

# REST

---

Материал из Википедии — свободной энциклопедии

**REST** (от англ. *Representational State Transfer* — «передача репрезентативного состояния» или «передача „самоописываемого“ состояния») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. Другими словами, REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать<sup>[1]</sup>. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы; прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. В широком смысле<sup>[*уточнить*]</sup> компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC<sup>[2]</sup>.

В интернете вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют «*REST-запрос*»), а необходимые данные передаются в качестве параметров запроса<sup>[3][4]</sup>.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «**RESTful**».

В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful веб-API. Дело в том, что REST является **архитектурным стилем**, в то время как SOAP является протоколом. Несмотря на то, что REST не является стандартом сам по себе, большинство RESTful-реализаций используют такие стандарты, как HTTP, URL, JSON и, реже, XML.

## Содержание

---

### История термина

### Свойства архитектуры REST

### Требования к архитектуре REST

1. Модель клиент-сервер
2. Отсутствие состояния
3. Кэширование
4. Единообразие интерфейса
5. Слои
6. Код по требованию (необязательное ограничение)

### Преимущества

### Примечания

### Литература

# История термина

---

Хотя данная концепция лежит в самой основе Всемирной паутины, термин «REST» был введён Роем Филдингом, одним из создателей протокола «HTTP», лишь в 2000 году<sup>[4]</sup>. В своей диссертации «*Архитектурные стили и дизайн сетевых программных архитектур*» («*Architectural Styles and the Design of Network-based Software Architectures*»)<sup>[5]</sup> в Калифорнийском университете в Ирвайне<sup>[3]</sup> он подвёл теоретическую основу под способ взаимодействия клиентов и серверов во Всемирной паутине, абстрагировав его и назвав «передачей представительного состояния» («Representational State Transfer»). Филдинг описал концепцию построения распределённого приложения, при которой каждый запрос (REST-запрос) клиента к серверу содержит в себе исчерпывающую информацию о желаемом ответе сервера (желаемом представительном состоянии), и сервер не обязан сохранять информацию о состоянии клиента («клиентской сессии»).

Стиль «REST» развивался параллельно с «HTTP 1.1», разработанным в 1996—1999 годах, основываясь на существующем дизайне «HTTP 1.0», разработанном в 1996 году<sup>[6]</sup>.

## Свойства архитектуры REST

---

Свойства архитектуры, которые зависят от ограничений, наложенных на REST-системы:

- Производительность: взаимодействие компонентов системы может являться доминирующим фактором производительности и эффективности сети с точки зрения пользователя;
- Масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов.

Рой Филдинг (один из главных авторов спецификации протокола HTTP) описывает влияние архитектуры REST на масштабируемость следующим образом:

- Простота унифицированного интерфейса;
- Открытость компонентов к возможным изменениям для удовлетворения изменяющихся потребностей (даже при работающем приложении);
- Прозрачность связей между компонентами системы для сервисных служб;
- Переносимость компонентов системы путём перемещения программного кода вместе с данными;
- Надёжность, выражающаяся в устойчивости к отказам на уровне системы при наличии отказов отдельных компонентов, соединений или данных.<sup>[3]</sup>

## Требования к архитектуре REST

---

Существует **пять** обязательных<sup>[7][8]</sup> ограничений для построения распределённых REST-приложений по Филдингу<sup>[3][9]</sup> и одно необязательное.

Накладываемые ограничения определяют работу сервера в том, как он может обрабатывать и отвечать на запросы клиентов. Действуя в рамках этих ограничений, система приобретает такие желательные свойства как производительность, масштабируемость, простота, способность к изменениям, переносимость, отслеживаемость и надёжность.

Если сервис-приложение нарушает *любое* из этих ограничительных условий, данную систему нельзя считать REST-системой<sup>[3]</sup>.

Обязательными условиями-ограничениями являются:

## 1. Модель клиент-сервер

Первым ограничением, применимым к гибридной модели, является приведение архитектуры к модели клиент-сервер. Разграничение потребностей является принципом, лежащим в основе данного накладываемого ограничения. Отделение потребности интерфейса клиента от потребностей сервера, хранящего данные, повышает переносимость кода клиентского интерфейса на другие платформы, а упрощение серверной части улучшает масштабируемость. Наибольшее же влияние на всемирную паутину, пожалуй, имеет само разграничение, которое позволяет отдельным частям развиваться независимо друг от друга, поддерживая потребности в развитии интернета со стороны различных организаций.<sup>[3]</sup>

## 2. Отсутствие состояния

Протокол взаимодействия между клиентом и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о *состоянии клиента* на сервере не хранится (Stateless protocol или «протокол без сохранения состояния»). Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. *Состояние сессии* при этом сохраняется на стороне клиента<sup>[3]</sup>. Информация о состоянии сессии может быть передана сервером какому-либо другому сервису (например, в службу базы данных) для поддержания устойчивого состояния, например, на период установления аутентификации. Клиент инициирует отправку запросов, когда он готов (возникает необходимость) перейти в новое состояние.

