
Яндекс

Разработка RESTful API with all the bells and whistles

Роман Акинфеев

Яндекс.Диск — это сервис, который позволяет хранить файлы и обмениваться ими, а также предоставляет доступ к файлам с любого устройства, подключённого к интернету.

Яндекс.Диск — это сервис, который позволяет хранить файлы и обмениваться ими, а также предоставляет доступ к файлам с любого устройства, подключённого к интернету.

- 20+ млн. зарегистрированных пользователей

Яндекс.Диск — это сервис, который позволяет хранить файлы и обмениваться ими, а также предоставляет доступ к файлам с любого устройства, подключённого к интернету.

- 20+ млн. зарегистрированных пользователей
- 10+ млн. загружаемых в сутки файлов

Яндекс.Диск — это сервис, который позволяет хранить файлы и обмениваться ими, а также предоставляет доступ к файлам с любого устройства, подключённого к интернету.

- 20+ млн. зарегистрированных пользователей
- 10+ млн. загружаемых в сутки файлов
- 7+ млрд. файлов

Yandex Disk APIs

WebDAV

Для работы с Диском, как с файловой системой

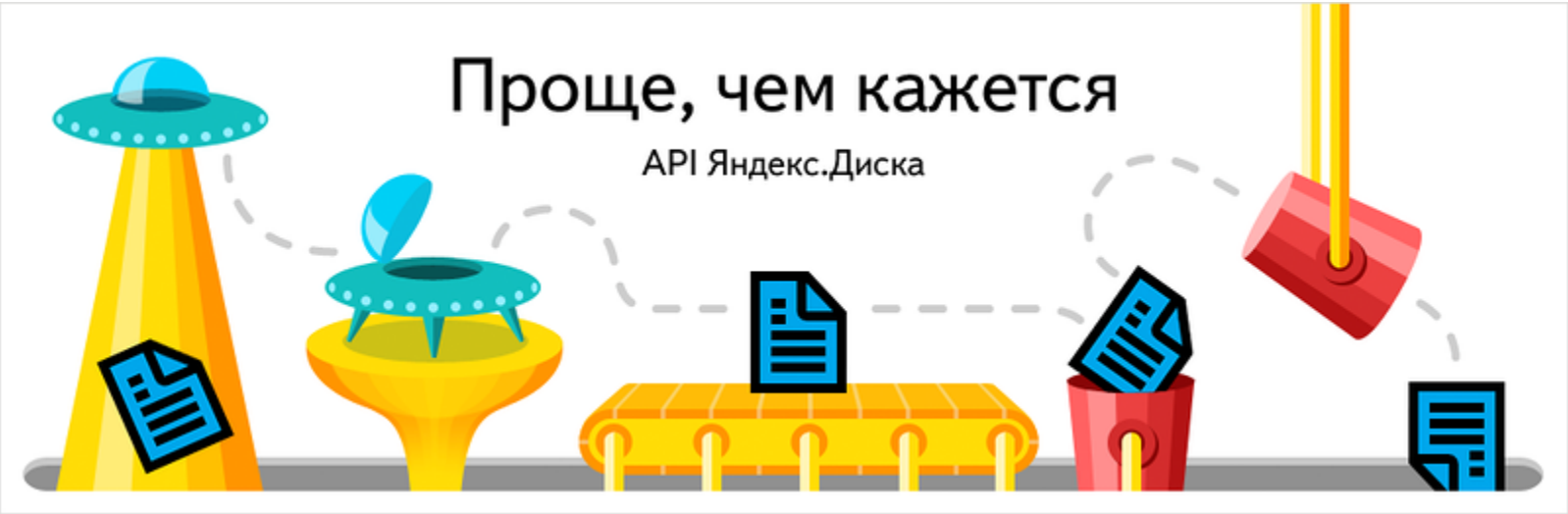
Yandex Disk APIs

WebDAV

Для работы с Диском, как с файловой системой

RESTful API

Для работы с Диском там, где WebDAV’а мало



Задачи

Задачи

- Простота изучения возможностей API

Задачи

- Простота изучения возможностей API
- Легко расширяемая функциональность

Задачи

- Простота изучения возможностей API
- Легко расширяемая функциональность
- Простота разработки под API

