Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

Занятие 9

Back – REST API

План занятия

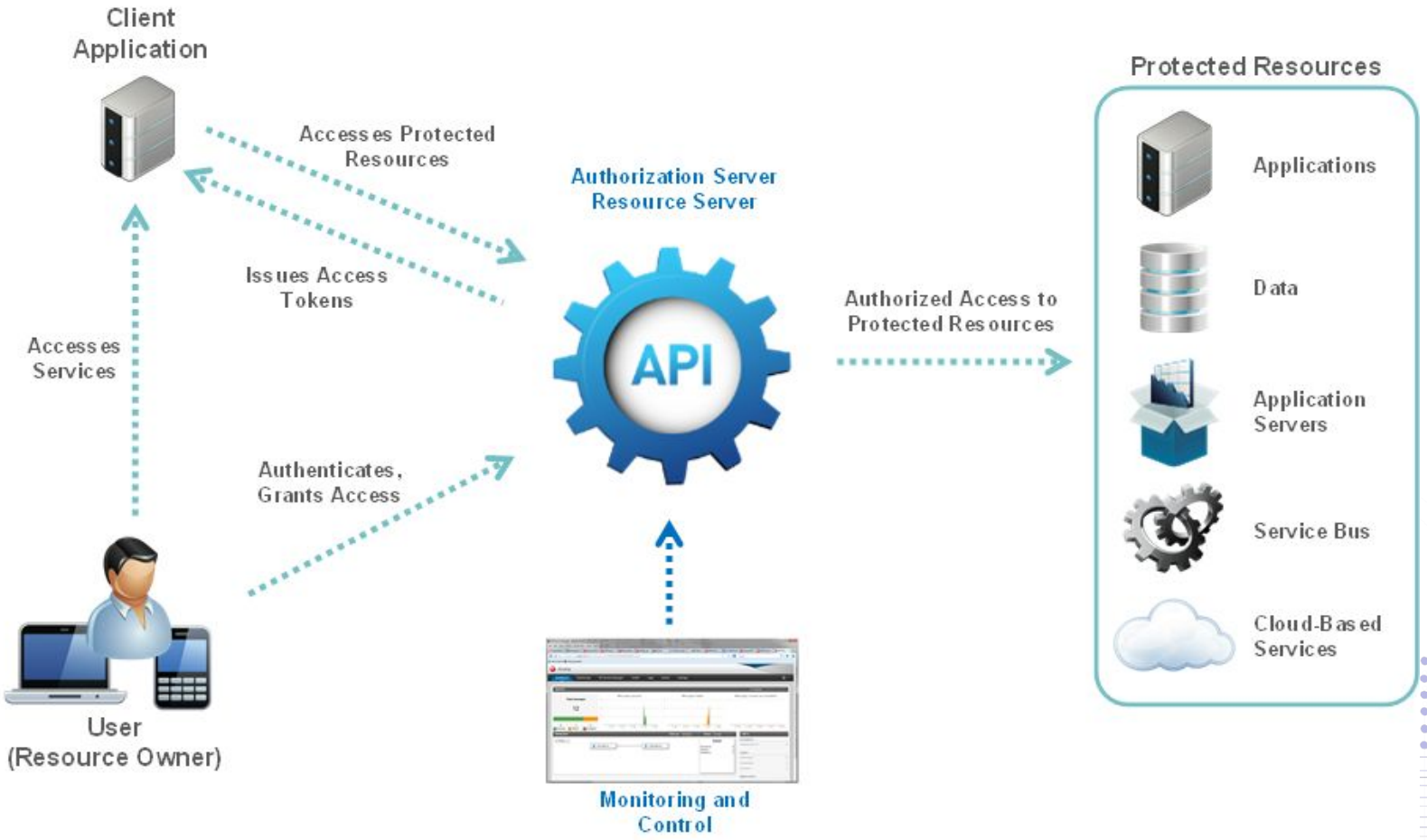
ПЕРЕДОВАЯ
ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

1. Введение
2. REST API
3. Создание REST на практике с помощью nodeJS и Express

Введение

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Большинство бизнес-приложений должны взаимодействовать с другими внутренними и сторонними приложениями для выполнения различных задач.



RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

RESTful API — это интерфейс, используемый двумя компьютерными системами для безопасного обмена информацией через Интернет или локальную сеть.

Что такое API?

Интерфейс прикладного программирования (API) определяет правила, которым необходимо следовать для связи с другими программными системами.

Разработчики внедряют или создают API-интерфейсы, чтобы другие приложения могли программно взаимодействовать с их приложениями.

Например, приложение с табелем рабочего времени содержит API, который запрашивает полное имя сотрудника и диапазон дат. Получив эту информацию, интерфейс внутренне обрабатывает табель рабочего времени сотрудника и возвращает количество часов, отработанных за указанный период.