Во время обработки клиентских запросов считается, что клиент находится в *переходном состоянии*. Каждое отдельное *состояние приложения* представлено связями, которые могут быть задействованы при следующем обращении клиента.

## 3. Кэширование

Как и во Всемирной паутине, клиенты, а также промежуточные узлы, могут выполнять кэширование ответов сервера. Ответы сервера, в свою очередь, должны иметь явное или неявное обозначение как кэшируемые или некашируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответ на последующие запросы. Правильное использование кэширования способно частично или полностью устранить некоторые проблемы клиент-серверного взаимодействия, ещё больше повышая производительность и масштабируемость системы.

## 4. Единообразие интерфейса

Наличие унифицированного интерфейса является фундаментальным требованием дизайна REST-сервисов<sup>[3]</sup>. Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо.

К унифицированным интерфейсам предъявляются следующие четыре ограничительных условия<sup>[10][11]</sup>:

## Идентификация ресурсов

Все ресурсы идентифицируются в запросах, например, с использованием URI в интернет-системах. Ресурсы концептуально отделены от представлений, которые возвращаются клиентам. Например, сервер может отсылать данные из базы данных в виде HTML, XML или JSON, ни один из которых не является типом хранения внутри сервера.

## Манипуляция ресурсами через представление

Если клиент хранит представление ресурса, включая метаданные — он обладает достаточной информацией для модификации или удаления ресурса.

## «Самоописываемые» сообщения

Каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать. К примеру, обработчик сообщения (parser), необходимый для извлечения данных, может быть указан в списке MIME-типов<sup>[3]</sup>.

## Гипермедиа как средство изменения состояния приложения (HATEOAS)

Клиенты изменяют состояние системы только через действия, которые динамически определены в гипермедиа на сервере (к примеру, гиперссылки в гипертексте). Исключая простые точки входа в приложение, клиент не может предположить, что доступна какая-то операция над каким-то ресурсом, если не получил информацию об этом в предыдущих запросах к серверу. Не существует универсального формата для предоставления ссылок между ресурсами, Web Linking (RFC 5988 -> RFC 8288) и JSON Hypermedia API Language (<https://tools.ietf.org/id/draft-kelly-json-hal-03.txt>) Архивная копия (<https://web.archive.org/web/20140627002807/https://tools.ietf.org/id/draft-kelly-json-hal-03.txt>) от 27 июня 2014 на Wayback Machine являются двумя популярными форматами предоставления ссылок в REST HYPERMEDIA сервисах.

## 5. Слои

Клиент обычно не способен точно определить, взаимодействует ли он напрямую с сервером или же с промежуточным узлом, в связи с иерархической структурой сетей (подразумевая, что такая структура образует слои). Применение промежуточных серверов способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования. Промежуточные узлы также могут подчиняться политике безопасности с целью обеспечения конфиденциальности информации<sup>[12]</sup>.

## 6. Код по требованию (необязательное ограничение)

REST может позволить расширить функциональность клиента за счёт загрузки кода с сервера в виде апплетов или скриптов. Филдинг утверждает, что дополнительное ограничение позволяет проектировать архитектуру, поддерживающую желаемую функциональность в общем случае, но, возможно, за исключением некоторых контекстов.

## Преимущества

---

Рой Филдинг указывал, что приложения, не соответствующие приведённым условиям, не могут называться REST-приложениями. Если же все условия соблюдены, то, по его мнению, приложение получит следующие преимущества<sup>[3][9]</sup>:

- Надёжность (за счёт отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна);
- Производительность (за счёт использования кэша);
- Масштабируемость;
- Прозрачность системы взаимодействия (особенно необходимая для приложений обслуживания сети);
- Простота интерфейсов;
- Портативность компонентов;
- Лёгкость внесения изменений;
- Способность эволюционировать, приспосабливаясь к новым требованиям (на примере Всемирной паутины).

