Проектирование API, часть 1

```
GET /api/v1/photos?limit=10
```

```
{
    "status": 200,
    "body": {
        "photos": [
            {"id": 1, "url":"//test.com/1.jpg"},
            {"id": 2, "url":"//test.com/2.jpg"}
        ]
    }
}
```

Проектирование API, часть 1

Основные требования к API:

1. Согласованность

Проектирование API, часть 1

Основные требования к API:

1. Согласованность
2. Расширяемость

Проектирование API, часть 1

Основные требования к API:

1. Согласованность
2. Расширяемость
3. Документация

Проектирование API, часть 1

Согласованность

```
GET /api/v1/photo&id=12
```

```
[12, "//exampl.com/1.jpg", "rvasily", true]
```

Проектирование API, часть 1

Согласованность

```
GET /api/v1/photo&id=12
[12, "//exampl.com/1.jpg", "rvasily", true]

GET /api/v1/photos
[
   {
       "id": 12,
       "user": "rvasily",
       "enabled": true
   }
]
```

Проектирование API, часть 1

Согласованность

```
GET /api/v1/photo&id=12
[12, "//exampl.com/1.jpg", "rvasily", true]
```

```
GET /api/v1/photos
[
   {
      "id": 12,
      "user": "rvasily",
      "enabled": true
   }
]
```

```
GET /api/v1/user/timeline
{
   "status": 200,
   "body": {
      "photos": [
         {"id":1,"url":"..."}
      ]
   }
}
```

Проектирование API, часть 1

Расширяемость

Плохо

Нечитабельно, некуда вставить даже параметры записи

```
GET /api/v1/photo&id=12
```

```
[12, "//exampl.com/1.jpg", "rvasily", true]
```

Проектирование API, часть 1

Расширяемость

Плохо

Некуда вставить дополнительные данные другого типа

```
GET /api/v1/photos
[
   {
       "id": 12,
       "url": "//exampl.com/1.jpg",
       "user_login": "rvasily",
       "followed": true
   }
]
```

Проектирование API, часть 1

Расширяемость

Нормально

Нет выделенного слоя метаданных

Можно вставить любые дополнительные данные

```
GET /api/v1/photos
{
   "photos": [
       {"id": 1, "url": "//example.com/1.jpg"}
   ]
}
```

Проектирование API, часть 1

Расширяемость

Хорошо

Есть выделенный слой метаданных

Можно расширить дополнительными блоками данных

```
GET /api/v1/photos
{
    "status": 200,
    "body": {
        "photos": [
            {"id": 1, "url": "//example.com/1.jpg"}
        ],
        "recommendations": {
            "videos": [{"id": 1, "title": "..."}]
        }
    }
}
```

Проектирование API, часть 1

Расширяемость

HTTP-status для ошибок транспорта

Логируем только HTTP-status

```
// HTTP-status 200
{
   "body": {
       "photos": [{"id": 1}]
   }
}
```

```
// HTTP-status 500
{
   "error": "database"
}
```

Проектирование API, часть 1

Расширяемость

HTTP-status для ошибок транспорта

Логируем только HTTP-status

```
// HTTP-status 200
{
    "body": {
        "photos": [{"id": 1}]
    }
}
```

```
// HTTP-status 500
{
    "error": "database"
}

// HTTP-status 200
// не делайте так!
{
    "error": "database"
}
```

Проектирование API, часть 1

Расширяемость

HTTP-status для ошибок транспорта

JSON-status для ошибок приложения

Надо логировать JSON-status через доп. хедер

```
// HTTP-status 200
{
    //все хорошо
    "status": 200,
    "body": {
        "photos": [{"id": 1}]
    }
}
```

```
// HTTP-status 200
{
    //запрос приняли
    //но не смогли обработать
    "status": 500,
    "body": {
        "error": "database"
    }
}
```

Проектирование API, часть 1

Документация

```
// ShowAccount godoc
// @Summary Show a account
// @Description get string by ID
// @Tags accounts
// @Accept  json
// @Produce  json
// @Param id path int true "Account ID"
// @Success 200 {object} model.Account
// @Failure 400 {object} httputil.HTTPError
// @Failure 404 {object} httputil.HTTPError
// @Failure 500 {object} httputil.HTTPError
// @Router /accounts/{id} [get]
```

Проектирование API, часть 1

Документация

- Swagger (schema-first)
- Swagger (from comments)
- GraphQL
- Protobuf + gRPC (schema-first)