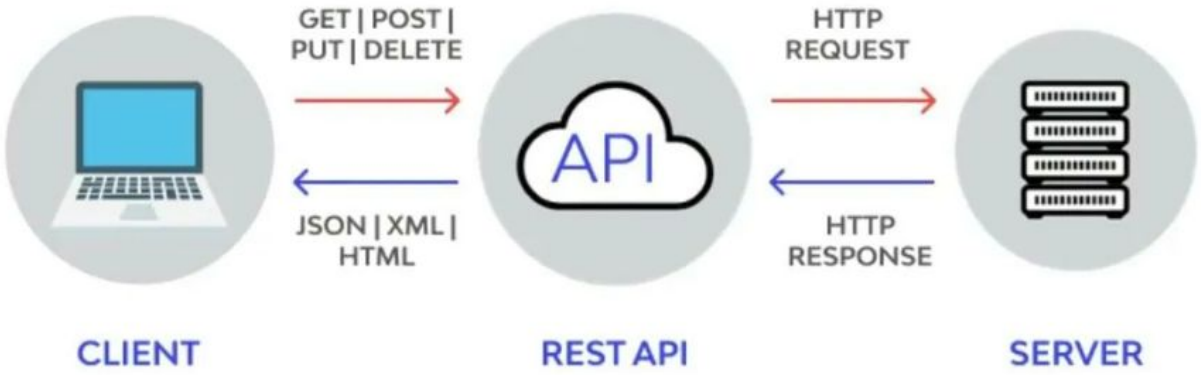
Таким образом, сетевой API функционирует как шлюз между клиентами и ресурсами в Интернете.

RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Клиенты — это пользователи, которые хотят получить доступ к информации в Интернете.
Клиентом может быть человек или программная система, использующая API.

Например, разработчики могут создавать программы, которые получают доступ к данным о погоде из метеосистемы. Также получить доступ к этим данным можно из браузера, посетив веб-сайт с информацией о погоде.



RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Ресурсы — это информация, которую различные приложения предоставляют своим клиентам.
Ресурсы могут быть изображениями, видео, текстом, числами или данными любого типа. Компьютер, который предоставляет ресурсы клиенту, также называется сервером. API позволяет организациям совместно использовать ресурсы и предоставляет веб-службы, обеспечивая безопасность, контроль и аутентификацию. Кроме того, API помогает определить, какие клиенты могут получить доступ к определенным внутренним ресурсам.

RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Representational State Transfer (REST) — это программная архитектура, которая определяет условия работы API.

Первоначально REST создавалась как руководство для управления взаимодействиями в сложной сети, такой как Интернет.

{ REST:API }

Архитектуру на основе REST можно использовать для поддержки высокопроизводительной и надежной связи в требуемом масштабе. Ее можно легко внедрять и модифицировать, обеспечивая прозрачность и кросс-платформенную переносимость любой системы API.

API-интерфейсы, соответствующие архитектурному стилю REST, называются REST API.

Веб-службы, реализующие архитектуру REST, называются веб-службами RESTful.

Как правило, термин RESTful API относится к сетевым RESTful API.

Однако REST API и RESTful API являются взаимозаменяемыми терминами.

REST принципы

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

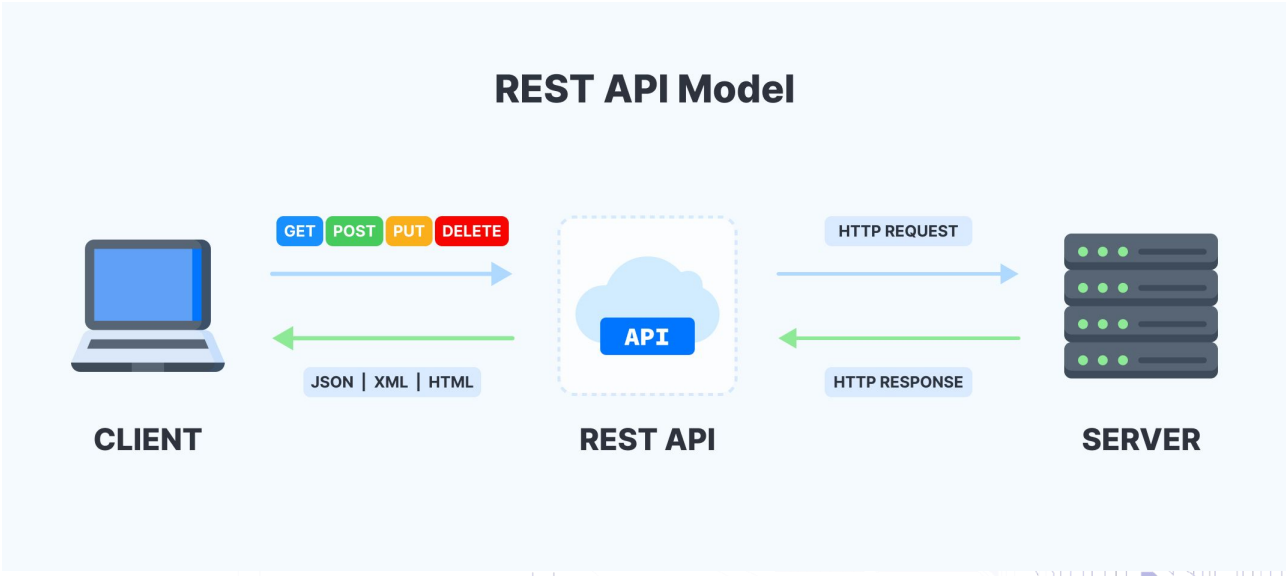
Единый интерфейс

Единый интерфейс является конструктивной основой любого веб-сервиса RESTful.

Это указывает на то, что сервер передает информацию в стандартном формате.

Отформатированный ресурс в REST называется представлением.
Этот формат может отличаться от внутреннего представления ресурса в серверном приложении.

Например, сервер может хранить данные в виде текста, но отправлять их в формате представления HTML.



REST принципы

Единый интерфейс накладывает четыре архитектурных ограничения:

1. Запросы должны идентифицировать ресурсы. Это происходит за счет единого идентификатора ресурсов.

