариях, поскольку потребляющему приложению может быть необходимо понимать форматы запросов и ответов. Однако широко используются Swagger (OpenAPI).

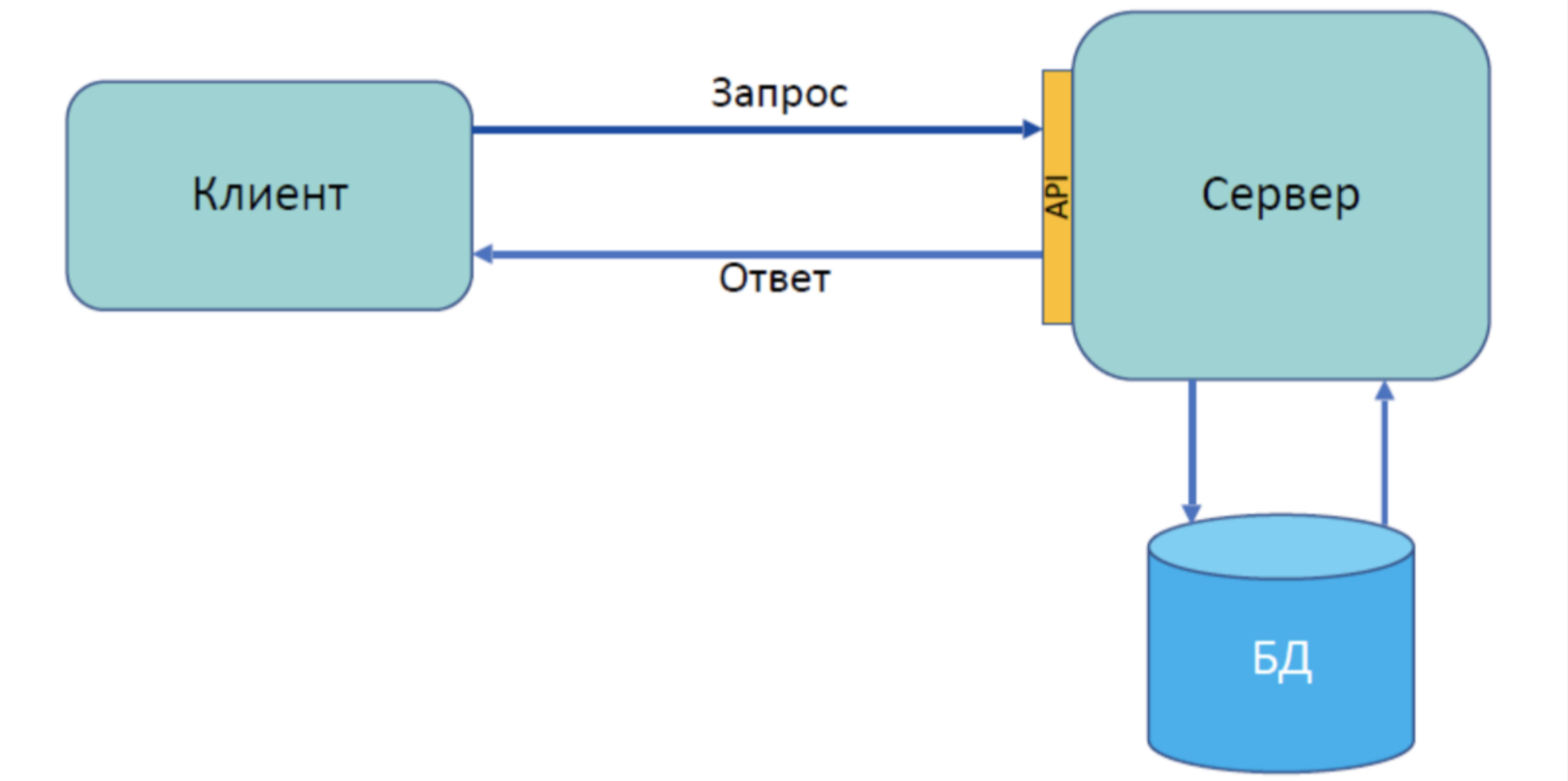
REST


Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
openapi: "3.0.0"
info:
version: 1.0.0
title: Swagger Petstore
license:
name: MIT
servers:
- url: http://petstore.swagger.io/v1
paths:
/pets:
get:
summary: List all pets
operationId: listPets
tags:
- pets
parameters:
- name: limit
in: query
description: How many items to return at one time (max 100)
required: false
schema:
type: integer
format: int32
responses:
'200':
description: A paged array of pets
headers:
x-next:
Swagger Petstore
1.0.0
OAS3
MIT
Servers
http://petstore.swagger.io/v1
pets
GET /pets List all pets
POST /pets Create a pet
GET /pets/{petId} Info for a specific pet