RESTful API Яндекс Диска

- Понятная и логичная структура

RESTful API Яндекс Диска

- Понятная и логичная структура
- Hypermedia API

RESTful API Яндекс Диска

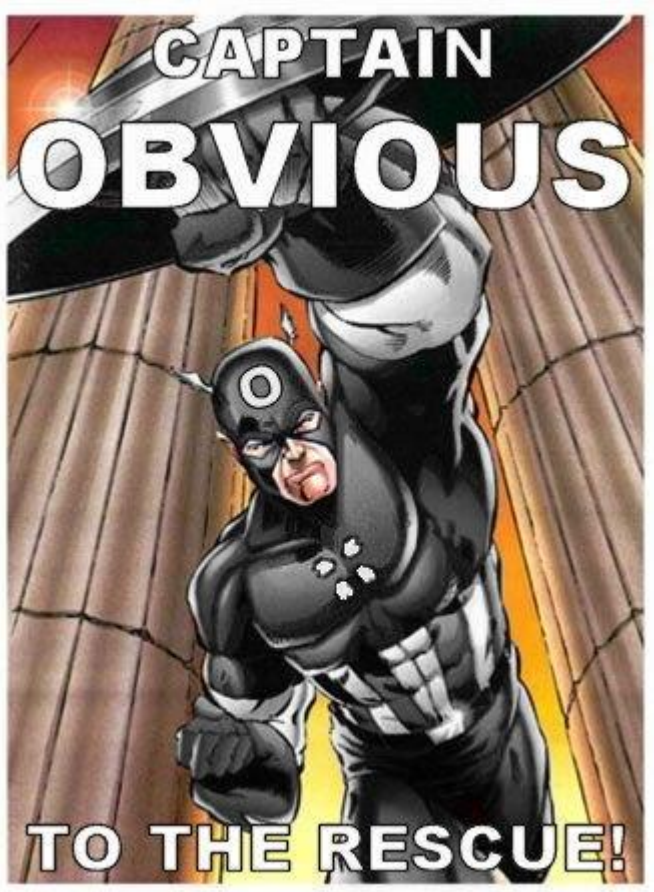
- Понятная и логичная структура
- Hypermedia API
- Self-descriptive & Machine-readable API

Что такое REST?

- Просто набор принципов
- Никаких готовых решений
- Только ограничения

Клиент-сервер и интерфейс

- Клиент и сервер знают интерфейс
- Клиент и сервер не знают друг друга

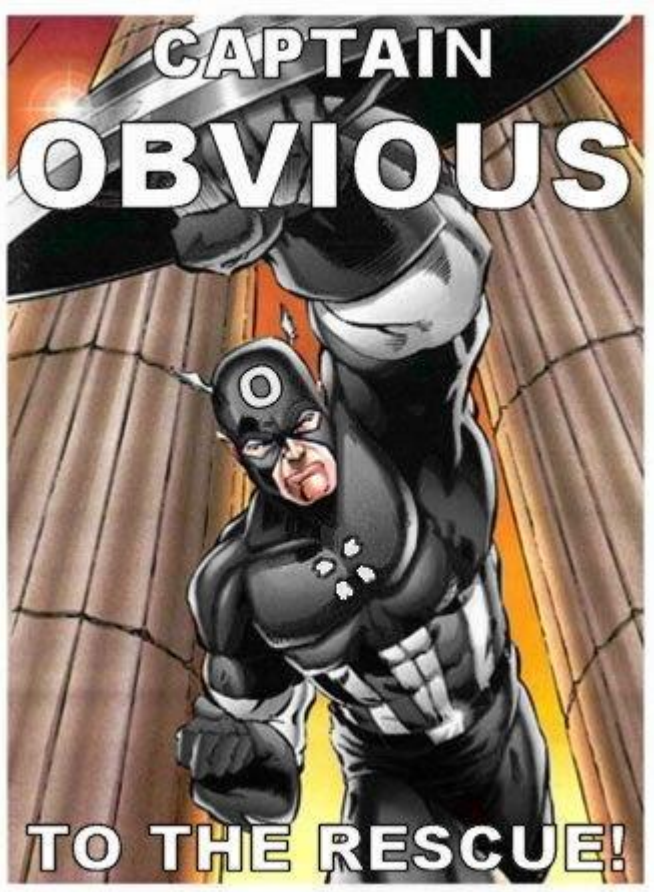


Клиент-сервер и интерфейс

- Клиент и сервер знают интерфейс
- Клиент и сервер не знают друг друга

Профит

- Много клиентов хороших и разных
- Сервер не замечает, как обновляются клиенты
- Клиенты не замечают, как обновляется сервер