2. Клиенты имеют достаточно информации в представлении ресурса, чтобы при желании изменить или удалить ресурс. Сервер выполняет это условие, отправляя метаданные, которые дополнительно описывают ресурс.

3. Клиенты получают информацию о дальнейшей обработке представлений. Сервер реализует это, отправляя описательные сообщения, где содержатся метаданные о том, как клиент может использовать их оптимальным образом.

4. Клиенты получают информацию обо всех связанных ресурсах, необходимых для выполнения задачи. Сервер реализует это, отправляя гиперссылки в представлении, чтобы клиенты могли динамически обнаруживать больше ресурсов.

REST принципы

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Отсутствие сохранения состояния

В архитектуре REST отсутствие сохранения состояния относится к методу связи, при котором сервер выполняет каждый клиентский запрос независимо от всех предыдущих запросов.

Клиенты могут запрашивать ресурсы в любом порядке, и каждый запрос либо изолирован от других запросов, либо его состояние не сохраняется.

Это конструктивное ограничение REST API подразумевает, что сервер может каждый раз полностью понять и выполнить запрос.

REST принципы

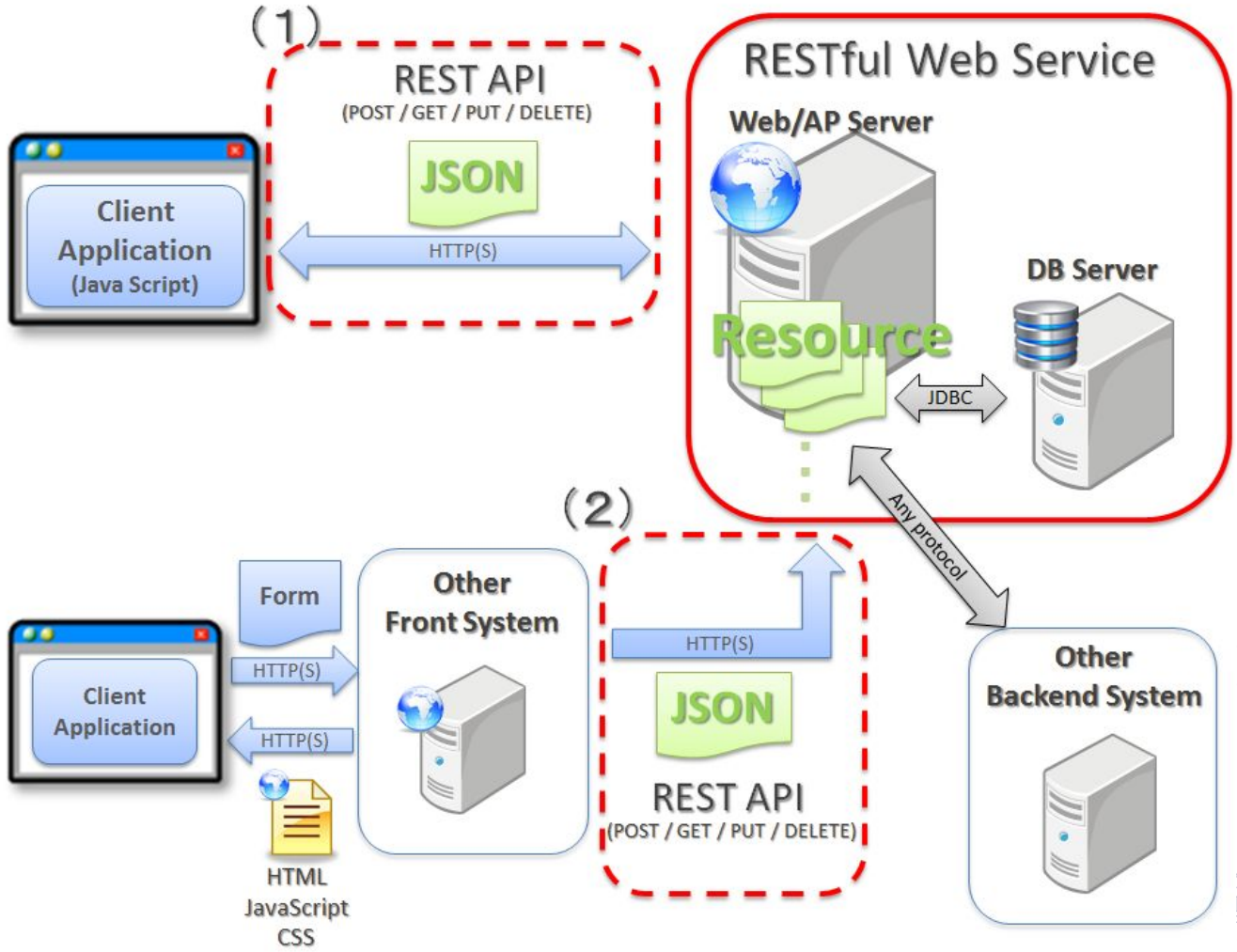
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Многоуровневая система

В многоуровневой системной архитектуре клиент может подключаться к другим авторизованным посредникам между клиентом и сервером и по-прежнему получать ответы от сервера.

Серверы также могут передавать запросы другим серверам. Вы можете спроектировать свою веб-службу RESTful для работы на нескольких серверах с несколькими уровнями (безопасностью, приложениями и бизнес-логикой), совместно выполняющих клиентские запросы.

Эти уровни остаются невидимыми для клиента.



