Разработка клиент-серверных приложений

Лекция 4

Преподаватель: к.т.н.,
доцент кафедры АСУ,
Муртазина М.Ш.

2022

??????????????????????

- *Рой Томас Филдинг, Архитектурные стили и проектирование сетевых программных архитектур*, 2000

- https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Принципы REST API

- единый интерфейс
  - Ресурсы должны быть однозначно идентифицированы посредством одного URL-адреса и только с помощью базовых методов сетевого протокола (DELETE, PUT, GET, HTTP).
- разграничение клиента и сервера
  - пользовательский интерфейс и вопросы сбора запросов — на стороне клиента.
  - доступ к данным, управление рабочей нагрузкой и безопасность — на стороне сервера.
- нет сохранения состояния
  - Все клиент-серверные операции должны быть без сохранения состояния.
  - Любое необходимое управление состоянием должно осуществляться на клиенте, а не на сервере.

Принципы REST API

- кэширование всегда разрешено
  - Все ресурсы должны разрешать кэширование, если явно не указано, что оно невозможно.
- многоуровневая система
  - REST API допускает архитектуру, которая состоит из нескольких уровней серверов
- код предоставляется по запросу
  - В большинстве случаев сервер отправляет обратно статические представления ресурсов в формате XML или JSON.
  - Однако при необходимости серверы могут отправлять исполняемый код непосредственно клиенту.

1. Согласованность

Проектирование согласованных данных

- Данные являются основой API – ресурсы, параметры, ответы и их свойства формируют API.
- Все значения, имена, типы, форматы и организация должны быть согласованы, чтобы потребители могли легко понять их.

Проектирование согласованных данных

- В плохо спроектированном банковском API номер счета можно представить в виде свойства **accountNumber** для результата цели «Получить счета», в виде свойства **number** для получения счета и свойства **source** для вводных данных цели «Перевод денег».

Получить счета	accountNumber	accountNumber
Получить счет	number	accountNumber
Перевести деньги	source	sourceAccountNumber

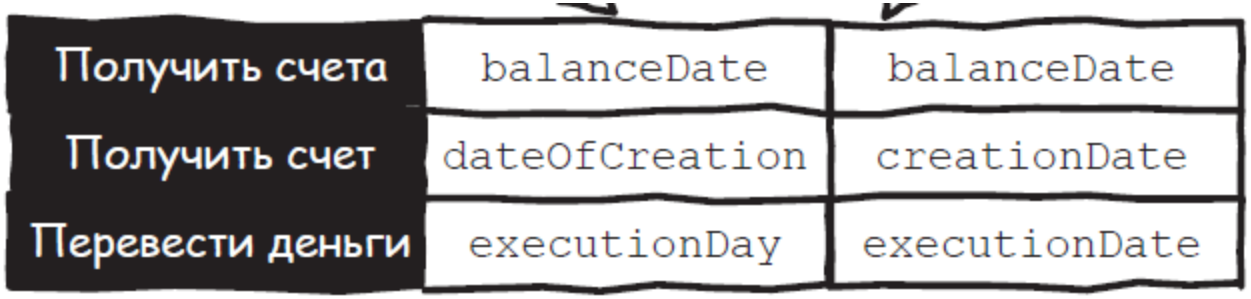
Проектирование согласованных данных

- Как только пользователи замечают номер счета в виде ***accountNumber***, они ожидают увидеть часть информации, всегда обозначаемую ***accountNumber***.

Получить счета	accountNumber	accountNumber
Получить счет	number	accountNumber
Перевести деньги	source	sourceAccountNumber

