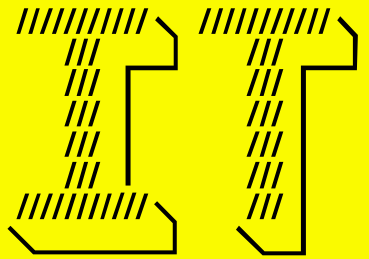
REST
of the best

Райффайзен



Всем привет

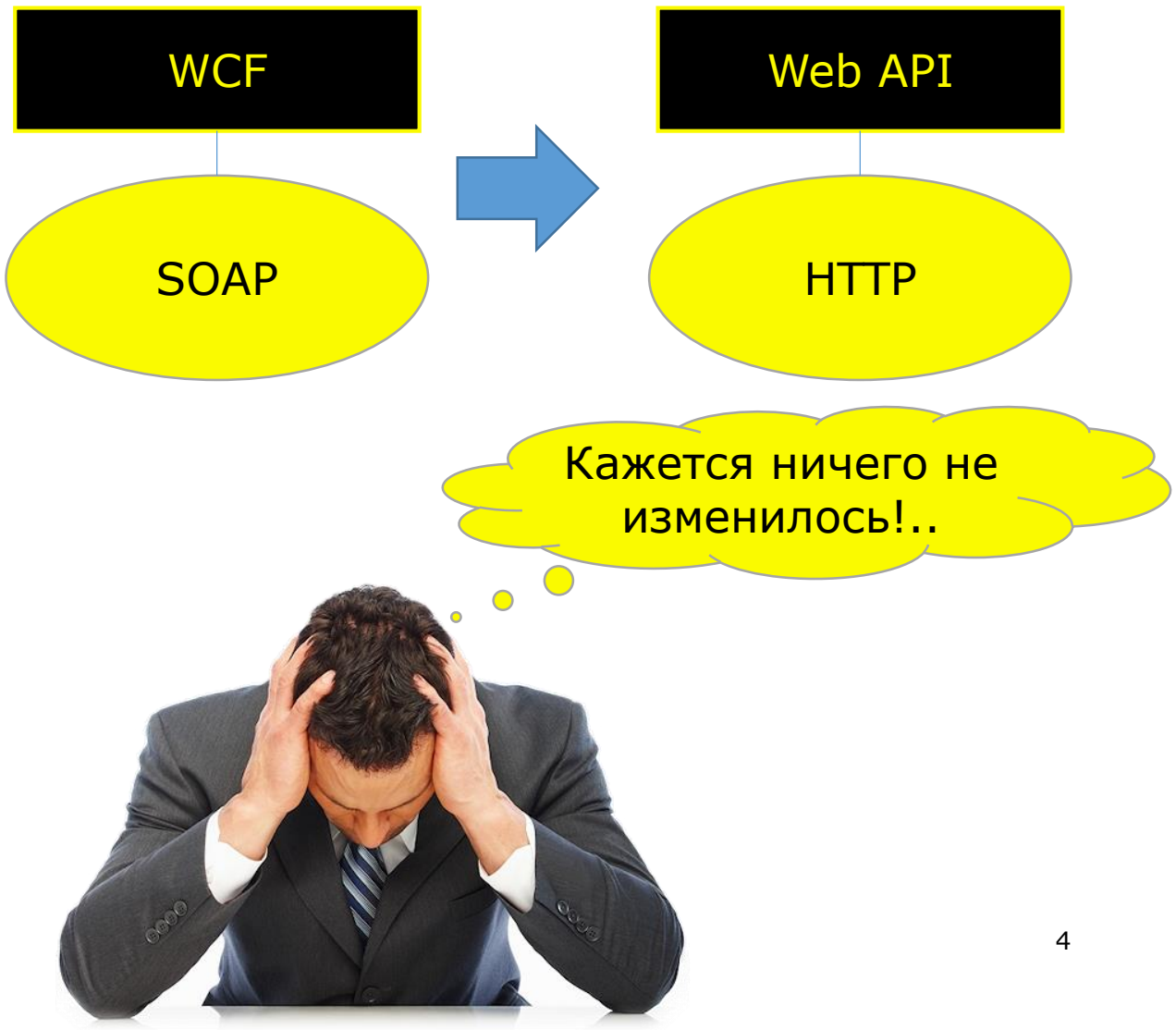
- Меня зовут Просин Роман.
- Мой путь разработчика начался более 7 лет назад
- 2 года я работаю разработчиком в ICDB Team Райффайзенбанка.
- Мы строим фронт для отделений, сервисы для других систем банка, меняем архитектуру старых сервисов.
- Помимо работы я люблю активный отдых, водить автомобиль, Айкидо и многое другое…

REST - что это?

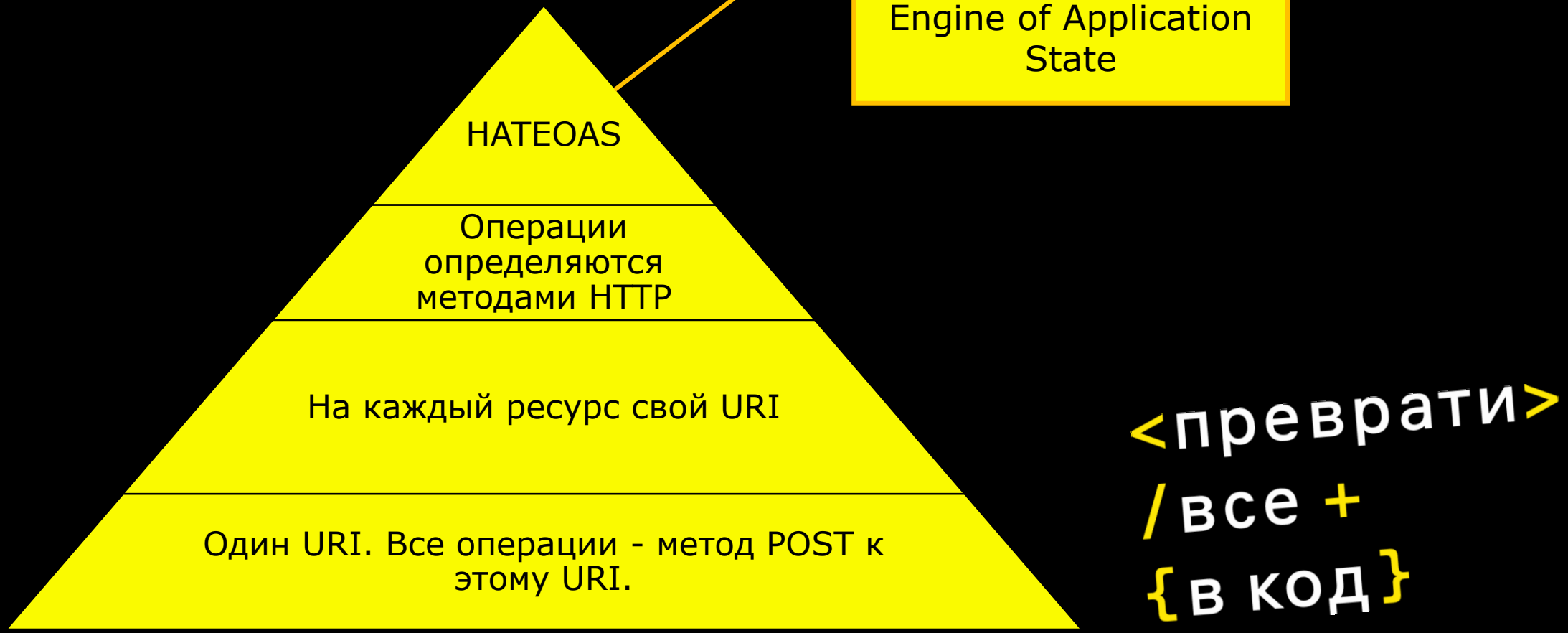
- Representational State Transfer
- It is **architecture style**
- Свойства:
  - Производительность
  - Масштабируемость
- Ограничения:
  - Клиент-сервер
  - Отсутствие состояния
  - Кэширование
  - Единообразие интерфейса
  - Слои
  - Код по требованию

Цель

```
[RoutePrefix("api/locations")]
ссылок: 2 | 0 изменений | 0 авторов, 0 изменений
public class LocationsService : ApiController, ILocations
{
  [Route("create")] [HttpPost]
  ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
  public int CreateLocation(CreateLocationRequest request)...

  [Route("{id:int}/update-name")][HttpPost]
  ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
  public void UpdateLocationName(int id, string name)...

  [Route("{id:int}/delete")][HttpPost]
  ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
  public void DeleteLocation(int id)...

  [Route("{id:int}/update-complexity")][HttpPost]
  ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
  public void UpdateLocationComplexity(int id, [FromUri]int complexity)...

  [Route("{id:int}/get")][HttpGet]
  ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
  public Game[] GetGames(int id)...

  [Route("{id:int}/getAll")][HttpGet]
  ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
  public Location[] GetLocations()...
}
```



Модель зрелости RESTful API



<превратим>
/все +
{в код}

По порядку

1

Архитектура

Основные концепции архитектуры RESTful сервисов.

2

Лучшие практики

Как достичь зрелости?