REST принципы

Емкость кэша

Веб-службы RESTful поддерживают кэширование, то есть процесс сохранения некоторых ответов на клиенте или на посреднике для сокращения времени ответа сервера.

Например, вы заходите на веб-сайт с общими изображениями верхнего и нижнего колонтитулов на каждой странице. Каждый раз, когда вы посещаете новую страницу веб-сайта, сервер должен повторно отправлять одни и те же изображения. Чтобы избежать этого, клиент кэширует или сохраняет эти изображения после первого ответа, а затем использует изображения из кэша.

Веб-службы RESTful управляют кэшированием с помощью ответов API, которые определяют себя как кэшируемые или некэшируемые.

REST принципы

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

Код по запросу

В архитектурном стиле REST серверы могут временно расширять или настраивать функциональные возможности клиента, передавая код программного обеспечения.

Например, когда вы заполняете регистрационную форму на каком-либо веб-сайте, ваш браузер сразу же выделяет все допущенные ошибки (например, неверные номера телефонов).

Это происходит благодаря коду, отправленному сервером.

REST преимущества

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Возможность масштабирования

Системы, реализующие REST API, могут эффективно масштабироваться благодаря оптимизации взаимодействия между сервером и клиентом по REST.

Отсутствие сохранения состояния снимает нагрузку с сервера: серверу не нужно сохранять информацию о предыдущих запросах клиента.

Отлаженное кэширование частично или полностью устраняет некоторые взаимодействия между клиентом и сервером.

Перечисленные функции предполагают масштабируемость и не ограничивают пропускную способность, что может привести к снижению производительности.

REST преимущества

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Гибкость

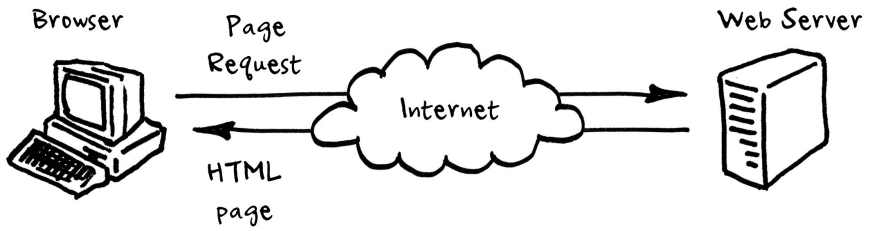
Веб-службы RESTful поддерживают полное разделение клиента и сервера.

Они упрощают и разделяют различные серверные компоненты, чтобы каждая часть могла развиваться независимо.

Изменения платформы или технологии в серверном приложении не влияют на клиентское приложение.

Возможность разделения функций приложения на уровни еще больше повышает гибкость.

Например, разработчики могут вносить изменения в уровень базы данных, не переписывая логику приложения.



REST преимущества

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Независимость

REST API не зависит от используемой технологии.

Вы можете создавать как клиентские, так и серверные приложения на разных языках программирования, не затрагивая структуру API.

Также можно изменить базовую технологию на любой стороне, не влияя на обмен данными.

RESTful API работает так

Базовый принцип работы RESTful API совпадает с принципом работы в Интернете. Клиент связывается с сервером с помощью API, когда ему требуется какой-либо ресурс.

Разработчики описывают принцип использования REST API клиентом в документации на API серверного приложения.

Основные этапы запроса REST API:

1. Клиент отправляет запрос на сервер. Руководствуясь документацией API, клиент форматирует запрос таким образом, чтобы его понимал сервер.

2. Сервер аутентифицирует клиента и подтверждает, что клиент имеет право сделать этот запрос.

3. Сервер получает запрос и внутренне обрабатывает его.

4. Сервер возвращает ответ клиенту. Ответ содержит информацию, которая сообщает клиенту, был ли запрос успешным. Также запрос включает сведения, запрошенные клиентом.

Клиентский запрос RESTful API

API RESTful требует, чтобы запросы содержали следующие основные компоненты:

Уникальный идентификатор ресурса

Сервер присваивает каждому ресурсу уникальный идентификатор ресурса. В случае со службами REST сервер идентифицирует ресурсы с помощью универсального указателя ресурсов (URL).
URL указывает путь к ресурсу. URL аналогичен адресу веб-сайта, который вы вводите в браузере для посещения веб-страницы. URL также называется адресом запроса и четко указывает серверу, что требуется клиенту.

Метод

RESTful API реализуется с помощью протокола передачи гипертекста (HTTP). Метод HTTP сообщает серверу, что ему необходимо сделать с ресурсом.

Клиентский запрос RESTful API

GET

Клиенты используют GET для доступа к ресурсам, расположенным на сервере по указанному URL. Они могут кэшировать запросы GET и отправлять параметры в запросе RESTful API, чтобы сообщить серверу о необходимости фильтровать данные перед отправкой.

POST

Клиенты используют POST для отправки данных на сервер. При этом они включают в запрос представления данных. Отправка одного и того же запроса POST несколько раз имеет побочный эффект — многократное создание одного и того же ресурса.