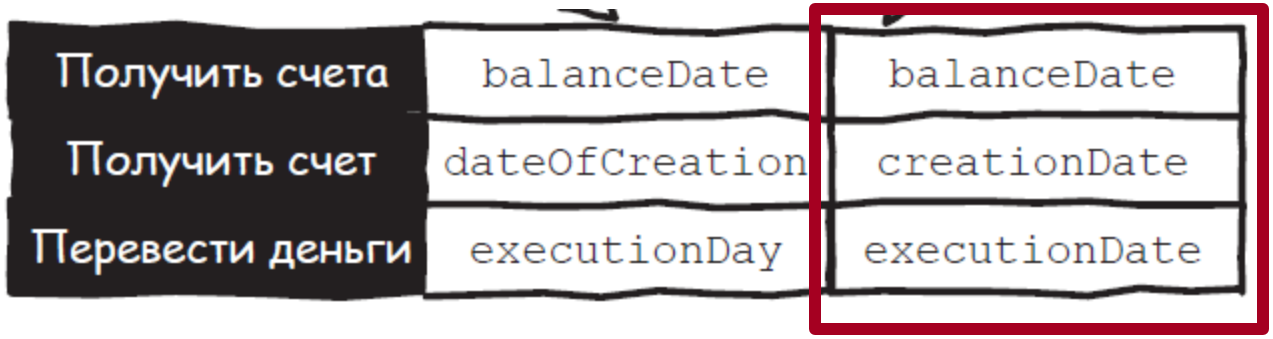
Проектирование согласованных данных

- Это также подходит и для несвязанных данных аналогичного типа или для представления схожих концепций.
- Например, balanceDate, dateOfCreation и executeDay обозначают дату, но первое свойство использует суффикс Date; второе – префикс dateOf, а третье – суффикс Day.



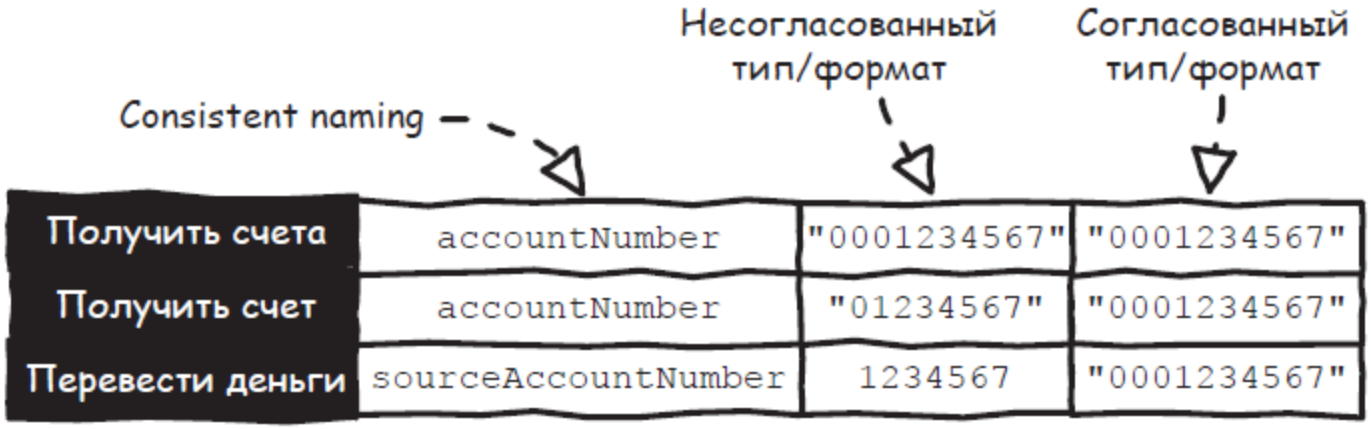
Проектирование согласованных данных

- Использование *общего суффикса или префикса в имени для предо*ставления дополнительной информации о характере того – неплохая практика, пока это делается согласованно. Например, использовать один и тот же суффикс Date для всех дат.