3

Checklist

Список факторов, которые следует учитывать при разработке и реализации веб-API

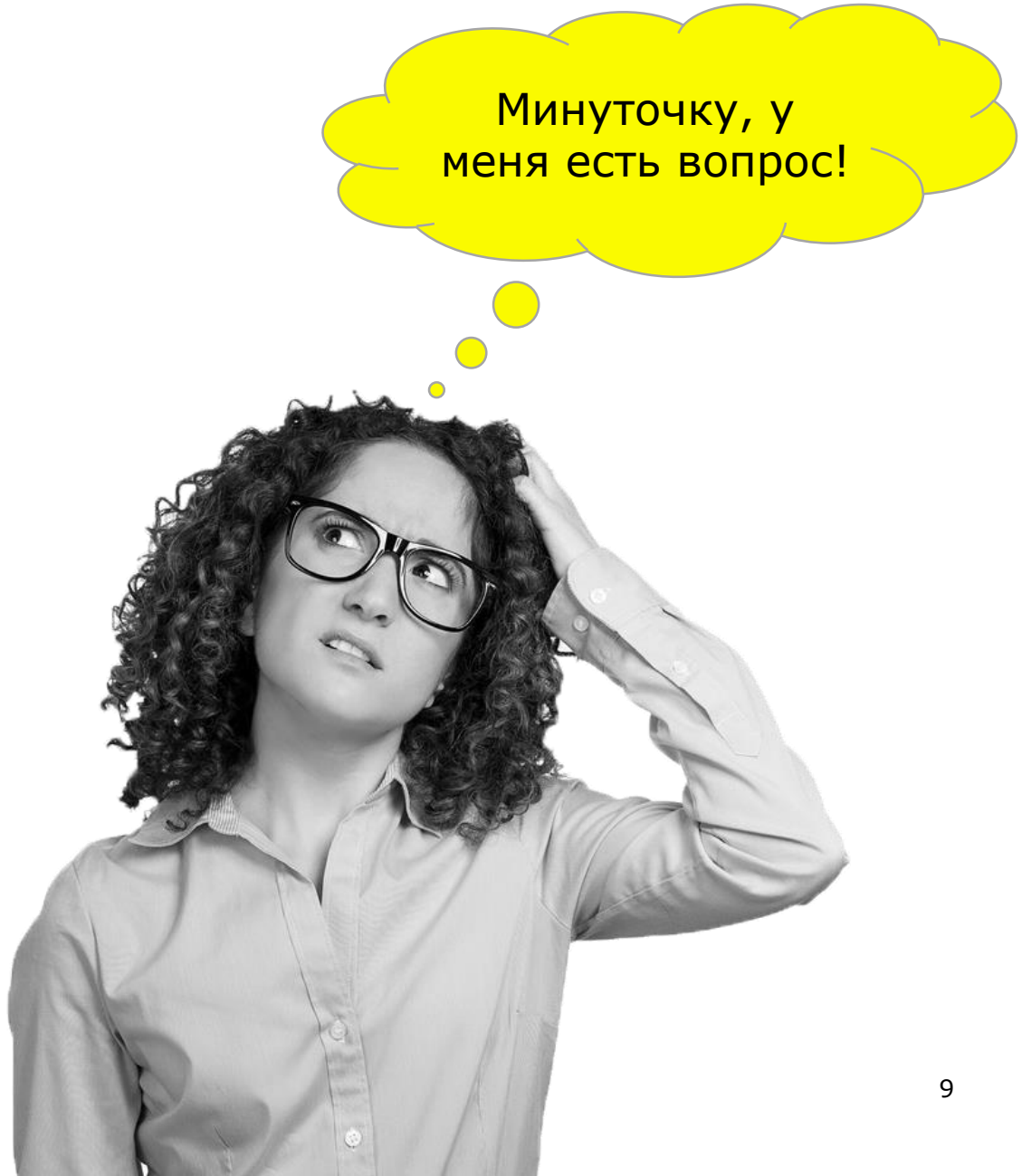
4

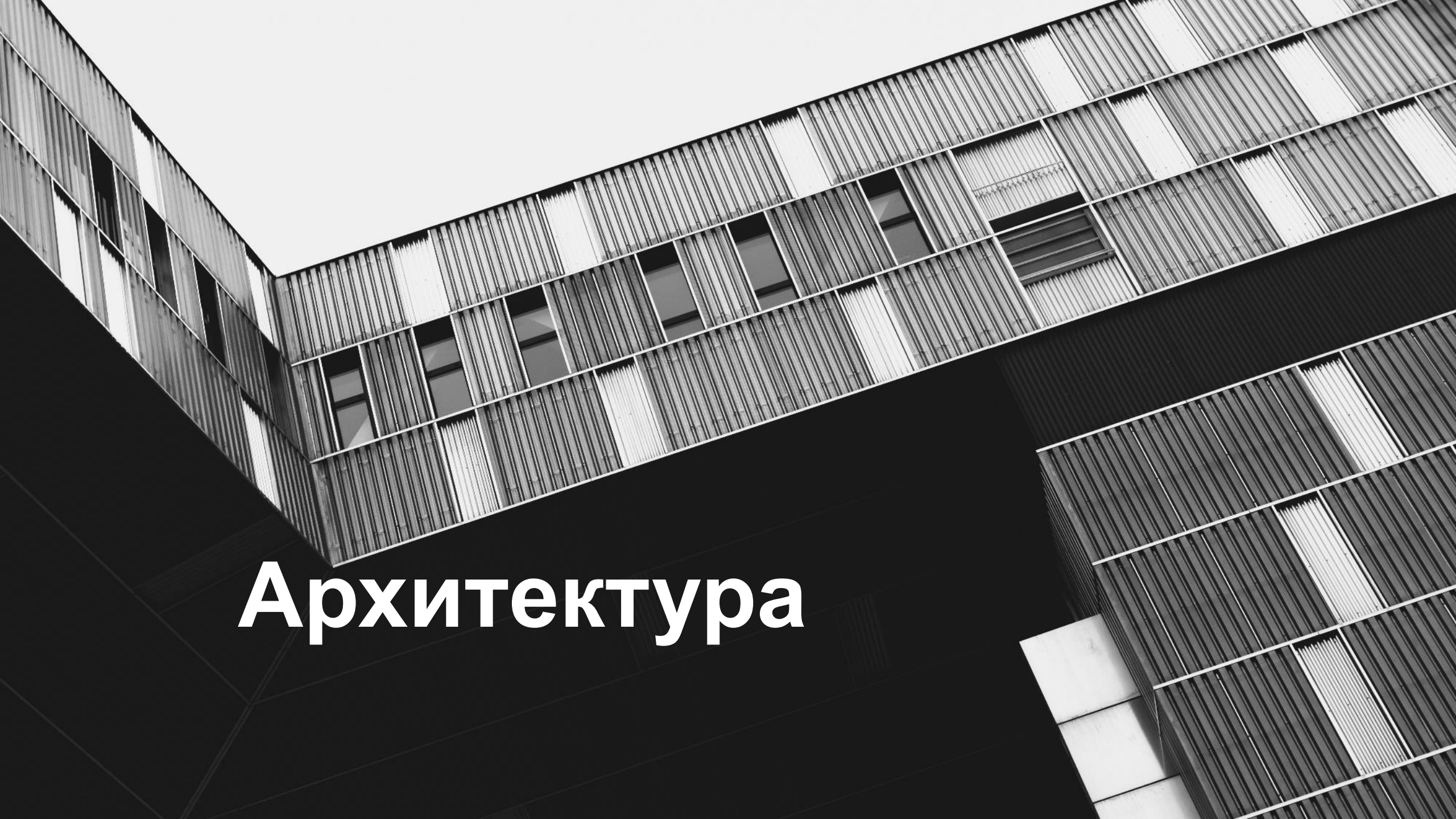
Open API

Возвращаемся к спецификации.
Генерируем клиентский код.
Документация всегда под рукой.

