

API First. Архитектура REST

API-сервисы раньше и сейчас

Давным-давно, когда компьютеры были большие, а мониторы — маленькие, в интернет выходили с настольного компьютера, а единственным пользовательским клиентом был веб-браузер. Сайты работали по стандартной системе: сервер получал запрос и отдавал клиенту готовые HTML-страницы.

В современном мире у такого подхода есть два недостатка:

- HTML-код нужен лишь браузеру. Например, мобильное приложение вполне обойдётся без HTML-форматирования, ему нужны лишь структурированные данные. Мобильное приложение само позаботится об отображении полученных данных.
- Поскольку HTML-код генерируется на сервере, при переходе от страницы к странице браузер перезагружает сайт целиком. Это неэффективно: приходится перерисовывать одинаковые элементы: шапку, меню, подвал. Разумнее получить данные об изменяющихся частях сайта и отрисовать только их.

Размышляя об этих проблемах, Тим Бернерс-Ли (его называют создателем интернета) и Рой Филдинг (его коллега) придумали принципы, которые позволяли бы масштабировать развитие Всемирной сети.

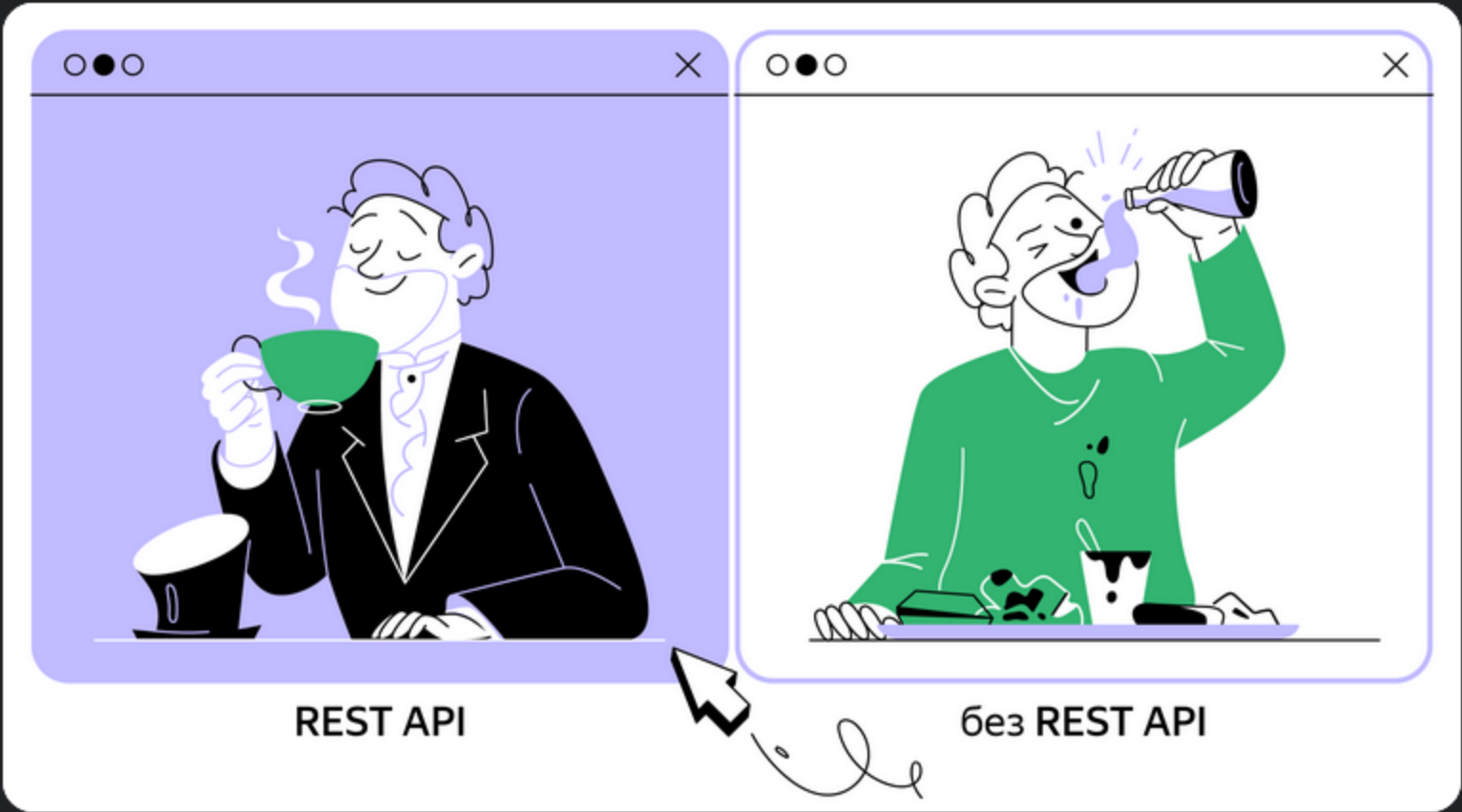
Основная идея в том, что сервер возвращает только запрошенные данные, а клиент сам разбирается, как эти данные отобразить. Мобильное приложение будет использовать свои методы отрисовки, браузер или чат-бот — свои. В результате можно ограничиться одним API для разных платформ.