Несогласованные и согласованные типы данных и форматы

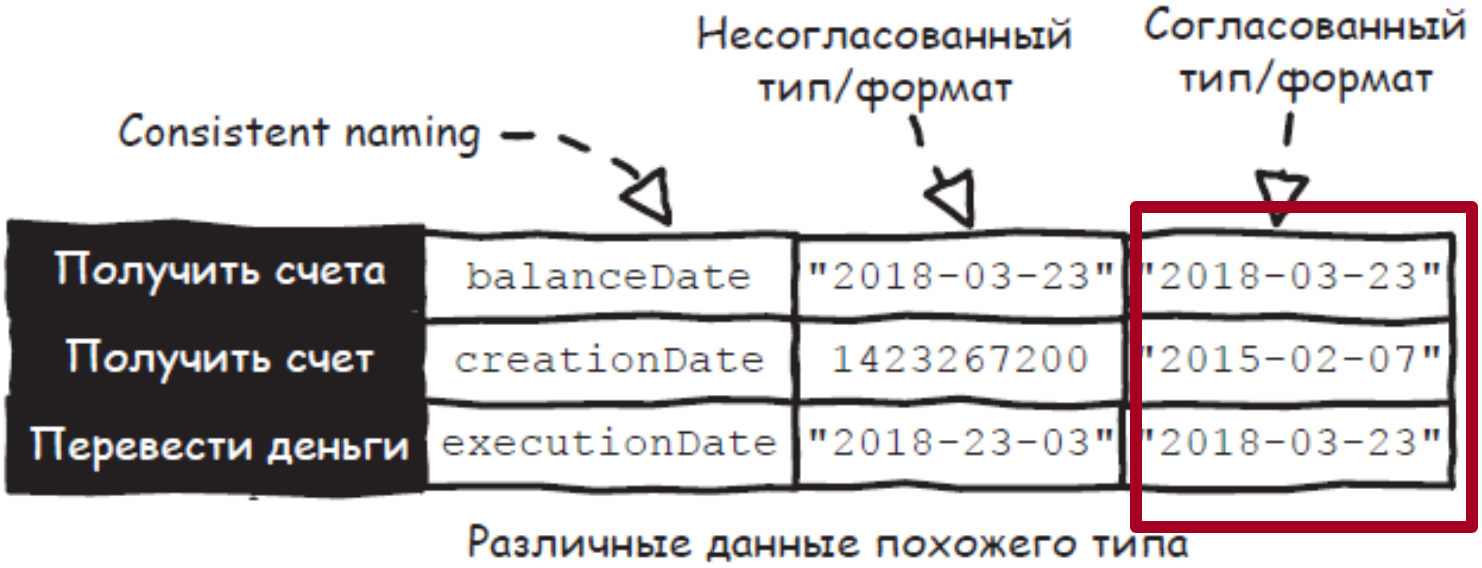
- Номер счета может быть представлен в виде строки «0001234567» и в виде числа 1234567 для входных данных цели «Перевод денег». Такие изменения неизбежно вызовут ошибки на стороне потребителя.



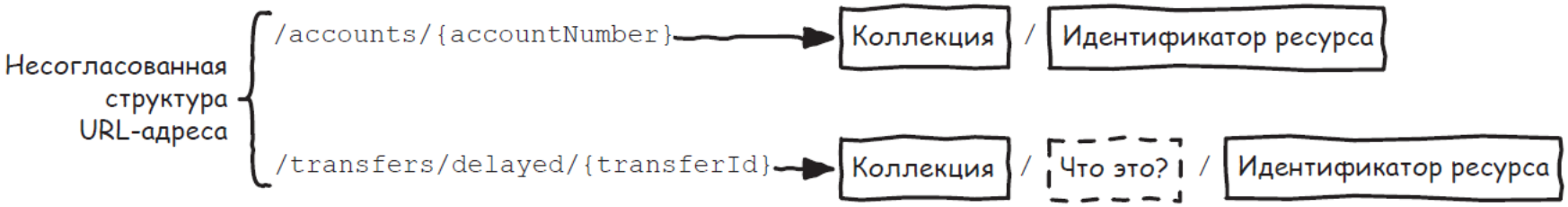
Одни и те же данные в разных контекстах

Несогласованные и согласованные типы данных и форматы

- ISO 8601 (например, 2018-03-23)
- UNIX (например, 1423267200)
- формате ГГГГ-ДД-ММ (например, 2018-23-03)