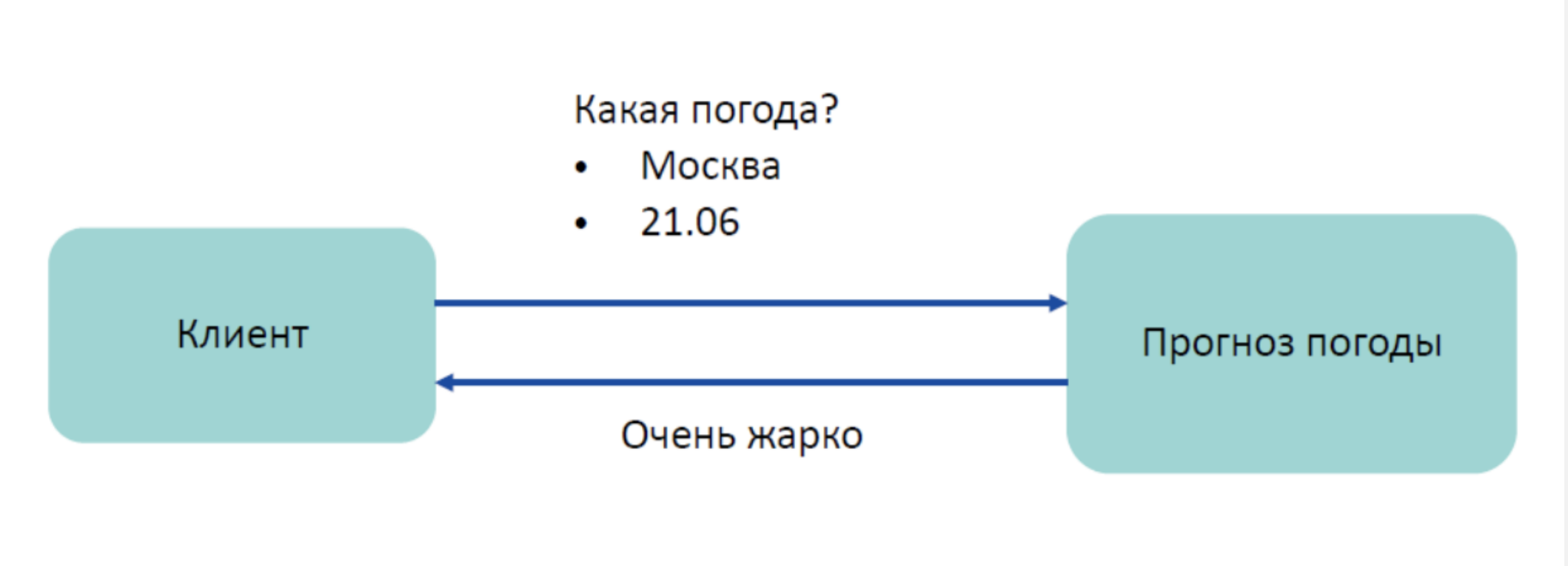
REST. ПРИНЦИПЫ

1. Клиент-серверная архитектура
2. Stateless
3. Кэширование
4. Единообразие интерфейса
5. Layered system
6. Code on demand