5

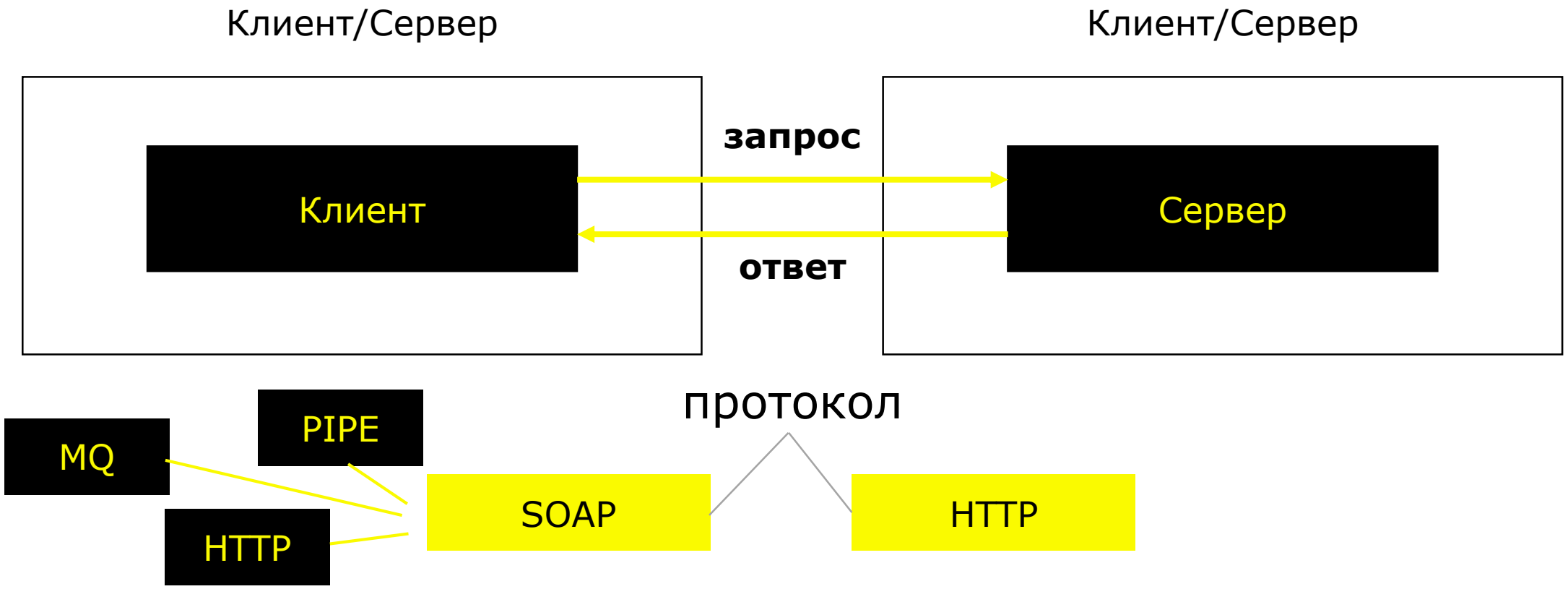
Подведём итог



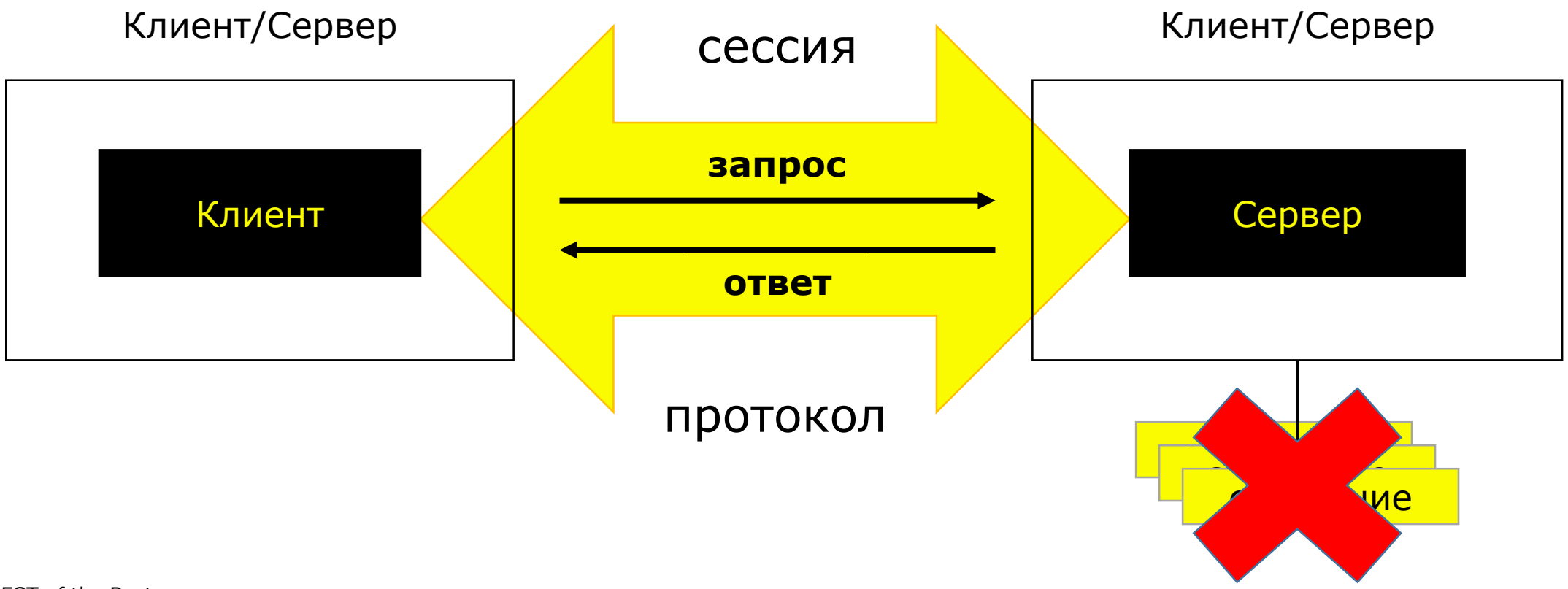


Архитектура

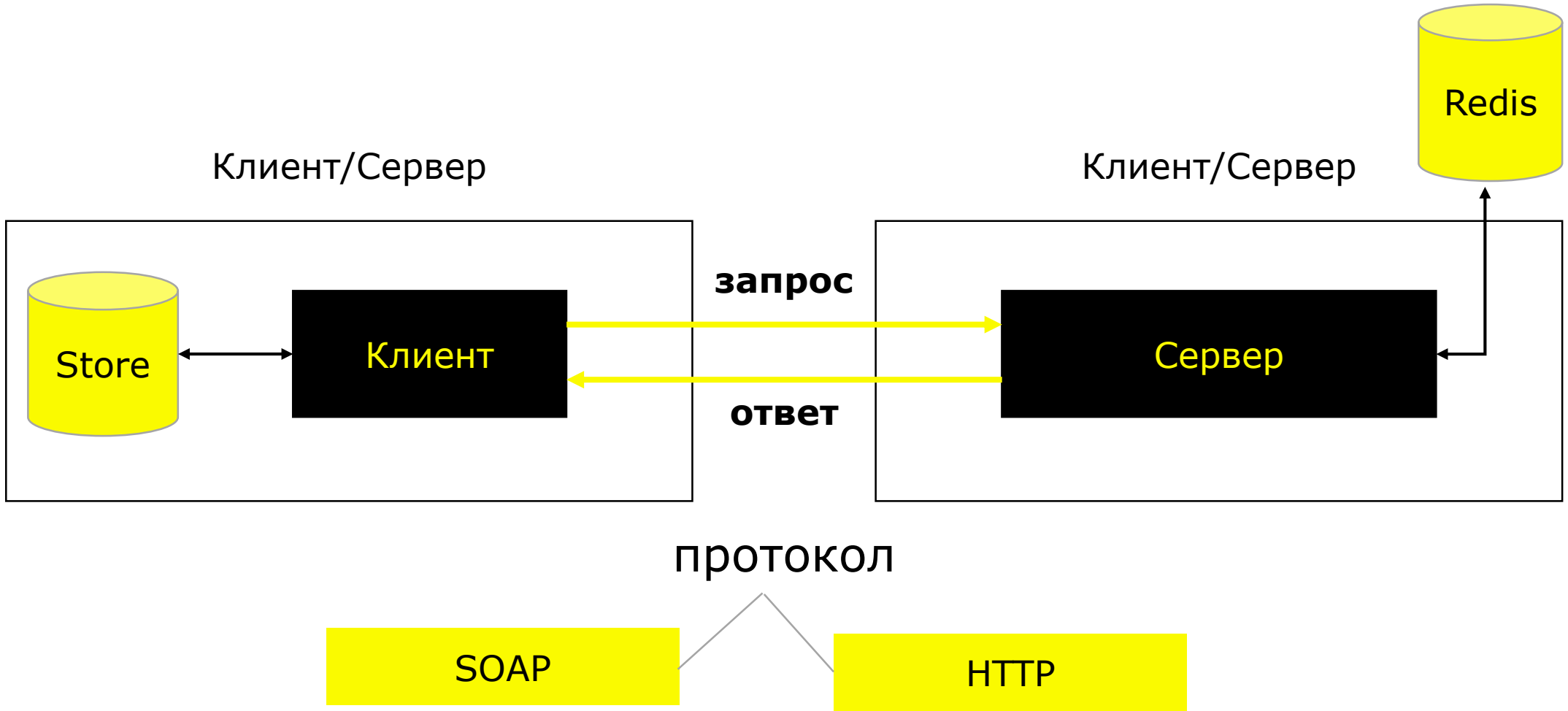
Клиент-Протокол-Сервер



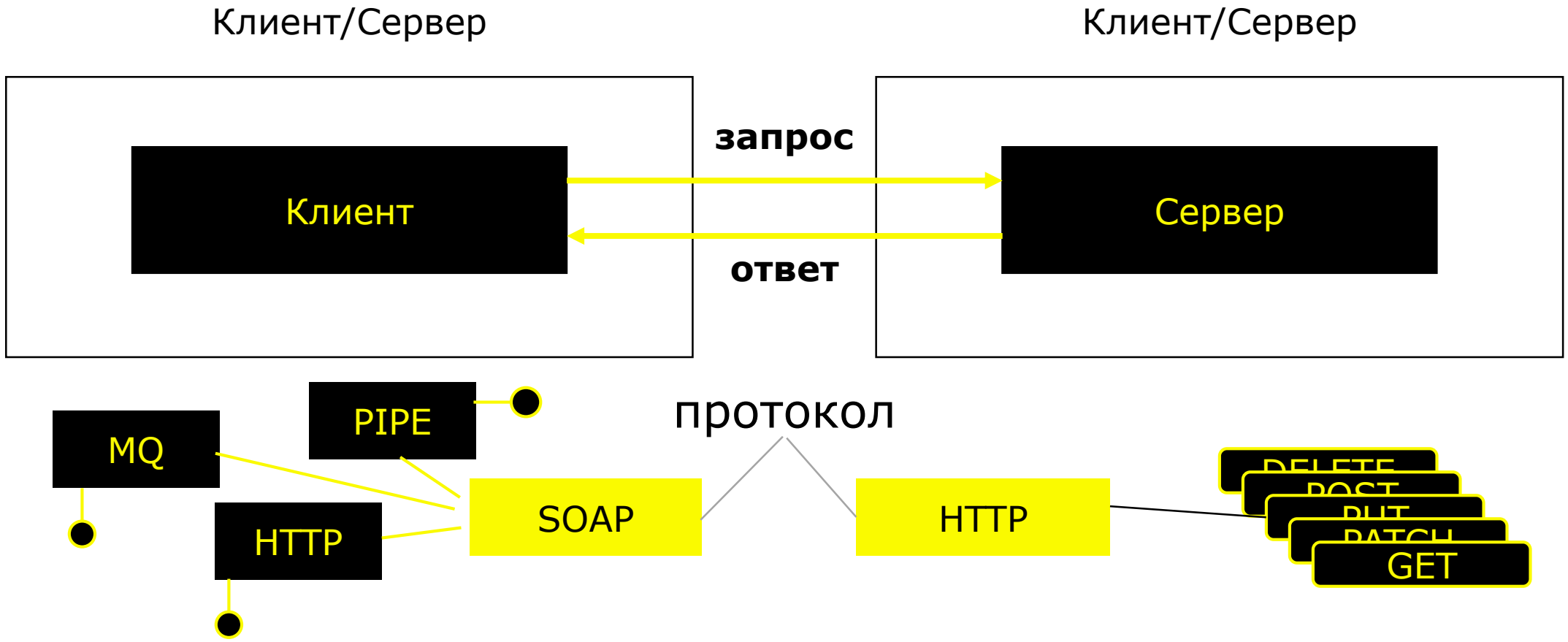
Отсутствие состояния



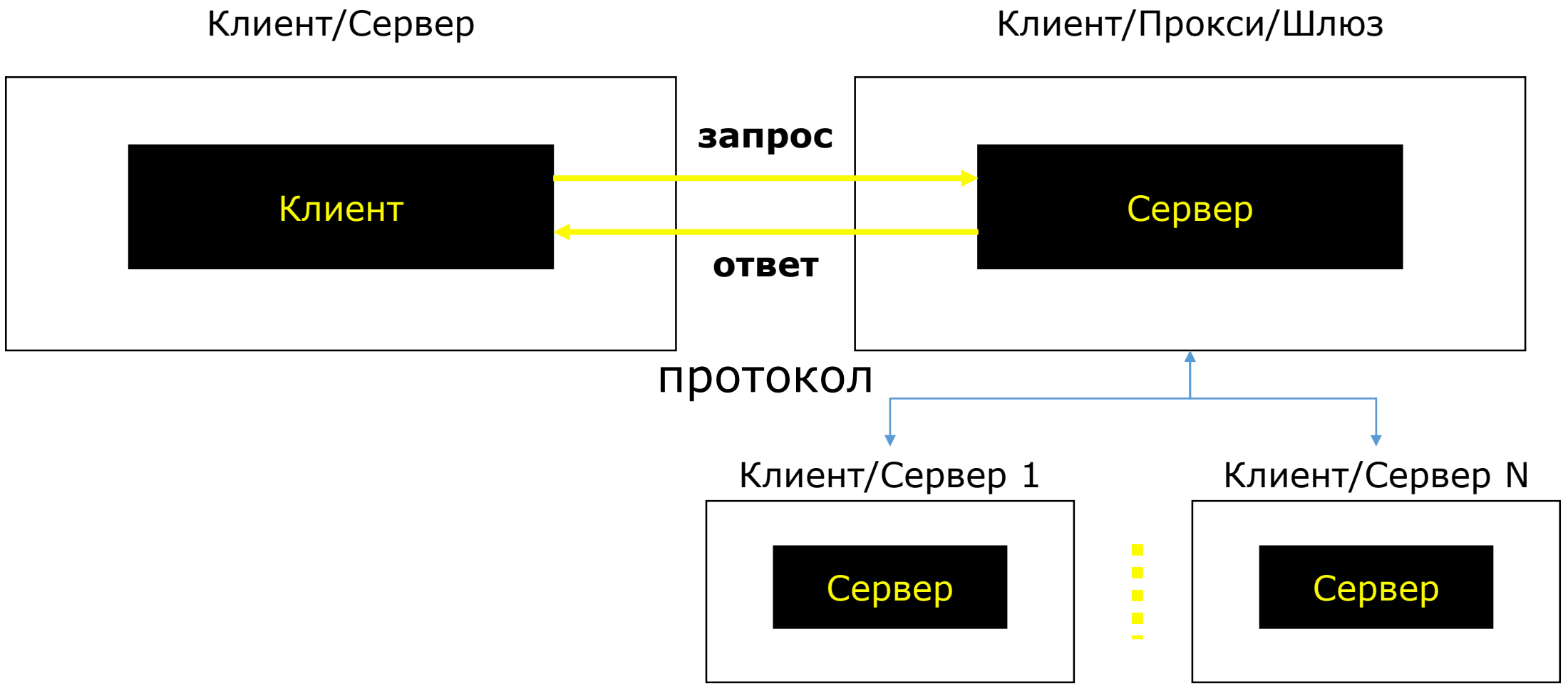
Кэширование