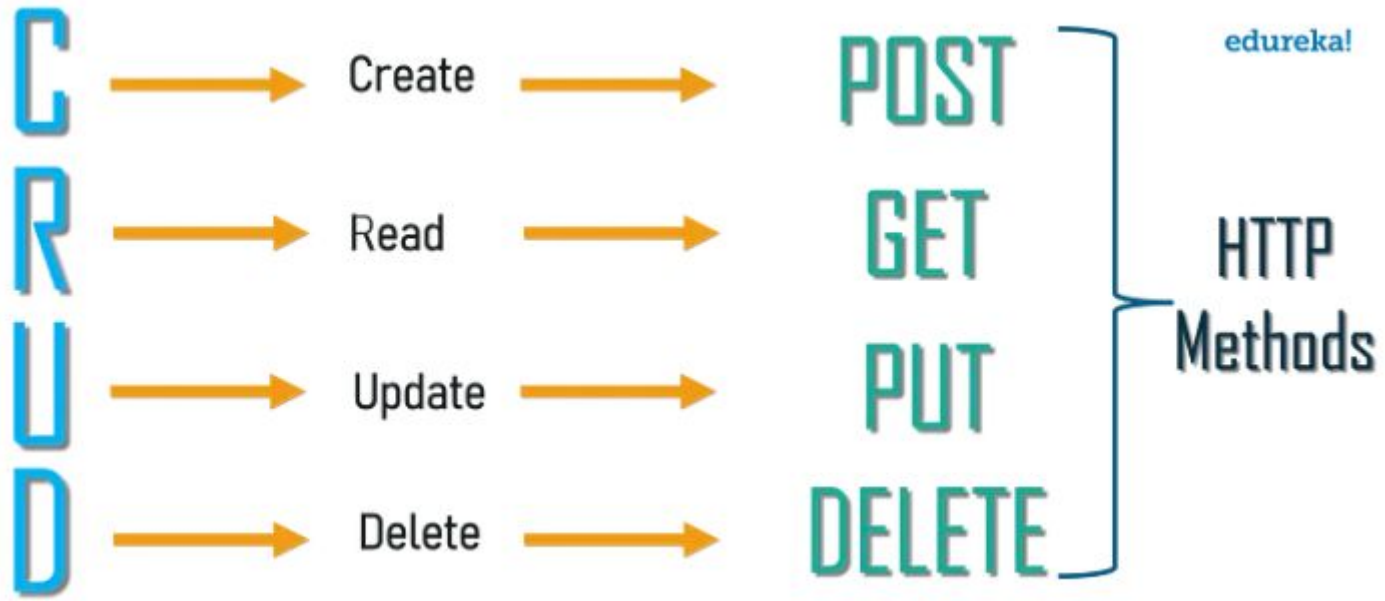
PUT

Клиенты используют PUT для обновления существующих на сервере ресурсов. В отличие от POST, отправка одного и того же запроса PUT несколько раз дает один и тот же результат в веб-службе RESTful.

DELETE

Клиенты используют запрос DELETE для удаления ресурса. Запрос DELETE может изменить состояние сервера. Однако если у пользователя нет соответствующей аутентификации, запрос завершается ошибкой.

CRUD операции



Клиентский запрос RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Заголовки HTTP - это метаданные, которыми обмениваются клиент и сервер.

Например, заголовок запроса указывает формат запроса и ответа, предоставляет информацию о статусе запроса и т. д.

Данные
Запросы REST API могут включать данные для успешной работы POST, PUT и других методов HTTP.

Параметры
Запросы RESTful API могут включать параметры, которые предоставляют серверу более подробную информацию о необходимых действиях.

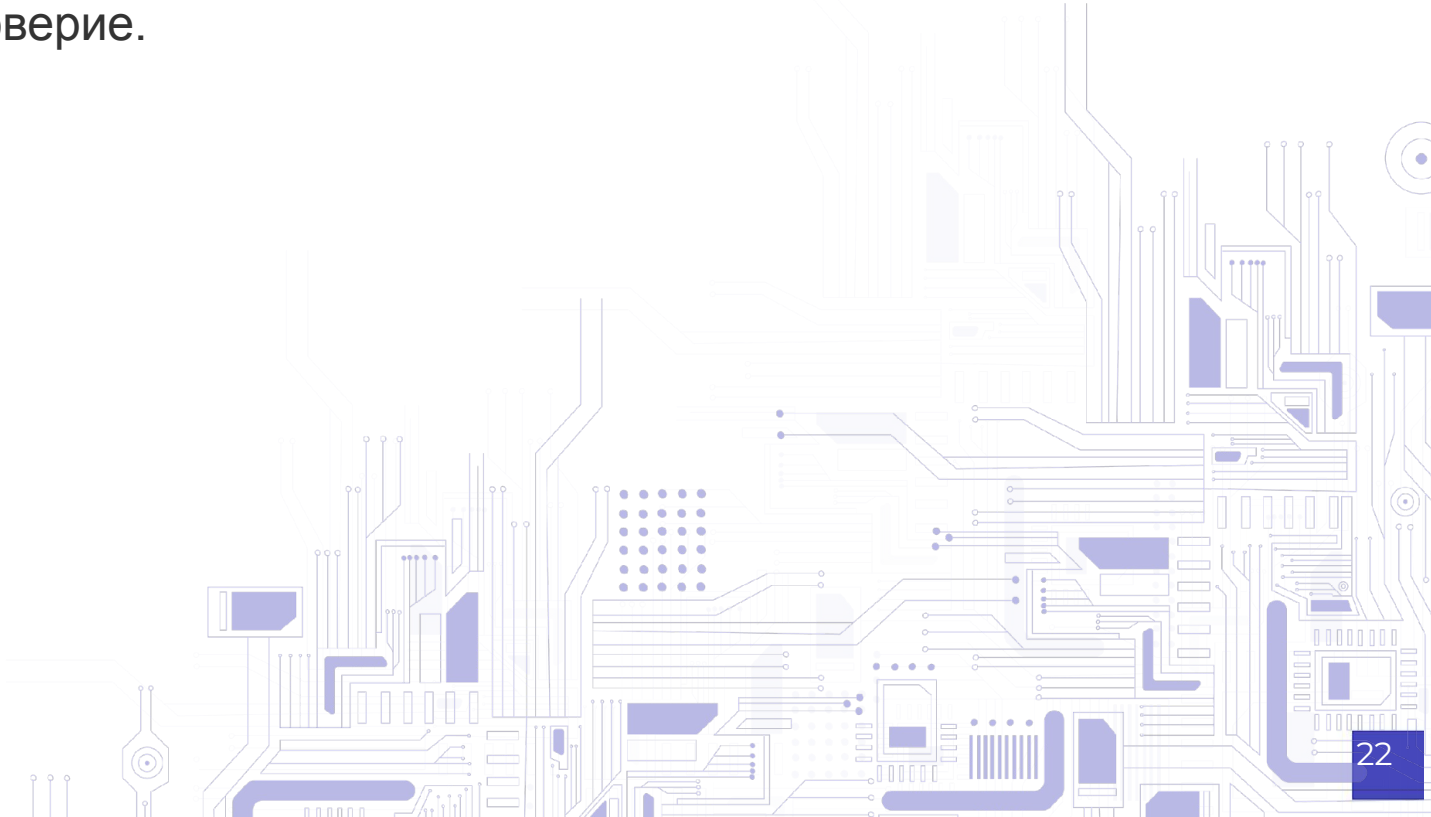
Типы параметров:
- Параметры пути, которые определяют детали URL.
- Параметры запроса, которые запрашивают дополнительную информацию о ресурсе.
- Параметры cookie, которые быстро аутентифицируют клиентов.