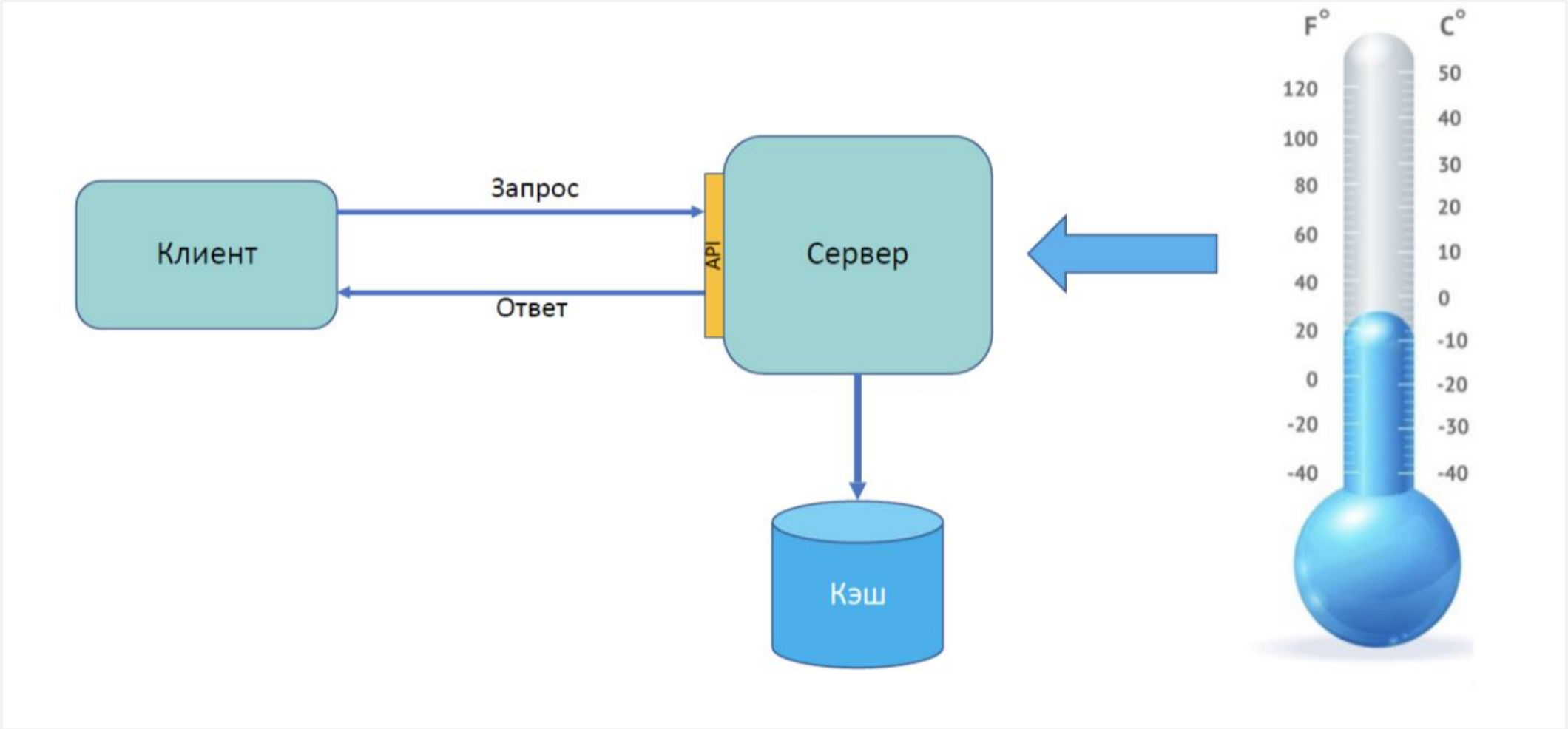
REST. КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА