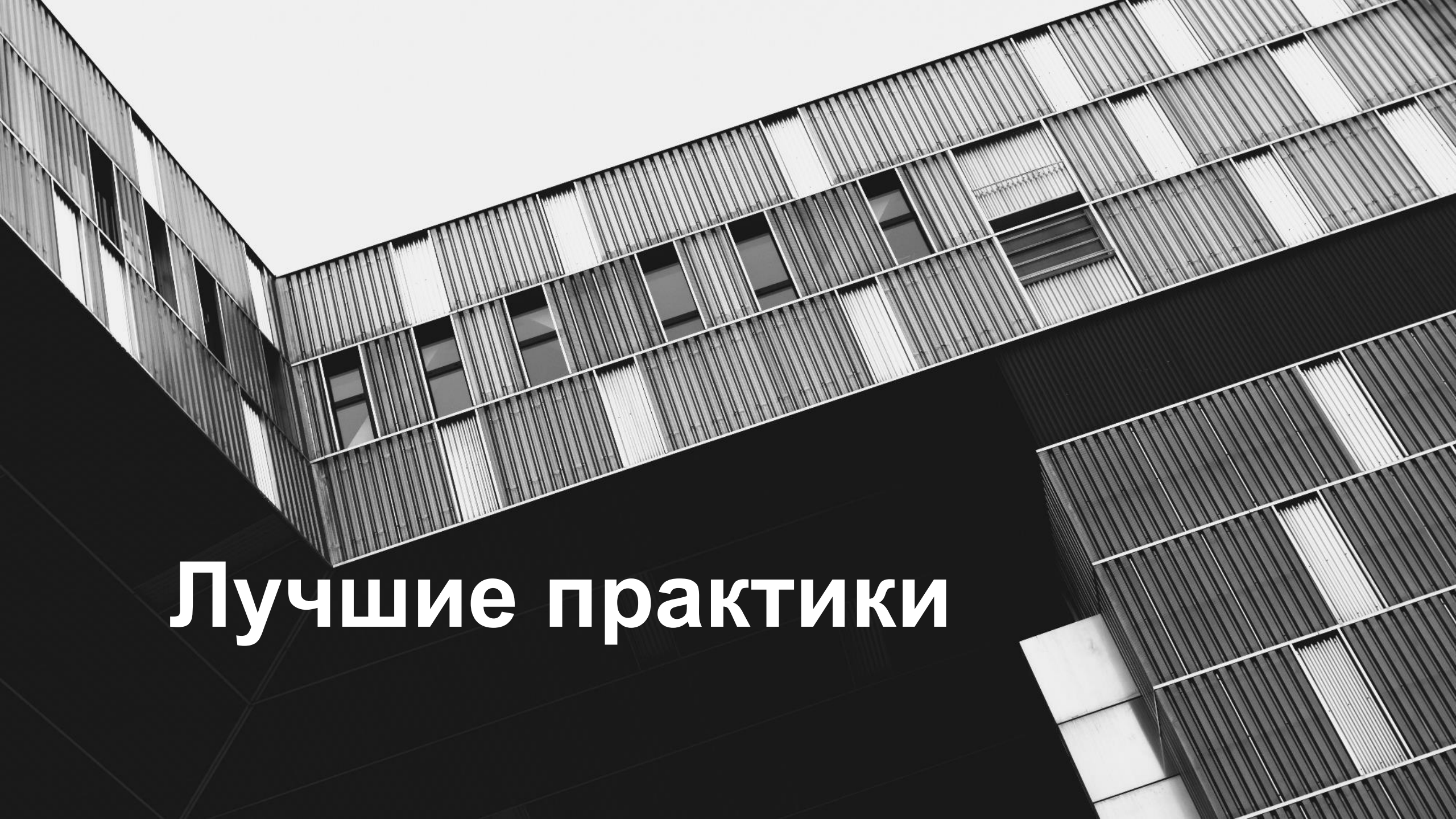


Единообразие интерфейса



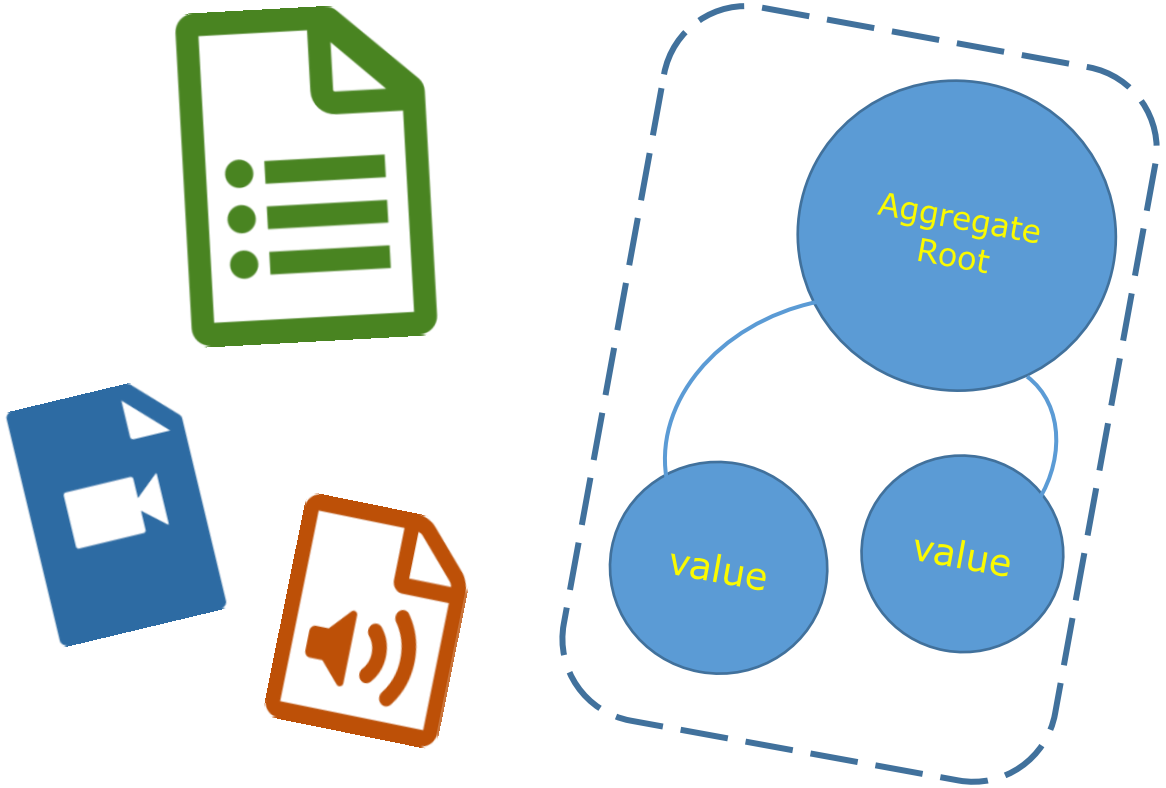
Слои


Клиент/Сервер
Клиент/Прокси/Шлюз
Клиент
запрос
ответ
Сервер
протокол
Клиент/Сервер 1
Клиент/Сервер N
Сервер
Сервер



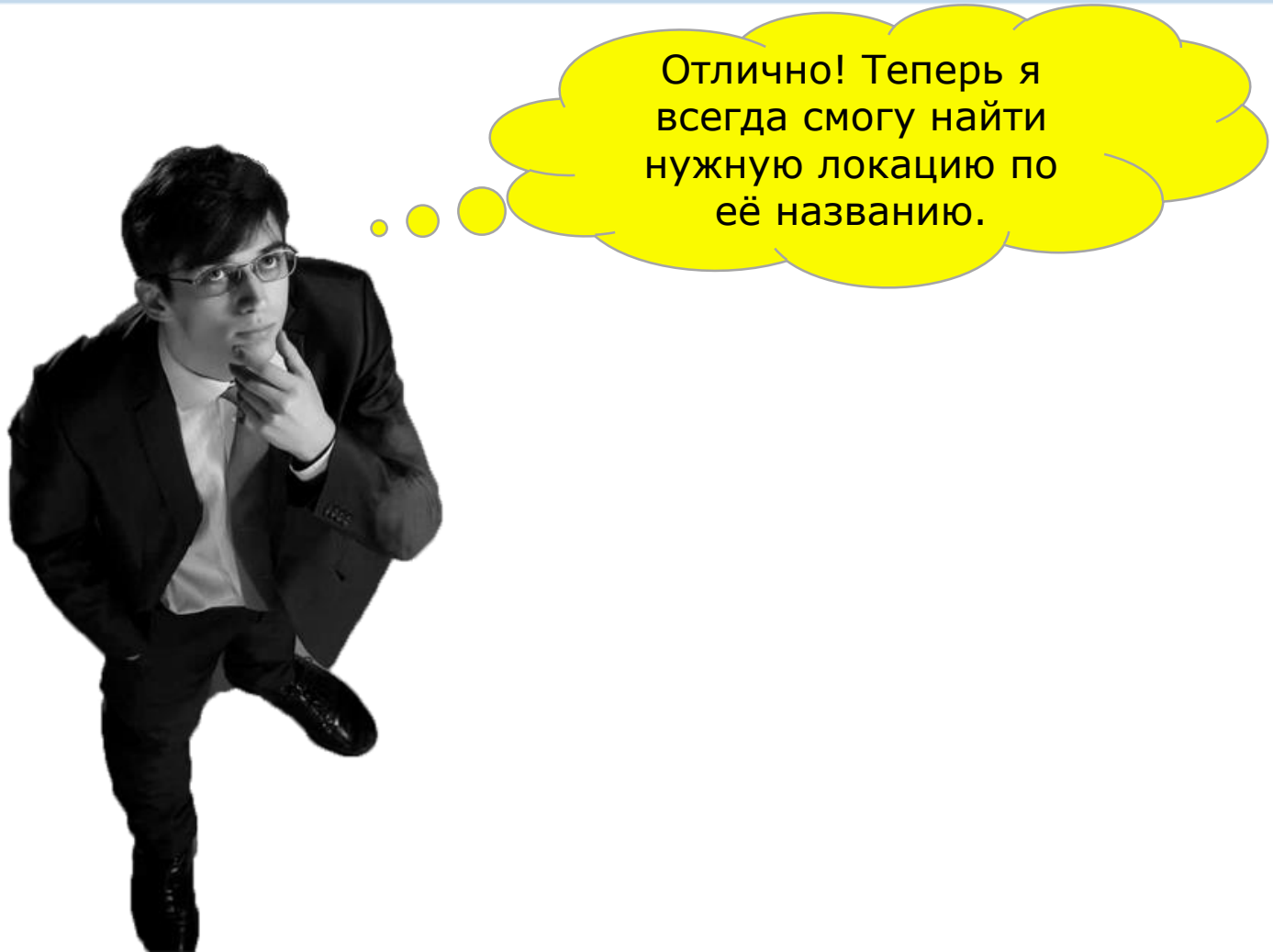
Лучшие практики