Методы аутентификации RESTful API

Веб-служба RESTful должна аутентифицировать запросы для последующей отправки ответа.

Аутентификация — это процесс подтверждения личности.

Например, для подтверждения личности можно использовать удостоверение личности или водительские права. Точно так же клиенты службы RESTful должны подтвердить свою личность серверу, чтобы установить доверие.



Методы аутентификации RESTful API

RESTful API поддерживает четыре распространенных метода аутентификации:

HTTP-аутентификация
HTTP определяет некоторые схемы аутентификации, которые можно использовать при реализации REST API.

Две такие схемы:
Базовая аутентификация. При базовой аутентификации клиент отправляет имя пользователя и пароль в заголовке запроса. Он кодирует их с помощью метода кодирования base64, который преобразует пару имя пользователя–пароль в набор из 64 символов для безопасной передачи.

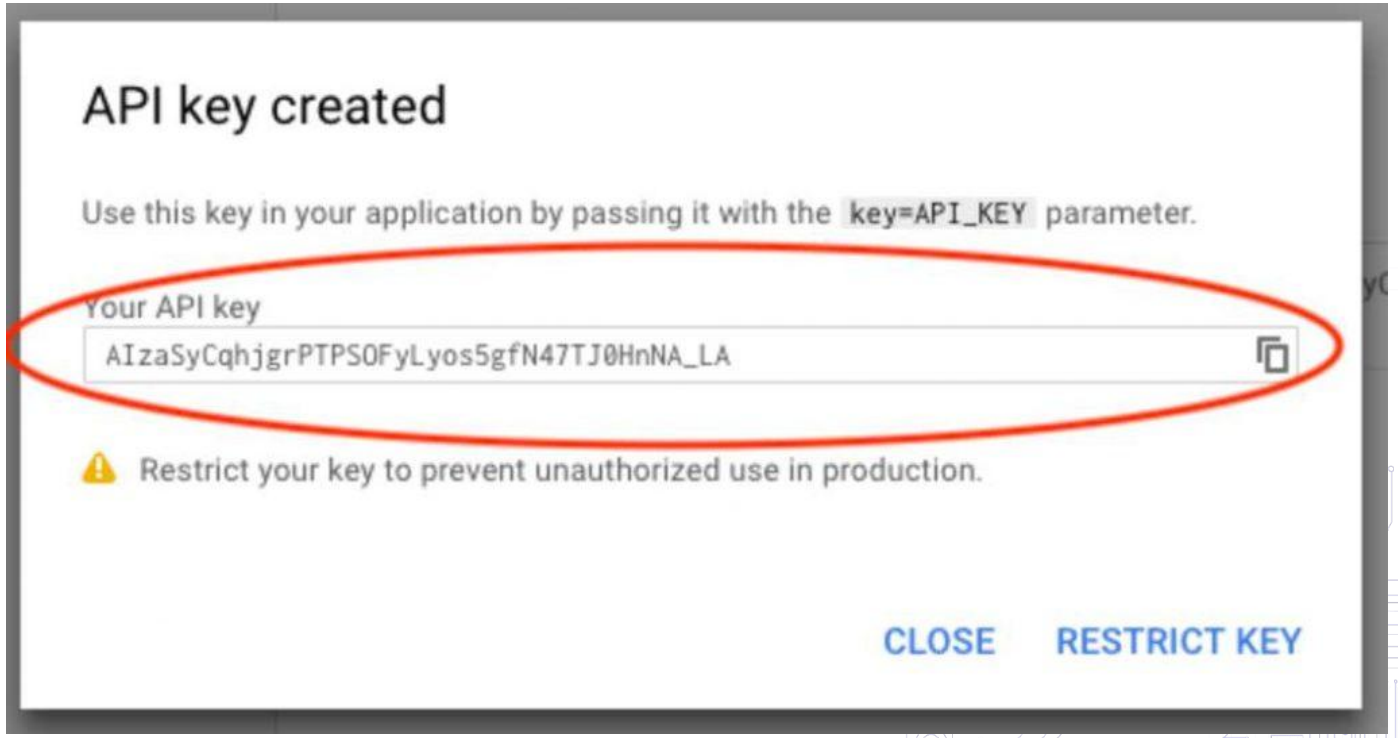
Аутентификация носителя — это процесс предоставления управления доступом носителю токена. Как правило, токен носителя представляет собой зашифрованную строку символов, которую сервер генерирует в ответ на запрос входа в систему. Клиент отправляет токен в заголовках запроса для доступа к ресурсам.