Stateless

- Сервер не хранит никакой информации о состоянии клиента между запросами

Stateless

- Сервер не хранит никакой информации о состоянии клиента между запросами
- Нет соединений
- Нет сессий
- Нет истории операций клиента

Stateless

- Сервер не хранит никакой информации о состоянии клиента между запросами
- Нет соединений
- Нет сессий
- Нет истории операций клиента

Профит

- Бэкэнд легко масштабируется
- Клиент не беспокоится ни о чём между запросами

Кэшируемость и многослойность

- Сервер сообщает, что и как кэшировать
- Клиент может не соединяться напрямую с сервером

Кэшируемость и многослойность

- Сервер сообщает, что и как кэшировать
- Клиент может не соединяться напрямую с сервером

Профит

- Возможность снизить нагрузку на бэкэнд
- Возможность работать с кэшем на клиенте

Рецепт

RESTful API Яндекс Диска

Ресурсы

Ресурсы

- Система оперирует ресурсами

Ресурсы

- Система оперирует ресурсами
- Ресурсы имеют чёткую структуру

Ресурсы

- Система оперирует ресурсами
- Ресурсы имеют чёткую структуру
- URL – уникальный идентификатор ресурса

Ресурсы

- Система оперирует ресурсами
- Ресурсы имеют чёткую структуру
- URL – уникальный идентификатор ресурса

Ресурсы API Диска:

- Файлы и папки – **resources**
- Асинхронные операции – **operations**

URL ресурсов

URL ресурсов

URL группируются в нэймспэйсы

/disk

URL ресурсов

URL группируются в нэймспэйсы

/disk

URL коллекции ресурсов всегда во множественном числе

/disk / resources

/disk / operations

URL ресурсов

URL группируются в нэймспэйсы

/disk

URL коллекции ресурсов всегда во множественном числе

/disk / resources

/disk / operations

URL коллекции + идентификатор = URL ресурса

/disk / resources ? path={path}

/disk / operations ? id={id}

URL ресурсов

URL группируются в нэймспэйсы

/disk

URL коллекции ресурсов всегда во множественном числе

/disk / resources

/disk / operations

URL коллекции + идентификатор = URL ресурса

/disk / resources ? path={path}

/disk / operations ? id={id}

/pets / kittens / {name}

HTTP-методы

Основные CRUD-операции:

GET, POST, PUT, DELETE

HTTP-методы

GET – безопасный. Делаем запрос, не задумываясь:

```
GET /disk/resources?path={path}
GET /disk/operations?id={id}
```

HTTP-методы

GET – безопасный. Делаем запрос, не задумываясь:

```
GET /disk/resources?path={path}
GET /disk/operations?id={id}
```

GET, PUT, DELETE – идемпотентные. Повторяем при обрыве, не задумываясь:

```
PUT /disk/resources?path={path}
DELETE /disk/resources?path={path}
```

HTTP-методы

GET – безопасный. Делаем запрос, не задумываясь:

```
GET /disk/resources?path={path}
GET /disk/operations?id={id}
```

GET, PUT, DELETE – идемпотентные. Повторяем при обрыве, не задумываясь:

```
PUT /disk/resources?path={path}
DELETE /disk/resources?path={path}
```

POST – опасный. Изменяет состояние ресурса, повторять опасно

HTTP-методы

GET – безопасный. Делаем запрос, не задумываясь:

```
GET /disk/resources?path={path}
GET /disk/operations?id={id}
```

GET, PUT, DELETE – идемпотентные. Повторяем при обрыве, не задумываясь:

```
PUT /disk/resources?path={path}
DELETE /disk/resources?path={path}
```

POST – опасный. Изменяет состояние ресурса, повторять опасно

OPTIONS – подскажет поддерживаемые ресурсом методы

Когда CRUD мало

Когда CRUD мало

Копирование и Перемещение

- Не идемпотентны
- Не безопасны

Когда CRUD мало

Копирование и Перемещение

- Не идемпотентны
- Не безопасны

Надо использовать POST, но метод один, а операции две

Когда CRUD мало

Копирование и Перемещение

- Не идемпотентны
- Не безопасны

Надо использовать POST, но метод один, а операции две

На помощь приходят они! Ad Hoc-методы!

```
POST /disk/resources/ copy ? path={path}&from={from}
POST /disk/resources/ move ? path={path}&from={from}
```

The Bells And Whistles

RESTful API Яндекс Диска

Hypermedia API

Предоставлять данные о том, что можно делать с каждым ресурсом.