Что такое «Ресурс»?



Идентификатор ресурса (URI)

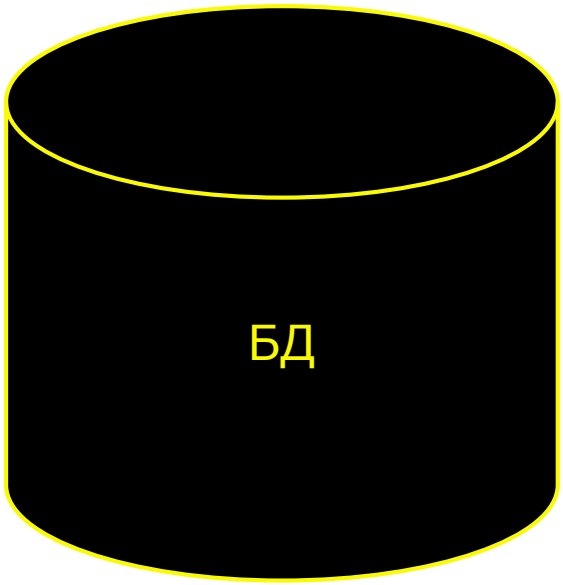
GET /api/locations/{name} Получает описание локации по заданному уникальному идентификатору.



Избегайте глаголов в URI

POST	/api/locations/create	Создаёт новую локацию по предоставленному описанию.