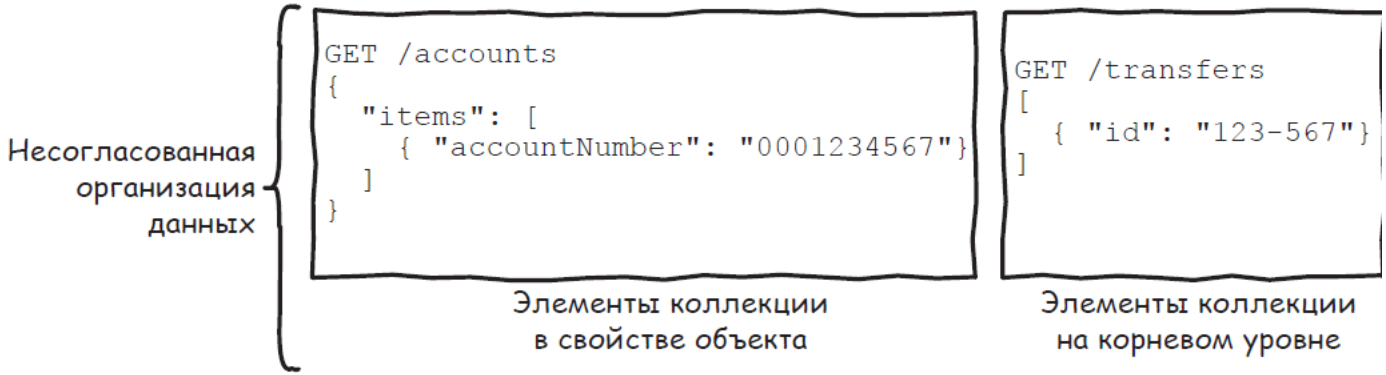
Несогласованная организация



- Согласованным – отличный способ стать предсказуемым, и это помогает потребителям интуитивно использовать ваш API.
- У URL-адресов /account/{accountNumber} и /transfer/delayed/{transferId} разная организация. /transfer/delayed/{transferId} вводит неожиданный уровень между именем коллекции и идентификатором ресурса, делая URL-адрес труднее для понимания. Вместо этого можно было бы использовать /delayed-transfer/{transferId}, например.

Несогласованная организация

- Каждый уровень URL-адреса всегда должен иметь одинаковое значение.
- Когда потребители привыкают к шаблону организации данных, они ожидают, что он будет использоваться повсеместно.
- Если каждый спроектированный ресурс коллекции представлен объектом, содержащим свойство items, являющееся массивом, не нужно проектировать его как простой массив. Потому что потребители будут удивлены этим изменением.



Проектирование согласованных целей

Поведение API определяется его целями:

- они ожидают входных данных и возвращают сообщения об успехе или ошибках,
- все эти цели можно использовать для формирования различных потоков.

Проектирование согласованных целей

Например, при перечислении транзакций на счете с помощью запроса

GET /accounts/{accountId}/transactions

может быть полезно иметь возможность получать только те транзакции, которые произошли между двумя датами.

fromDate=2021-02-07 и toDate=2021-03-17

Проектирование согласованных целей

- То же самое касается сообщений, возвращаемых в случае успеха или ошибки.
- Можно использовать только небольшое подмножество всех существующих кодов состояния HTTP
- Например, если все цели, ведущие к созданию чего-либо, возвращают код состояния **200 OK** без использования таких кодов, как 201 Created или 202 Accepted, было бы разумно избегать введения новых целей, возвращающих разные успешные коды состояния HTTP вместо обычного ответа 200 OK.

Четыре уровня согласованности

- уровень 1 – согласованность в рамках API;
- уровень 2 – согласованность API-интерфейсов организации/компании/команды;
- уровень 3 – согласованность с областью (областями) API (например, в каких единицах измерения возвращать ответ);
- уровень 4 – согласованность с остальным миром (общепринятые практики, стандарты).

Ограничения REST: единообразие интерфейса

- Архитектурный стиль REST гласит, что «…все взаимодействия должны руководствоваться концепцией идентифицированных ресурсов, которыми манипулируют посредством представления состояний ресурсов и стандартных методов».
- Стандартный метод – в действительности мощная концепция, которая помогает обеспечить согласованность. В основном весь протокол HTTP (особенно HTTP-методы, а также коды состояния HTTP) обеспечивает согласованную структуру для REST API, что делает их полностью предсказуемыми.