REST. STATELESS



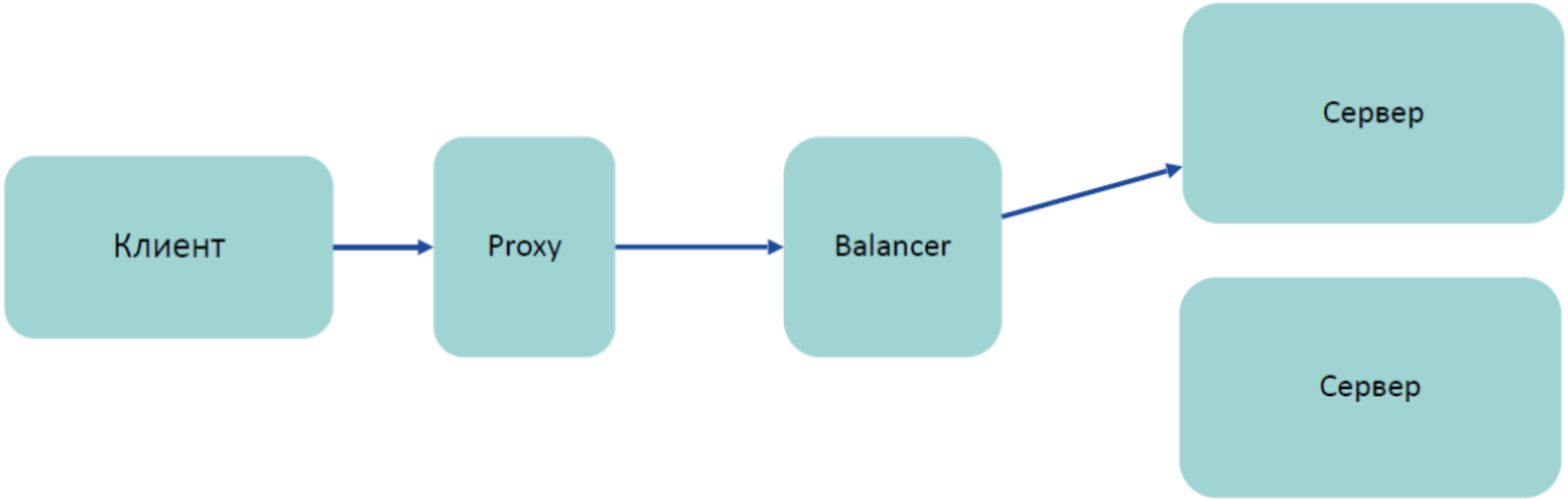
REST. КЭШИРОВАНИЕ



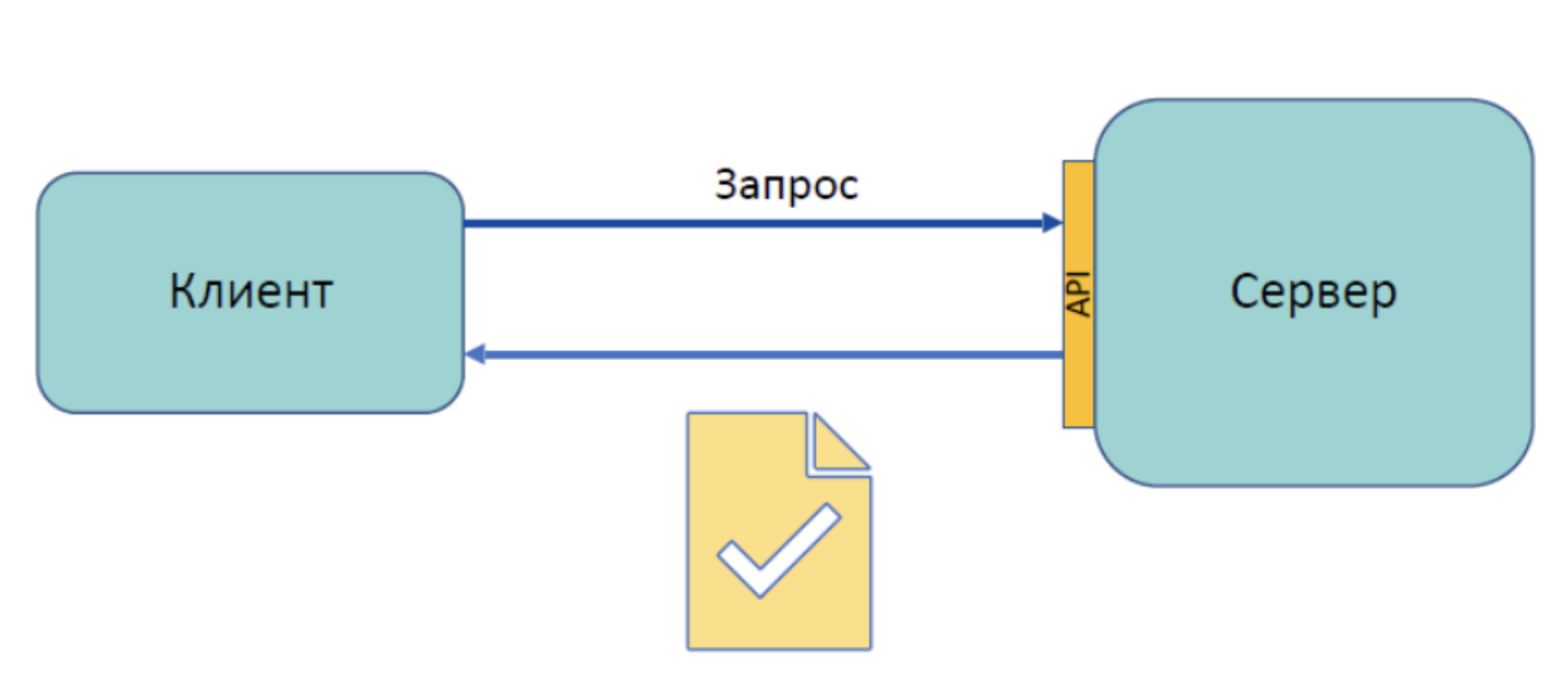
REST. ЕДИНООБРАЗИЕ ИНТЕРФЕЙСА

Создать пользователя: **POST /users**
Удалить пользователя по ID: **DELETE /users/1**
Получить всех пользователей: **GET /users**
Получить пользователя по ID: **GET /users/1**