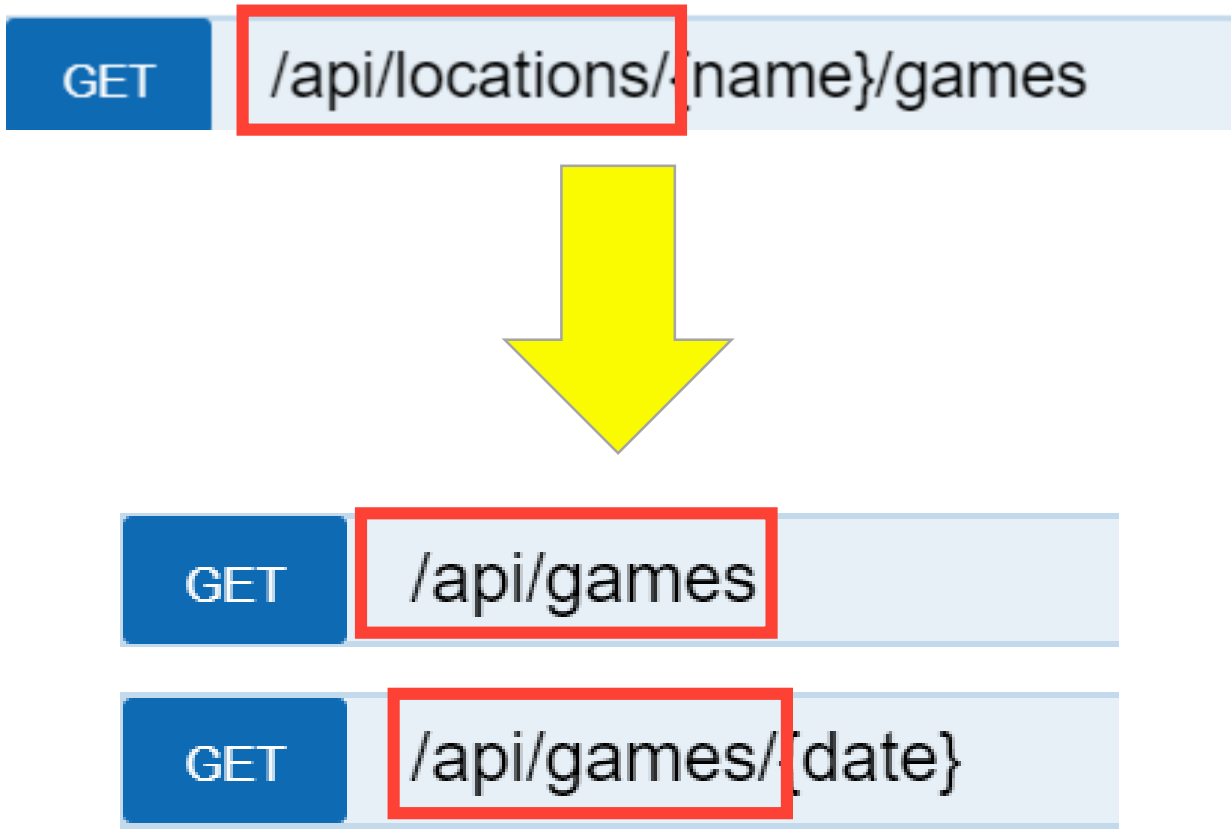
Физическое хранение



Locations (Part 1)

Locations (Part 2)

Коллекции – это ресурсы



Избегайте сложных URI ресурсов

Райффайзен

GET /api/locations/{name}/games/{scheduledAt}/employees/{employeeName}/games

Получает список игр, которые закреплены за указанным сотрудником в указанной локации

Кажется этот URI слишком длинный.

- api/employees/{name}/games

Избегайте «болтливых» API

GET /api/locations/{name}

GET /api/locations/{name}/complexity

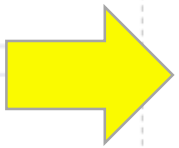
GET /api/locations/{name}/withGames

```
public class LocationModel
{
    ссылок: 6 | 0 изменений | 0 авторов, 0 изменений
    public LocationModel(string name, int maxPlayers, Complexity complexity)
    {
        Name = name;
        MaxPlayers = maxPlayers;
        Complexity = complexity;
    }

    ссылок: 4 | 0 изменений | 0 авторов, 0 изменений
    public string Name { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public int MaxPlayers { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public Complexity Complexity { get; }
}
```

```
[
  {
    "name": "string",
    "maxPlayers": 0,
    "complexity": 0
  }
]
```

Избегайте «болтливых» API

```
public class LocationModel
{
    ссылок: 6 | 0 изменений | 0 авторов, 0 изменений
    public LocationModel(string name, int maxPlayers, Complexity complexity)
    {
        Name = name;
        MaxPlayers = maxPlayers;
        Complexity = complexity;
    }

    ссылок: 4 | 0 изменений | 0 авторов, 0 изменений
    public string Name { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public int MaxPlayers { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public Complexity Complexity { get; }
}
```



```
public class LocationModel
{
    ссылок: 3 | 0 изменений | 0 авторов, 0 изменений
    public LocationModel(string name, int maxPlayers, Complexity complexity)
    {
        Name = name;
        MaxPlayers = maxPlayers;
        Complexity = complexity;
        Games = new List<GameModel>();
        Links = new List<LinkModel>();
    }

    ссылок: 4 | 0 изменений | 0 авторов, 0 изменений
    public string Name { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public int MaxPlayers { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public Complexity Complexity { get; }
    ссылок: 0 | 0 изменений | 0 авторов, 0 изменений
    public List<EmployeeModel> Employees { get; }
    ссылок: 3 | 0 изменений | 0 авторов, 0 изменений
    public List<GameModel> Games { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public List<LinkModel> Links { get; }

}
```

Определяйте операции в терминах методов HTTP

Райффайзен

Locations

Show/Hide | List Operations | Expand Operations

GET	/api/locations	Получает список локаций студии.
POST	/api/locations	Создаёт новую локацию по предоставленному описанию.
PUT	/api/locations	Создаёт или обновляет локацию по предоставленному описанию.
DELETE	/api/locations/{name}	Удаляет указанную локацию.
GET	/api/locations/{name}	Получает описание локации по заданному уникальному идентификатору.
POST	/api/locations/{name}	Создаёт новую локацию по предоставленному описанию.
DELETE	/api/locations/{name}/games	Удаляет список игр в указанной локации.
GET	/api/locations/{name}/games	Получает описание массив с рассписанием игр в указанной локации.
PUT	/api/locations/{name}/games	Создаёт или обновляет список игр в указанной локации.
GET	/api/locations/{name}/employees	Получает список сотрудников прикреплённых к указанной локации.

Схема использования HTTP методов

Райффайзен IT

Ресурс	POST	GET	PUT	DELETE
locations	Создаёт новый	Возвращает все	Обновляет все	Удаляет все
Locations/{name}	Вернёт ошибку	Вернёт экземпляр	Обновит экземпляр если он есть	Удалит экземпляр
locations/{name}/games	Создаст новую игру в локации	Вернёт все игры в локации	Обновит все игры в локации	Удалит все игры в локации

PUT vs PATCH

HTTP Method	Идемпотентность
POST	
GET	✚
PUT	✚
PATCH	✚
DELETE	✚

PUT vs PATCH

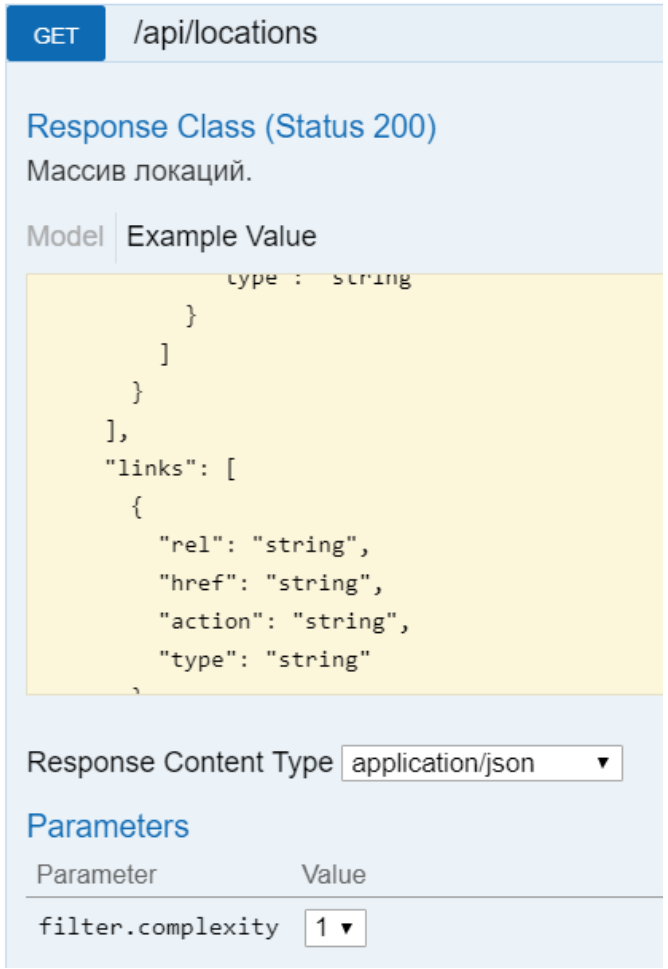
```
/// <summary> Получает описание локации по заданному уникальному идентификатору.
[HttpGet][Route("{name}")][ResponseType(typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.OK, Description = "Описание локации.", Type = typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.NotFound, Description = "Локация с указанным наименованием не найдена.")]
ссылок: 0 | 0 изменений | 0 авторов, 0 изменений
public IHttpActionResult GetLocation(string name)
{
  //...,

  if (location is null)
  {
    return NotFound();
  }

  //var eTag = new EntityTagHeaderValue(location.LastModifiedAt.ToString());
  var eTag = new EntityTagHeaderValue(location.GetHashCode().ToString());

  return new OkResultWithCaching<LocationModel>(location, this)
  {
        //...,
        ETag = eTag
  };
}
```