## Примечания

---

1. Что такое REST API (<https://www.youtube.com/watch?v=J4Fy6lmLBr0>) (рус.). Дата обращения: 11 августа 2021. Архивировано (<https://web.archive.org/web/202108111430/http://www.youtube.com/watch?v=J4Fy6lmLBr0>) 11 августа 2021 года.
2. *Машнин Тимур Сергеевич*. Технология Web-сервисов платформы Java. — БХВ-Петербург, 2012. — С. 115. — 560 с. — ISBN 978-5-9775-0778-3.
3. Chapter 5 of Roy Fielding's dissertation «Representational State Transfer (REST)» ([http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)) Архивная копия ([https://web.archive.org/web/20210513160155/http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://web.archive.org/web/20210513160155/http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)) от 13 мая 2021 на Wayback Machine
4. Fielding discussing the definition of the REST term (<http://tech.groups.yahoo.com/group/rest-discuss/message/6735>). Tech.groups.yahoo.com. Дата обращения: 28 ноября 2013. Архивировано (<https://web.archive.org/web/20101022222125/http://tech.groups.yahoo.com/group/rest-discuss/message/6735>) 22 октября 2010 года.
5. Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST) ([http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)). www.ics.uci.edu. Дата обращения: 1 декабря 2015. Архивировано ([https://web.archive.org/web/20210513160155/https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://web.archive.org/web/20210513160155/https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)) 13 мая 2021 года.
6. rest-discuss : Message: Re: [rest-discuss] RFC for REST? (<http://tech.groups.yahoo.com/group/rest-discuss/message/6757>) (11 ноября 2009). Дата обращения: 1 декабря 2015. Архивировано (<https://web.archive.org/web/20091111012314/http://tech.groups.yahoo.com/group/rest-discuss/message/6757>) 11 ноября 2009 года.
7. *Thomas Erl, Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian*. 5.1 // SOA with REST (неопр.) / Thomas Erl. — Prentice Hall, 2013. — ISBN 978-0-13-701251-0.
8. Richardson, Leonard; Ruby, Sam (2007), *RESTful Web service* (<https://books.google.com/books?id=XUaErakHsoAC>), O'Reilly Media, ISBN 978-0-596-52926-0, Дата обращения: 18 января 2011, "The main topic of this book is the web service architectures which can be considered RESTful: those which get a good score when judged on the criteria set forth in Roy Fielding's dissertation." Источник (<https://web.archive.org/web/20120219072006/https://books.google.com/books?id=XUaErakHsoAC>). Дата обращения: 30 ноября 2016. Архивировано 19 февраля 2012 года.
9. *Thomas Erl, Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian*. 5.1 // SOA with REST. — Prentice Hall, 2013. — ISBN 978-0-13-701251-0.
10. Wilde, Pautasso, 2011, REST Definition.
11. *Н. Л. Подколотный, А. В. Семенычев, Д. А. Рассказов, В. Г. Боровский, Е. А. Ананько, Е. В. Игнатьева, Н. Н. Подколотная, О. А. Подколотная, Н. А. Колчанов* Распределённая система RESTful-web-сервисов для реконструкции и анализа генных сетей. Вавиловский журнал генетики и селекции, т. 16, N 4/1, 2012

12. *Hartley Brody*. How HTTPS Secures Connections: What Every Web Dev Should Know (<https://blog.hartleybrody.com/https-certificates/>) (англ.). Архивировано (<https://web.archive.org/web/20151224052719/https://blog.hartleybrody.com/https-certificates/>) 24 декабря 2015 года.

## Литература

---

- *Erik Wilde, Cesare Pautasso*. REST: From Research to Practice. — Springer Science & Business Media, 2011. — 528 p. — ISBN 978-1-4419-8303-9.

## Ссылки

---

- *Roy Fielding*. Architectural Styles and the Design of Network-based Software Architectures (<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>) (англ.) (2000). Дата обращения: 20 февраля 2009. Архивировано (<https://www.webcitation.org/67gOwyTek?url=http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>) 15 мая 2012 года.
- *Cesare Pautasso; Olaf Zimmerman; Frank Leymann*. RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision (<http://www.jopera.org/docs/publications/2008/restws>) (англ.). 17th International World Wide Web Conference (WWW2008). Дата обращения: 20 февраля 2009. Архивировано (<https://www.webcitation.org/67gOxPVRw?url=http://www.jopera.org/docs/publications/2008/restws>) 15 мая 2012 года.
- *Джон Фландерс*. Введение в службы RESTful с использованием WCF (<http://msdn.microsoft.com/ru-ru/magazine/dd315413.aspx>). MSDN Magazine (январь 2009). Дата обращения: 20 февраля 2009. Архивировано (<https://www.webcitation.org/67gOy2qOM?url=http://msdn.microsoft.com/ru-ru/magazine/dd315413.aspx>) 15 мая 2012 года.
- *Alex Rodriguez*. RESTful Web services: The basics (<http://www.ibm.com/developerworks/library/ws-restful/>) (англ.). IBM. Дата обращения: 15 декабря 2015. Архивировано (<https://web.archive.org/web/20151222150557/http://www.ibm.com/developerworks/library/ws-restful/>) 22 декабря 2015 года.
- *Todd Fredrich*. REST API Tutorial (<http://www.restapitutorial.com/>) (англ.). Дата обращения: 27 октября 2016. Архивировано (<https://web.archive.org/web/20170225065615/http://www.restapitutorial.com/>) 25 февраля 2017 года.

---

Источник — <https://ru.wikipedia.org/w/index.php?title=REST&oldid=135120790>

---

Эта страница в последний раз была отредактирована 25 декабря 2023 в 13:23.

Текст доступен по лицензии Creative Commons «С указанием авторства — С сохранением условий» (CC BY-SA); в отдельных случаях могут действовать дополнительные условия.

Wikipedia® — зарегистрированный товарный знак некоммерческой организации «Фонд Викимедиа» (Wikimedia Foundation, Inc.)