Методы аутентификации RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Ключи API

Ключи API — это еще один вариант аутентификации REST API. При таком подходе сервер генерирует уникальное значение и присваивает его первому клиенту. Всякий раз, когда клиент пытается получить доступ к ресурсам, он использует для верификации уникальный ключ API. Ключи API менее надежны: поскольку клиент должен передавать ключ, повышается вероятность его кражи.

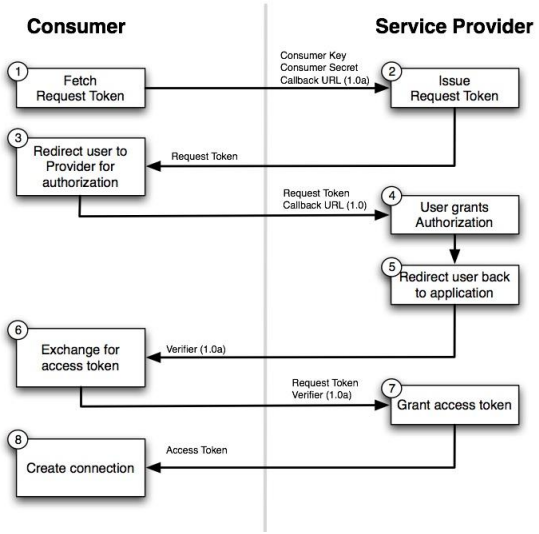


Методы аутентификации RESTful API

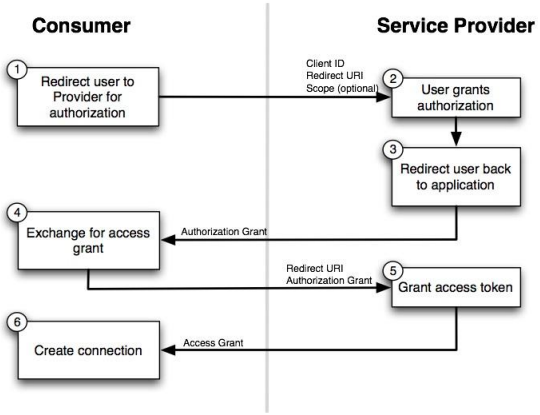
ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

OAuth сочетает в себе пароли и токены для безопасного входа в любую систему. Сначала сервер запрашивает пароль, а затем дополнительный токен для завершения процесса авторизации. Он может проверять токен в любое время, а также через определенный период времени в соответствии с областью и сроком действия.

OAuth 1.0



OAuth 2.0



Ответ от сервера по RESTful API

Ответ сервера содержит следующие компоненты:

Строка состояния
Строка состояния содержит трехзначный код состояния, который сообщает об успешном или неудачном выполнении запроса.

Например, коды 2XX указывают на успешное выполнение, а коды 4XX и 5XX — на ошибки. Коды 3XX указывают на перенаправление URL.

Коды состояния:
- 200: общий ответ об успешном выполнении
- 201: ответ об успешном выполнении метода POST
- 400: неверный запрос, который сервер не может обработать
- 404: ресурс не найден

Ответ от сервера по RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Информационный - 1xx

Этот класс кода состояния указывает на предварительный ответ. По умолчанию в DRF не используются коды состояния 1xx.

HTTP_100_CONTINUE
HTTP_101_SWITCHING_PROTOCOLS
HTTP_102_PROCESSING
HTTP_103_EARLY_HINTS

Успешно - 2xx

Этот класс кода состояния указывает на то, что запрос клиента был успешно получен, понят и принят.

HTTP_200_OK
HTTP_201_CREATED
HTTP_202_ACCEPTED
HTTP_203_NON_AUTHORITATIVE_INFORMATION
HTTP_204_NO_CONTENT
HTTP_205_RESET_CONTENT
HTTP_206_PARTIAL_CONTENT
HTTP_207_MULTI_STATUS
HTTP_208_ALREADY_REPORTED
HTTP_226_IM_USED

Ответ от сервера по RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Перенаправление - 3xx

Этот класс кода состояния указывает на то, что агенту пользователя необходимо предпринять дополнительные действия для выполнения запроса.

HTTP_300_MULTIPLE_CHOICES
HTTP_301_MOVED_PERMANENTLY
HTTP_302_FOUND
HTTP_303_SEE_OTHER
HTTP_304_NOT_MODIFIED
HTTP_305_USE_PROXY
HTTP_306_RESERVED
HTTP_307_TEMPORARY_REDIRECT
HTTP_308_PERMANENT_REDIRECT

Ответ от сервера по RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Ошибка клиента - 4xx

Код состояния класса 4xx предназначен для случаев, когда клиент, похоже, ошибся. За исключением ответа на запрос HEAD, сервер ДОЛЖЕН включать объект, содержащий объяснение ситуации с ошибкой, а также то, является ли она временной или постоянной.

