

# REpresentational State Transfer

## Клиент-сервер (Client-server)

Интерфейс является отдельной программой от сервера с бизнес логикой и базой данных.

## Независим от состояния (Stateless)

Сервер не хранит и не запоминает состояние клиента. Каждый запрос понятен в отрыве от контекста.

## Кэшируемый (Cache)

В сообщении может быть пометка, что данные можно закешировать и обращаться к ним из памяти.

## Унификация интерфейса (Uniform interface)

### Идентификация ресурса (identification of resources)

Каждый ресурс должен иметь уникальный идентификатор.

### Манипуляция через представление (manipulation through representations)

Клиент предоставляет серверу ресурс в уже измененном виде. А сервер уже решает применить эти изменения или нет.

### Самоописанные сообщения (self-descriptive messages)

Данных в сообщении достаточно, чтобы их понять и интерпретировать.

### Гипермедиа как двигатель состояния (hypermedia as the engine of application state)

Сервер подсказывает клиенту какие дальнейшие запросы он может делать.

### Многоуровневость (Layered)

Серверов может быть большое множество. Но каждый знает только о существовании ближайшего.

### Код по запросу (Code on demand)

Сервер может отправить на клиент код для непосредственного исполнения.



[youtube.com/@AcademyNWZ](https://youtube.com/@AcademyNWZ)



[t.me/svyatamesto](https://t.me/svyatamesto)

# REpresentational State Transfer

## Принцип построения URI:

Для описания ресурса используй существительное во множественном числе.

/api/posts - коллекция ресурсов  
/api/posts/42 - конкретный ресурс

URI может быть составным, если ресурс существует в контексте другого ресурса

/api/posts/42/comments - коллекция ресурсов  
/api/posts/42/comments/12 - конкретный ресурс

## Заголовки запроса:

**Accept** – типы медиа, разрешенные в ответе

**Authorization** – авторизационные данные для запроса

**Cache-Control** – определяет, разрешено ли кэширование

**Content-Type** – тип тела запроса

**User-Agent** – ПО, при помощи которого осуществлен запрос – например, браузер

## Заголовки ответа:

**Cache-Control** – определяет, можно ли кэшировать запрос, и как надолго

**Content-Encoding** – тип кодировки в данных

**Content-Type** – формат ответа

**Location** – определяет, был ли запрос перенаправлен, или где создавался ресурс

## Методы запроса:

**GET** – получить ресурс или коллекцию

**POST** – создание или запрос без сайд-эффекта

**PUT** – обновление

**DELETE** – удаление

## HTTP статус коды:

**1xx** – Информационные

**2xx** – Успешные

**3xx** – Перенаправления

**4xx** – Ошибки запроса

**5xx** – Ошибки сервера

200 OK

Запрос успешно обработан

201 Created

Новый ресурс создан

301 Moved Permanently

URL ресурса изменился. Новый адрес в ответе.

400 Bad Request

Ошибка в запросе.  
Неверные данные.

401 Unauthorized

Клиент не авторизован для этого запроса

404 Not Found

Запрашиваемый ресурс не найден

500 Internal Server Error

Ошибка на сервере

Полный список всех кодов: [ссылка](#)



[youtube.com/@AcademyNWZ](https://youtube.com/@AcademyNWZ)



[t.me/svyatamesto](https://t.me/svyatamesto)