PUT vs PATCH

```
/// <summary> Создаёт или обновляет локацию по предоставленному описанию.
[HttpPut][Route][ResponseType(typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.Created, Description = "Описание локации обновлено.", Type = typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.NoContent, Description = "Описание локации обновлено.", Type = typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.NotFound, Description = "Локация с указанным наименованием не найдена.")]
ссылок: 0 | 0 изменений | 0 авторов, 0 изменений
public IHttpActionResult UpdateOrCreateLocation(LocationModel location)
{
  var nonMatchEtags = Request.Headers.IfNoneMatch;
  var hashedLocation = location.GetHashCode().ToString();

  if (nonMatchEtags.Count > 0
    && string.CompareOrdinal(nonMatchEtags.First().Tag, hashedLocation) == 0)
  {
    //Update...
    return StatusCode(HttpStatusCode.NoContent);
  }

  if(location is null)
  {
    location = new LocationModel("Name", 5, Complexity.Easy);
    return Created($"api/locations/{location.Name}", location);
  }
  else
  {
    return StatusCode(HttpStatusCode.PreconditionFailed);
  }
}
```

412 HTTP Precondition Failed

Фильтрация




```
/// <summary> Получает список локаций студии.
[HttpGet]
[Route]
[ResponseType(typeof(IEnumerable<LocationModel>))]
[SwaggerResponse(HttpStatusCode.OK,
    Description = "Массив локаций.",
    Type = typeof(IEnumerable<LocationModel>))]
[SwaggerResponse(HttpStatusCode.NotFound,
    Description = "Локации соответствующие условию отбора не найдены.")]
ссылок: 0 | 0 изменений | 0 авторов, 0 изменений
public IHttpActionResult GetLocations([FromUri] LocationFilter filter = null)
{
  if(filter is null)
  {
    return Ok(_locations);
  }
  else
  {
    var filteredLocations = _locations.Where(location => location.Complexity == filter.Complexity);

    if (filteredLocations.Any())
    {
      return Ok(_locations);
    }
    else
    {
      return NotFound();
    }
  }
}
```

Фильтрация по старинке

GET /api/locations/{name}

GET /api/locations/{name}/complexity

GET /api/locations/{name}/withGames

```
public class LocationModel
{
    ссылок: 6 | 0 изменений | 0 авторов, 0 изменений
    public LocationModel(string name, int maxPlayers, Complexity complexity)
    {
        Name = name;
        MaxPlayers = maxPlayers;
        Complexity = complexity;
    }

    ссылок: 4 | 0 изменений | 0 авторов, 0 изменений
    public string Name { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public int MaxPlayers { get; }
    ссылка: 1 | 0 изменений | 0 авторов, 0 изменений
    public Complexity Complexity { get; }
}
```

```
[
  {
    "name": "string",
    "maxPlayers": 0,
    "complexity": 0
  }
]
```

Постраничный вывод

GET /api/locations

Response Class (Status 200)

Массив локаций.

Model | Example Value


```
        "links": [
          {
            "rel": "string",
            "href": "string",
            "action": "string",
            "type": "string"
          }
        ]
      }
    ],
    "locations": [
```


Response Content Type application/json

Parameters

Parameter	Value
limit	
offset	

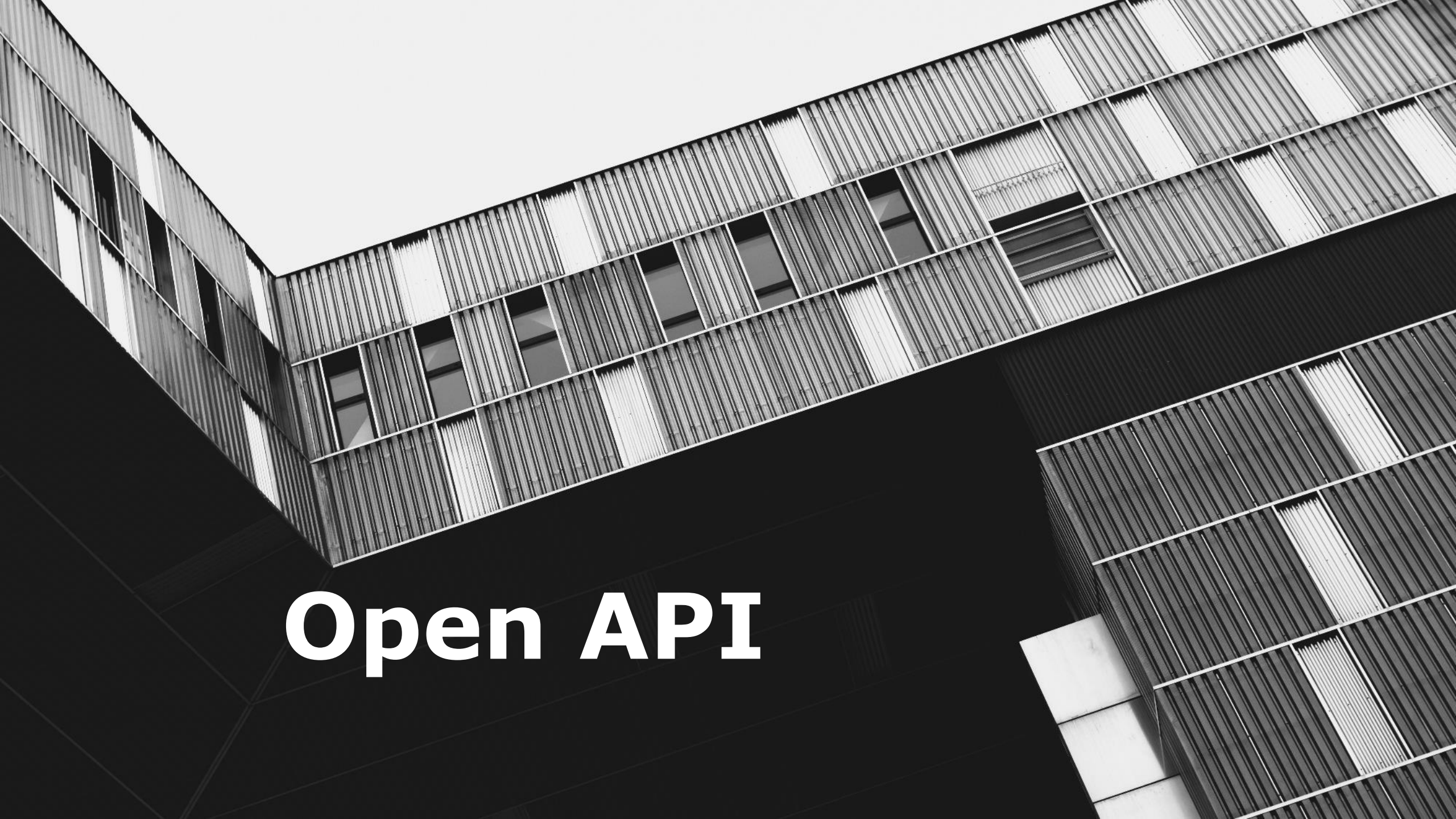
```
/// <summary> Получает список локаций студии.
[HttpGet]
[Route]
[ResponseType(typeof(IEnumerable<LocationModel>))]
[SwaggerResponse(HttpStatusCode.OK,
    Description = "Массив локаций.",
    Type = typeof(IEnumerable<LocationModel>))]
[SwaggerResponse(HttpStatusCode.NotFound,
    Description = "Локации соответствующие условию отбора не найдены.")]
ссылок: 0 | 0 изменений | 0 авторов, 0 изменений
public IHttpActionResult GetLocationsRange([FromUri]int limit = 25, [FromUri]int offset = 0)
{
  var filteredLocations = _locations.Skip(offset).Take(limit);

  if (filteredLocations.Any())
  {
    return Ok(_locations);
  }
  else
  {
    return NotFound();
  }
}
```