HTTP_400_BAD_REQUEST
HTTP_401_UNAUTHORIZED
HTTP_402_PAYMENT_REQUIRED
HTTP_403_FORBIDDEN
HTTP_404_NOT_FOUND
HTTP_405_METHOD_NOT_ALLOWED
HTTP_406_NOT_ACCEPTABLE
HTTP_407_PROXY_AUTHENTICATION_REQUIRED
HTTP_408_REQUEST_TIMEOUT
HTTP_409_CONFLICT
HTTP_410_GONE
HTTP_411_LENGTH_REQUIRED
HTTP_412_PRECONDITION_FAILED
HTTP_413_REQUEST_ENTITY_TOO_LARGE
HTTP_414_REQUEST_URI_TOO_LONG
HTTP_415_UNSUPPORTED_MEDIA_TYPE
HTTP_416_REQUESTED_RANGE_NOT_SATISFIABLE
HTTP_417_EXPECTATION_FAILED
HTTP_421_MISDIRECTED_REQUEST
HTTP_422_UNPROCESSABLE_ENTITY
HTTP_423_LOCKED
HTTP_424_FAILED_DEPENDENCY
HTTP_425_TOO_EARLY
HTTP_426_UPGRADE_REQUIRED
HTTP_428_PRECONDITION_REQUIRED
HTTP_429_TOO_MANY_REQUESTS
HTTP_431_REQUEST_HEADER_FIELDS_TOO_LARGE
HTTP_451_UNAVAILABLE_FOR_LEGAL_REASONS

Ответ от сервера по RESTful API

Ошибка сервера - 5xx

Коды состояния ответа, начинающиеся с цифры "5", указывают на случаи, когда сервер знает, что он ошибся или не в состоянии выполнить запрос. За исключением ответа на запрос HEAD, сервер ДОЛЖЕН включать объект, содержащий объяснение ситуации с ошибкой, а также то, является ли она временной или постоянной.

HTTP_500_INTERNAL_SERVER_ERROR
HTTP_501_NOT_IMPLEMENTED
HTTP_502_BAD_GATEWAY
HTTP_503_SERVICE_UNAVAILABLE
HTTP_504_GATEWAY_TIMEOUT
HTTP_505_HTTP_VERSION_NOT_SUPPORTED
HTTP_506_VARIANT_ALSO_NEGOTIATES
HTTP_507_INSUFFICIENT_STORAGE
HTTP_508_LOOP_DETECTED
HTTP_509_BANDWIDTH_LIMIT_EXCEEDED
HTTP_510_NOT_EXTENDED
HTTP_511_NETWORK_AUTHENTICATION_REQUIRED

Ответ от сервера по RESTful API

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА УНИВЕРСИТЕТА ИННОПОЛИС

Текст сообщения
Текст ответа содержит представление ресурса.
Сервер выбирает подходящий формат представления на основе содержания заголовков запроса.

Клиенты могут запрашивать информацию в форматах XML или JSON: они определяют запись данных в виде обычного текста.

Например, если клиент запрашивает имя и возраст человека по имени Джон, сервер возвращает представление JSON в следующем формате: '{"name":"John", "age":30}’

Заголовки
Ответ также содержит заголовки или метаданные об ответе. Они дают более подробный контекст ответа и включают такую информацию, как название сервера, кодировка, дата и тип контента.

Полезное

ПЕРЕДОВАЯ ИНЖЕНЕРНАЯ ШКОЛА
УНИВЕРСИТЕТА ИННОПОЛИС

1. Способы описания API

- https://swagger.io/specification/
- https://raml.org

2. Инструменты для тестирования API

- https://www.postman.com
- https://www.soapui.org

3. Большой список открытых API

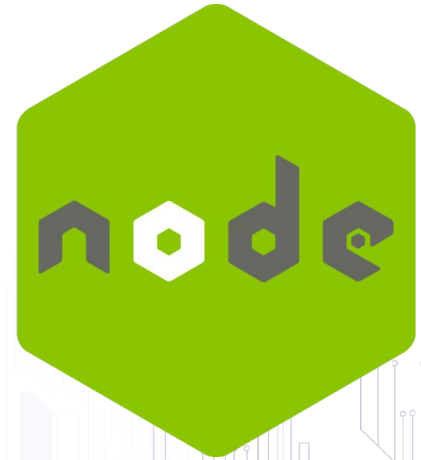
- https://github.com/docops-hq/learnapidoc-ru/blob/m...

NodeJS + Express

Node.js — это опенсорсная кроссплатформенная среда выполнения для JavaScript, которая работает на серверах. С момента выпуска этой платформы в 2009 году она стала чрезвычайно популярной и в наши дни играет весьма важную роль в области веб-разработки.

Express - это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

Node.js с Express — это популярный дуэт, используемый многими приложениями во всем мире



NodeJS + Express

Node.js — это опенсорсная кроссплатформенная среда выполнения для JavaScript, которая работает на серверах. С момента выпуска этой платформы в 2009 году она стала чрезвычайно популярной и в наши дни играет весьма важную роль в области веб-разработки.

Express - это минималистичный и гибкий веб-фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб-приложений.

Передовые инженерные школы

МИНОБРНАУКИ РОССИИ

УНИВЕРСИТЕТ ИННОПОЛИС

Спасибо за внимание