Такой подход получил название **API First**, то есть сначала данные, а затем — интерфейсы для их отображения.

Так появился **REST**.

REST

REST (англ. *REpresentational State Transfer*, «передача состояния представления») — это набор принципов, которых следует придерживаться при создании API. Если API сделан по этим принципам, его называют **RESTful API** (или просто **REST API**).



Эти принципы стандартизируют передачу данных по сети; они похожи на правила вежливости, когда людям (серверам) удобно находиться в одном обществе (интернете) и понимать друг друга. И хотя за нарушение этих правил никого не накажут — всем становится лучше, если эти правила соблюдаются.

Принципы REST

Эти принципы ввёл Рой Филдинг в 2000 году в своей диссертации «Архитектурные стили и дизайн сетевых программных структур».

1. Клиент-сервер. Разделение ответственности между клиентом и сервером

Клиент и сервер отвечают за разные вещи. Ответственность клиента — пользовательский интерфейс, ответственность сервера — данные. Если API возвращает HTML-страницу, его нельзя назвать REST API: ведь при этом сервер берёт на себя ответственность за интерфейс.

2. Отсутствие состояния. Сервер не хранит состояние

Каждый запрос должен быть независимым, как будто он сделан в первый раз. Сервер не должен хранить какую-либо информацию о клиенте. Каждый запрос клиента к серверу должен содержать всю информацию, необходимую для обработки этого запроса: кто запрашивает данные, какие данные запрашиваются.

3. Единый интерфейс

Интерфейс обращения к серверу одинаков для всех и не зависит от клиента. Запрос к данным может быть сформирован из браузера, мобильного приложения или с «умного» чайника по одним и тем же правилам.

4. Многоуровневость

Первый принцип гласит, что в коммуникации участвуют двое: клиент и сервер. Но можно строить более сложные системы, не нарушая этого принципа.

API сервиса Яндекс Такси может использовать API Яндекс Карт. Клиент (приложение Яндекс Такси в телефоне) взаимодействует только с API сервиса Яндекс Такси, а этот сервис, в свою очередь, выполняет роль клиента для сервиса Яндекс Карты. При таком взаимодействии есть одно условие — каждый компонент должен видеть только свой уровень данных. Например, сервис Яндекс Карты не должен видеть все данные, которые приложение на телефоне отправило в Яндекс Такси.

5. Кешируемость

Данные ответа могут быть закешированы. Это значит, что полученные данные можно сохранить на клиенте, а при идентичном запросе взять их из памяти клиента — кеша, а не ждать их с сервера. Нет смысла запрашивать данные повторно, если они никак не изменились.

6. Код по запросу

Этот принцип необязательный. Он гласит, что функциональность клиента может быть расширена кодом, приходящим с сервера. Сейчас такое можно встретить повсеместно: JavaScript используется для «оживления» страниц и исполнения каких-то сценариев на стороне клиента. Но принципы формулировались в 2000 году — тогда исполняемый код с сервера возвращали не так часто. Потому и выделили это в отдельный принцип.

Что такое REST?

- ☒ Это набор принципов, которых следует придерживаться при создании систем обмена данными.
Правильно, API является как раз одной из таких систем.
- ☐ Это принцип, согласно которому не нужно сильно напрягаться при написании кода и важно часто отдыхать, ведь *rest* переводится с английского как «отдых».
- ☒ Архитектурный стиль, который приводит к повышению производительности и упрощению архитектуры приложения в целом.
Да, верно.

1 из 3 правильно и 0 неправильно

Как вам урок?



Нашли опечатку

К следующему уроку

Краткий пересказ урока