Cache


```
/// <summary> Получает описание локации по заданному уникальному идентификатору.
[HttpGet][Route("{name}")][ResponseType(typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.OK, Description = "Описание локации.", Type = typeof(LocationModel))]
[SwaggerResponse(HttpStatusCode.NotFound, Description = "Локация с указанным наименованием не найдена.")]
ссылок: 0 | 0 изменений | 0 авторов, 0 изменений
public IHttpActionResult GetLocation(string name)
{
  //...,

  if (location is null)
  {
    return NotFound();
  }

  //var eTag = new EntityTagHeaderValue(location.LastModifiedAt.ToString());
  var nonMatchEtags = Request.Headers.IfNoneMatch;
  var eTag = new EntityTagHeaderValue($"\"{location.GetHashCode()}\"");
  var cacheControlHeader = new CacheControlHeaderValue();
  cacheControlHeader.Public = true;
  cacheControlHeader.MaxAge = new TimeSpan(0, 10, 0);

  var isModified = string.CompareOrdinal(nonMatchEtags.First().Tag, location.GetHashCode().ToString()) == 0;

  if (nonMatchEtags.Count > 0 && isModified)
    return StatusCode(HttpStatusCode.NotModified);
  else
    return new OkResultWithCaching<LocationModel>(location, this)
    {
      //...,
      ETag = eTag
    };
}
```

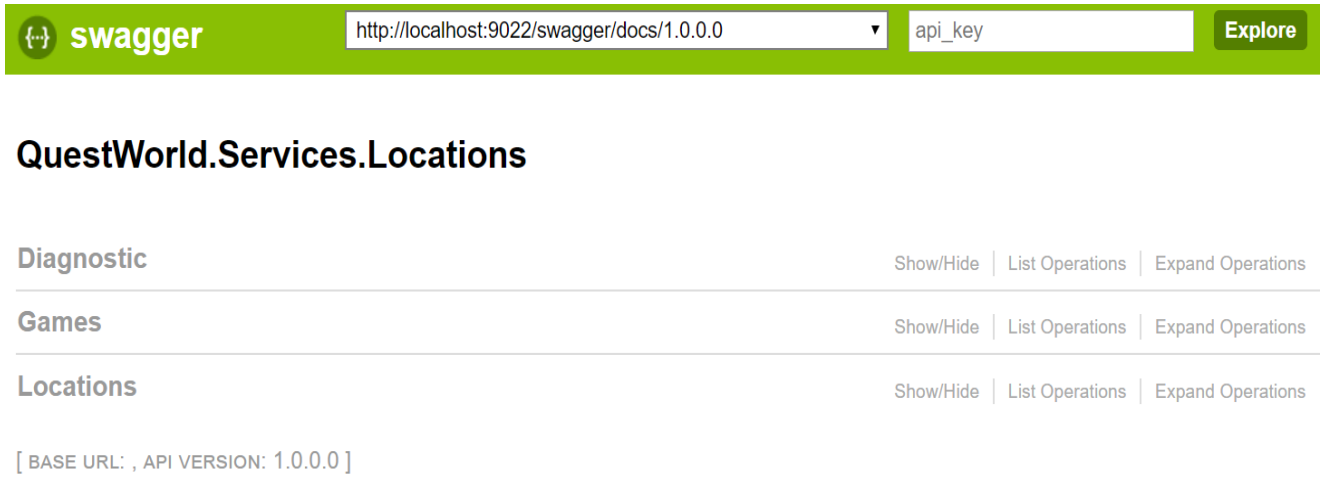
Hypermedia


```
{
  "name": "Sacrifice",
  "maxPlayers": 6,
  "complexity": 2,
  "employees": null,
  "games": [],
  "links": [
    {
      "rel": "self",
      "href": "localhost:9022/api/locations/sacrifice",
      "action": "GET",
      "type": "application/json"
    },
    {
      "rel": "self",
      "href": "localhost:9022/api/locations/sacrifice",
      "action": "PUT",
      "type": "application/json"
    },
    {
      "rel": "self",
      "href": "localhost:9022/api/locations/sacrifice",
      "action": "DELETE",
      "type": ""
    },
    {
      "rel": "games",
      "href": "localhost:9022/api/locations/sacrifice/games",
      "action": "POST",
      "type": "application/json"
    }
  ],
```

Open API

Open API Initiative




```
[SwaggerResponse(HttpStatusCode.OK, Description = "Массив локаций.",
[SwaggerResponse(HttpStatusCode.NotFound, Description = "Локации со
```

Info Object Example

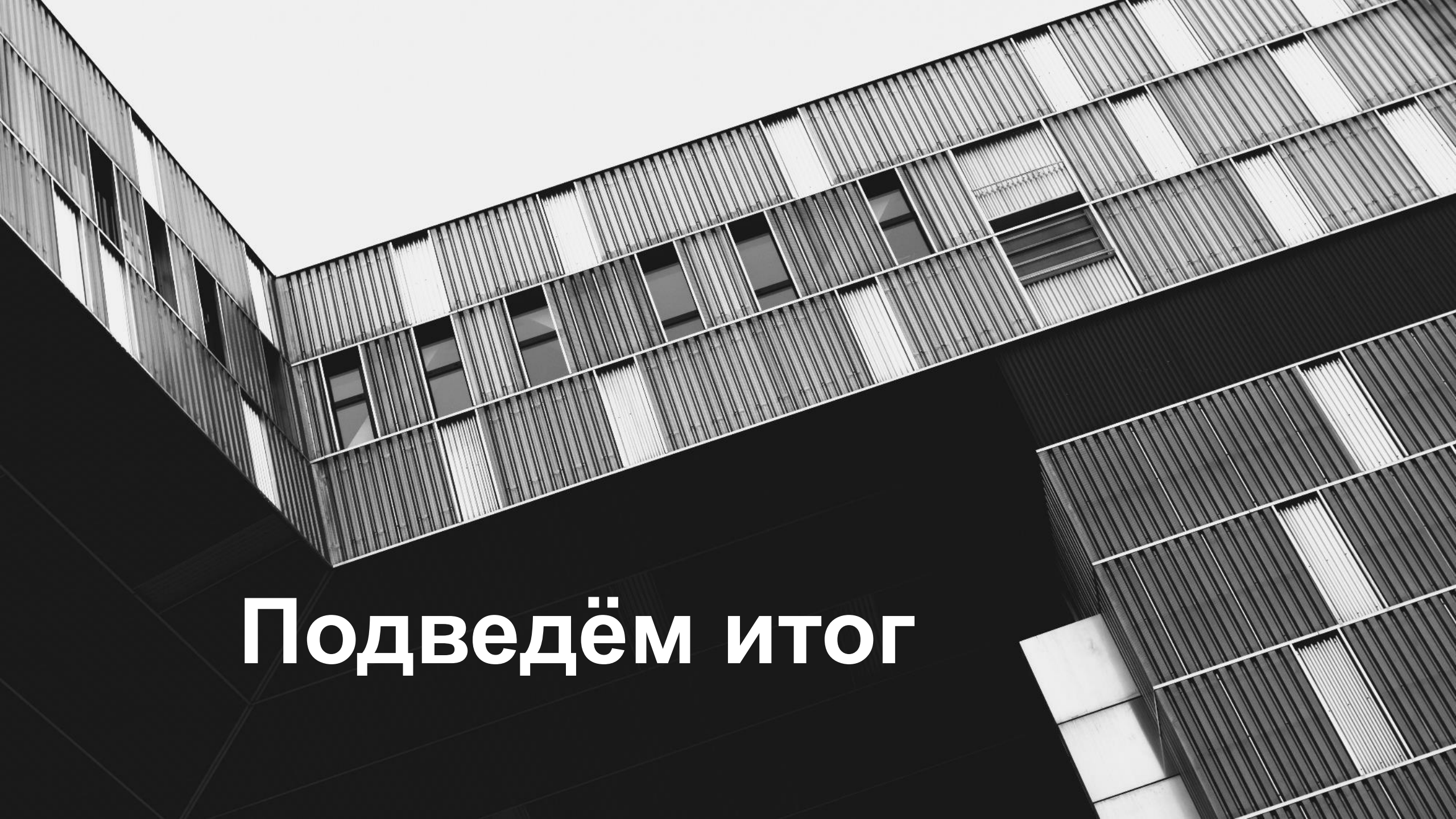
```
{
  "title": "Sample Pet Store App",
  "description": "This is a sample server for a pet store.",
  "termsOfService": "http://example.com/terms/",
  "contact": {
    "name": "API Support",
    "url": "http://www.example.com/support",
    "email": "support@example.com"
  },
  "license": {
    "name": "Apache 2.0",
    "url": "https://www.apache.org/licenses/LICENSE-2.0.html"
  },
  "version": "1.0.1"
}
```

```
title: Sample Pet Store App
description: This is a sample server for a pet store.
termsOfService: http://example.com/terms/
contact:
  name: API Support
  url: http://www.example.com/support
  email: support@example.com
license:
  name: Apache 2.0
  url: https://www.apache.org/licenses/LICENSE-2.0.html
version: 1.0.1
```

Checklist

Checklist

- ✓ Проверьте все маршруты
- ✓ Убедитесь, что все маршруты защищены
- ✓ Убедитесь, что сообщения запроса и ответа хорошо сформированы
- ✓ Убедитесь, что каждая операция возвращает правильные коды состояния
- ✓ Проверьте обработку исключений



Подведём итог

Итог

- ✓ Масштабируемость:
  - ✓ Ресурсы
  - ✓ HTTP Methods
  - ✓ Open API Specification
  - ✓ Фильтры
  - ✓ HATEOAS
- ✓ Производительность:
  - ✓ Постраничный вывод
  - ✓ Кэш

Спасибо
за внимание!

Вопросы?

prosin-roma@ya.ru

Райффайзен