Hypermedia API

Предоставлять данные о том, что можно делать с каждым ресурсом.

Hypermedia as the Engine of Application State

Клиент должен взаимодействовать с сервером посредством hypermedia-контролов, получаемых от сервера.

Hypermedia API

Предоставлять данные о том, что можно делать с каждым ресурсом.

Hypermedia as the Engine of Application State

Клиент должен взаимодействовать с сервером посредством hypermedia-контролов, получаемых от сервера.

Профит

- Клиент не дёргает захардкоденные URL
- Клиент переходит по ссылкам

Hypermedia API

- Collection+JSON
- HAL
- DocJSON
- JSON API
- JSON Hyperschema

Hypertext Application Language

- Чрезвычайно прост
- Есть draft RFC-стандарта
- Уже встречается в публичных API
- MIME-type: `application/hal+json`

Hypertext Application Language

- Чрезвычайно прост
- Есть draft RFC-стандарта
- Уже встречается в публичных API
- MIME-type: `application/hal+json`

Благодаря HAL

- Клиент знает, что и как может сделать с объектом
- Сервер может управлять набором доступных действий

Обычное API

```
# пусть у нас есть объект папки
print folder
{
    “name”: “foo”,
    “path”: “disk:/foo”,
    “type”: “dir”
}
# удаляем папку
URL = 'https://cloud-api.yandex.net/v1/disk/resources'
query = {}
query['path'] = folder['path']
query['permanently'] = True
qs = urlencode(query)
url = URL + '?' + qs
request('DELETE', url)
<Response [204]>
```

Hypermedia API

```
# пусть у нас есть объект папки с HAL-ссылками
print folder
{
    "_links": {
      "delete": {
        "href": "https://cloud-
api.yandex.net/v1/disk/resources?path=disk%3A%2Ffoo&permanently=True",
        "method": "DELETE"
      },
    },
    “name”: “foo”,
    “path”: “disk:/foo”,
    “type”: “dir”
}
# удаляем папку
action = folder[‘_links’][‘delete’]
request(action[‘method’], action[‘href’])
<Response [204]>
```

Self-describing & Machine-readable

Машиночитаемые способы описания REST API

- RAML
- WADL
- JSON Schema + JSON HyperSchema
- Swagger
- IO Docs
- Apiary Blueprints

Swagger

- Описывает API в виде JSON
- Развивается как стандарт
- Прост для понимания
- Имеет экосистему инструментов