2. Адаптируемость

АДАПТИВНОГО ДИЗАЙНА API

три распространенных способа создания адаптивного дизайна API:

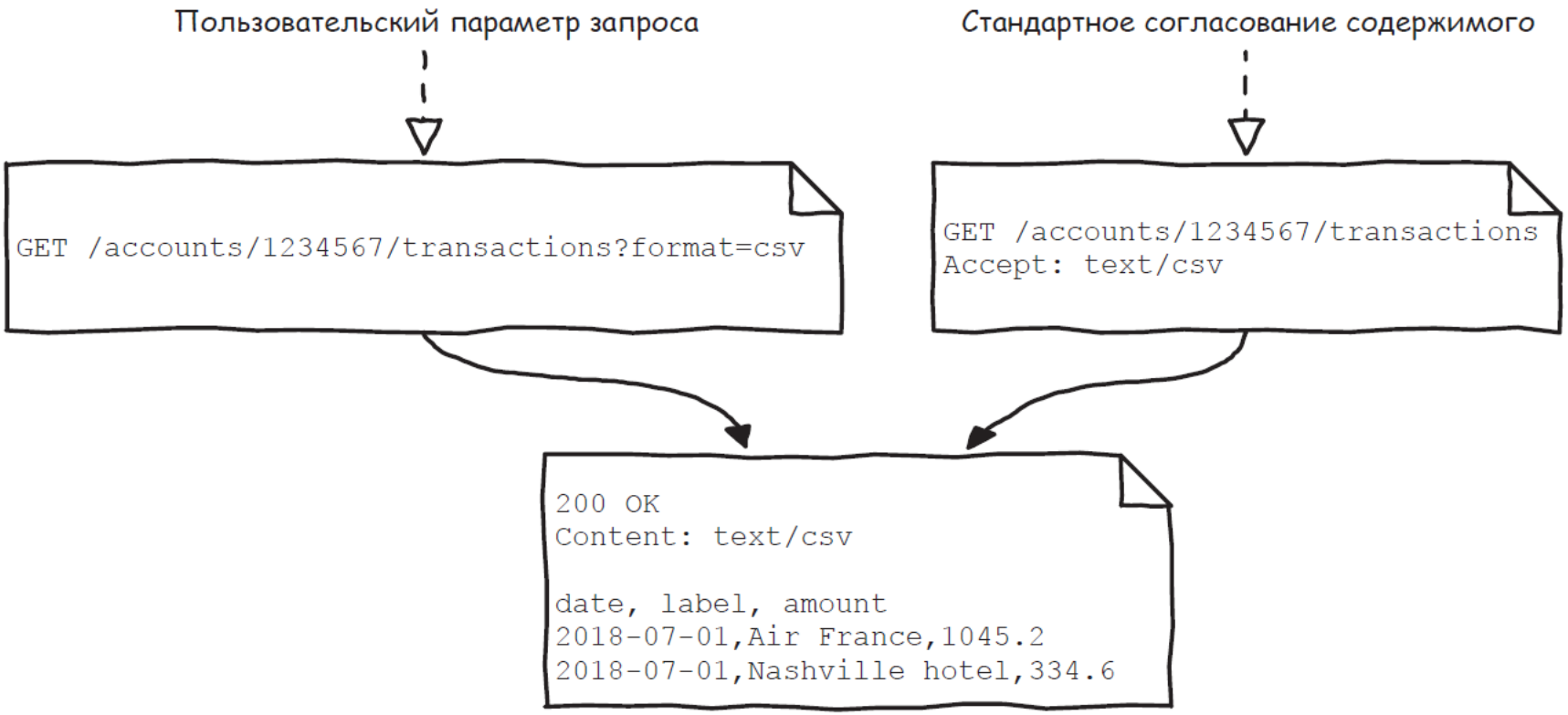
- предоставление и принятие различных форматов;
- интернационализация и локализация;
- обеспечение фильтрации, нумерации страниц и сортировки.