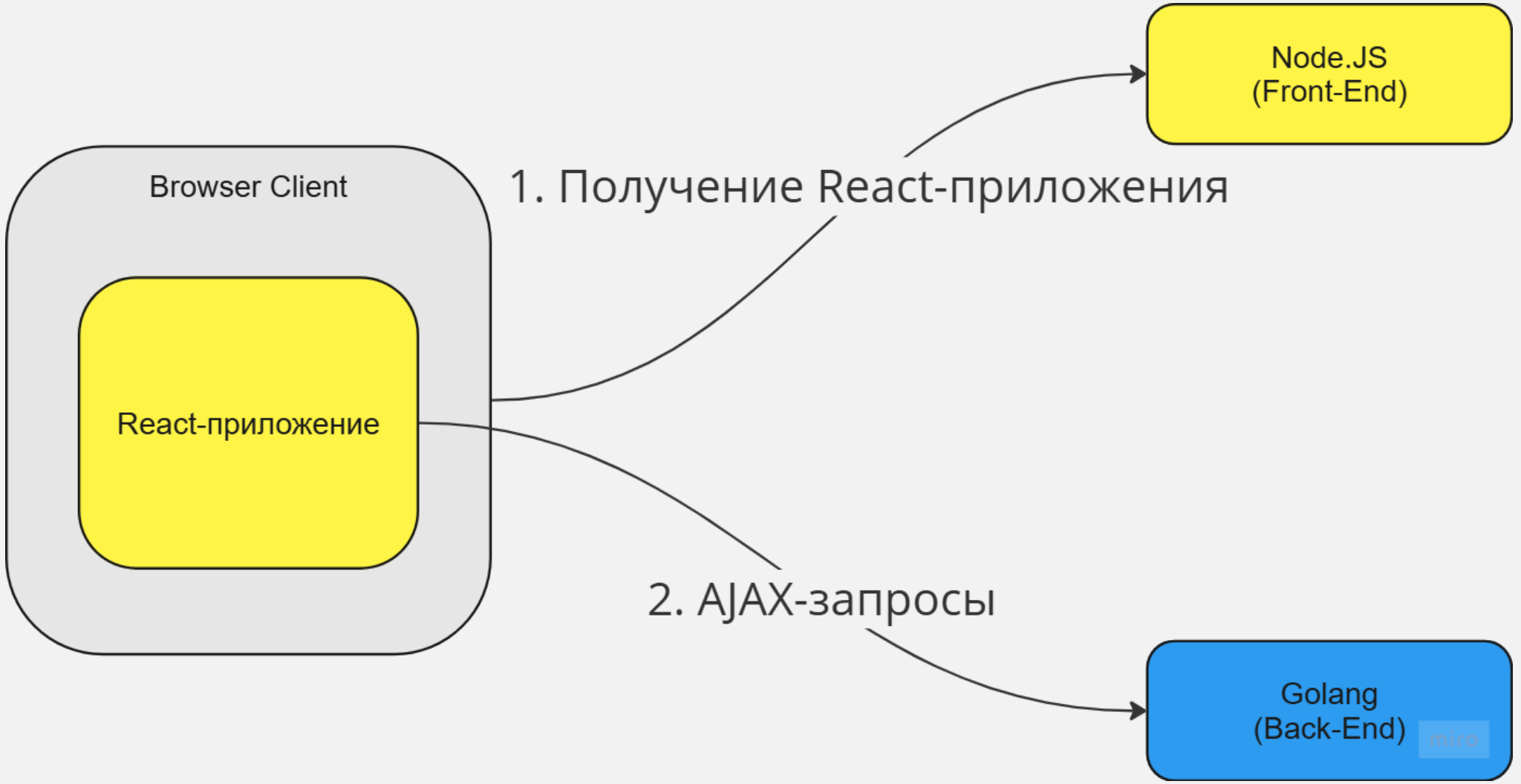
REST. LAYERED SYSTEM



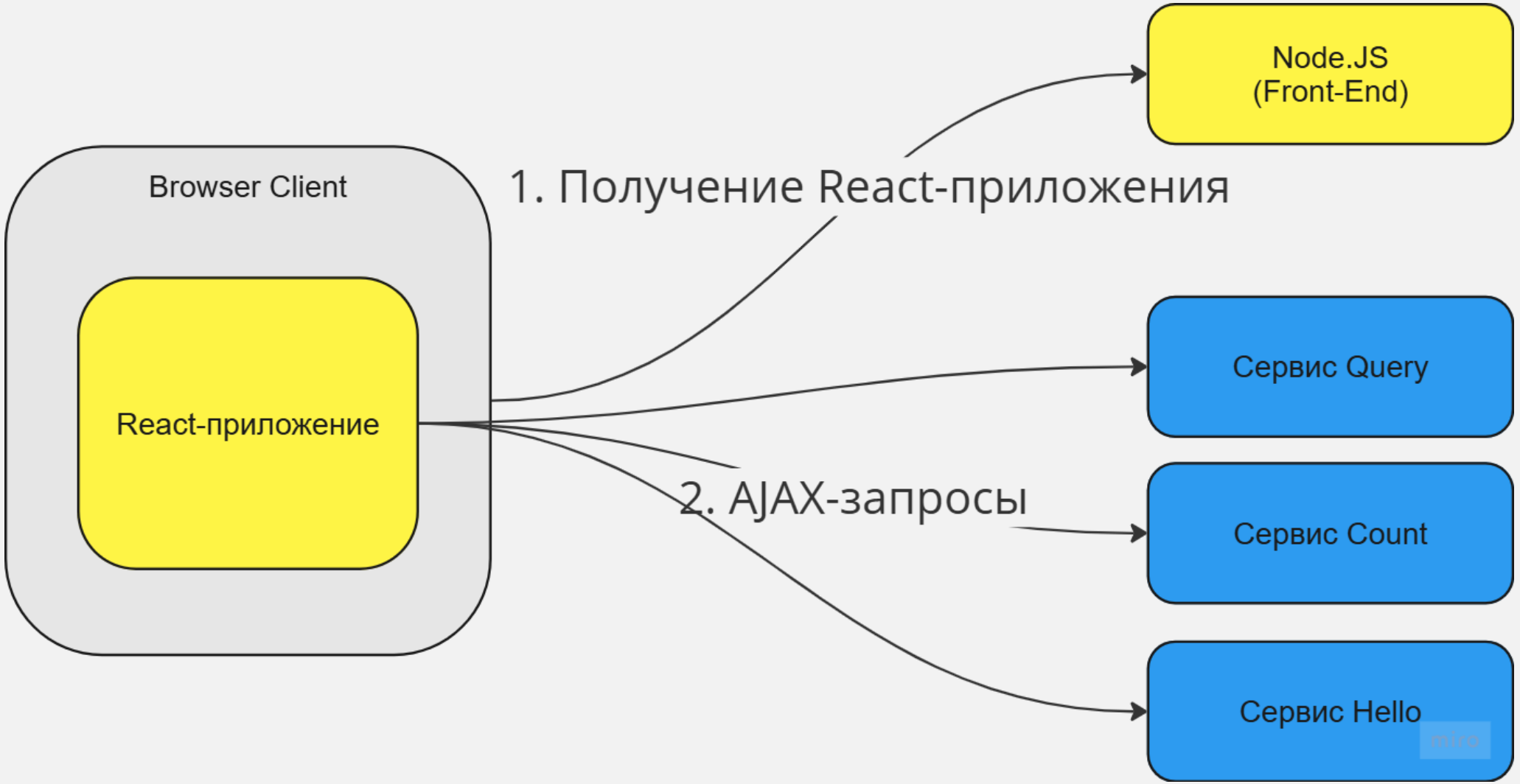
REST. CODE ON DEMAND



REST. CODE ON DEMAND



REST. КАК ЭТО ВЫГЛЯДИТ В 7-Й ЛАБЕ



ИСТОЧНИКИ

- **Введение в REST API — RESTful веб-сервисы**
  https://habr.com/ru/articles/483202/
- **Что такое API**
  https://habr.com/ru/articles/464261/
- **REST, что же ты такое?**
  https://habr.com/ru/articles/590679/

СПАСИБО ЗА ВНИМАНИЕ :3

12.11.2024