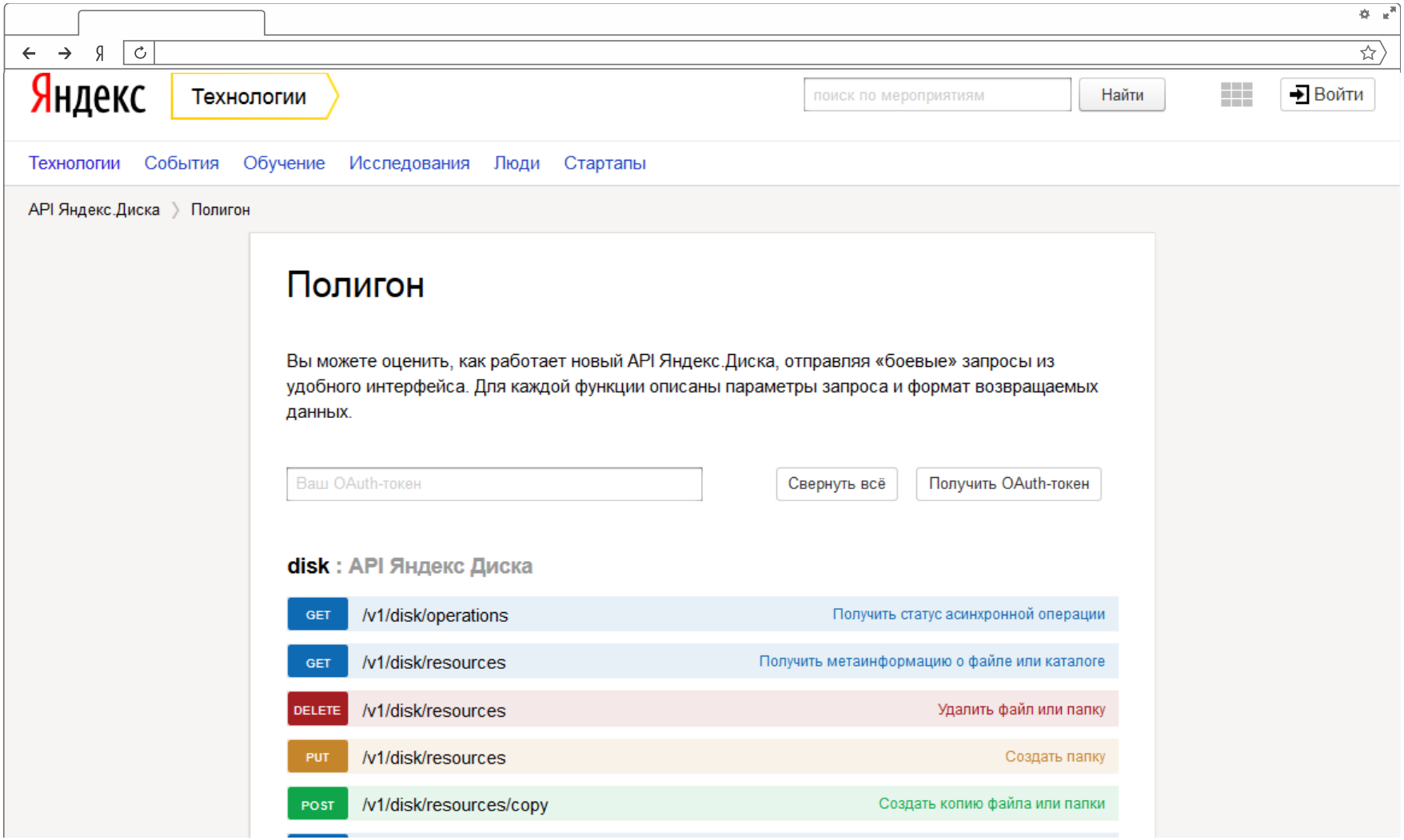
Swagger описывает

- Структуру API
- Параметры операций над ресурсами
- Структуру возвращаемых объектов

Профит от Swagger

- Универсальные Swagger-клиенты
- Автогенерация частей нативных SDK
- Готовый open source UI для API

tech.yandex.ru/disk/poligon



Что в итоге получилось

Что в итоге получилось

- Понятная и расширяемая структура

Что в итоге получилось

- Понятная и расширяемая структура
- Удобная навигация по API с помощью ссылок

Что в итоге получилось

- Понятная и расширяемая структура
- Удобная навигация по API с помощью ссылок
- Самодокументируемость и машиночитаемость

Рецепт правильного REST API

Рецепт правильного REST API

- Структура в виде ресурсов и операций над ними

Рецепт правильного REST API

- Структура в виде ресурсов и операций над ними
- Доступ к объектам через коллекции

Рецепт правильного REST API

- Структура в виде ресурсов и операций над ними
- Доступ к объектам через коллекции
- Старайтесь придерживаться RFC 7231

Рецепт правильного REST API

- Структура в виде ресурсов и операций над ними
- Доступ к объектам через коллекции
- Старайтесь придерживаться RFC 7231
- Используйте Ad Hoc-методы там где не хватает HTTP

Рецепт правильного REST API

- Структура в виде ресурсов и операций над ними
- Доступ к объектам через коллекции
- Старайтесь придерживаться RFC 7231
- Используйте Ad Hoc-методы там где не хватает HTTP
- Добавляйте в API hypermedia-контролы

  HAL – http://stateless.co/hal_specification.html

Рецепт правильного REST API

- Структура в виде ресурсов и операций над ними
- Доступ к объектам через коллекции
- Старайтесь придерживаться RFC 7231
- Используйте Ad Hoc-методы там где не хватает HTTP
- Добавляйте в API hypermedia-контролы

  HAL – http://stateless.co/hal_specification.html
- Описывайте API с помощью машиночитаемых форматов

  Swagger – https://helloreverb.com/developers/swagger

Для чего использовать API Диска

- Работа с контентом пользователей
- Синхронизация данных между устройствами
- Хранение user-related данных приложений

Спасибо за внимание!

Роман Акинфеев

Разработчик back-end и REST API Яндекс.Диска

Клуб разработчиков: clubs.ya.ru/apidisk/

Сервис «Полигон»: tech.yandex.ru/disk/poligon