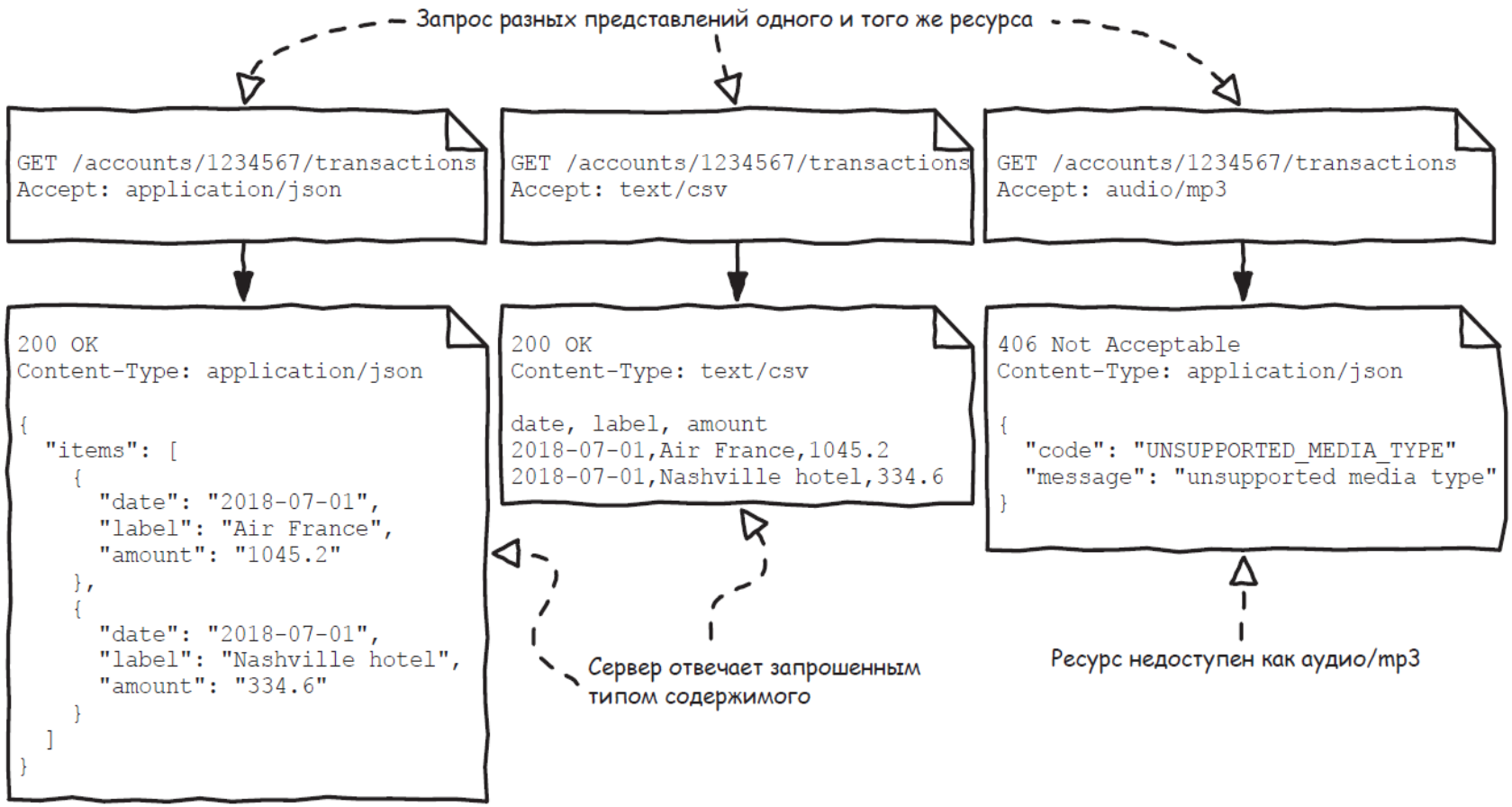
Предоставление и принятие разных форматов

GET /accounts/{accountId}/transactions?format=CSV

GET /accounts/{accountId}/transactions?format=JSON

Два варианта запроса списка транзакций в виде CSV-документа




Запрос разных представлений одного и того же ресурса
GET /accounts/1234567/transactions
Accept: application/json
GET /accounts/1234567/transactions
Accept: text/csv
GET /accounts/1234567/transactions
Accept: audio/mp3
200 OK
Content-Type: application/json
{
"items": [
{
"date": "2018-07-01",
"label": "Air France",
"amount": "1045.2"
},
{
"date": "2018-07-01",
"label": "Nashville hotel",
"amount": "334.6"
}
]
}
200 OK
Content-Type: text/csv
date, label, amount
2018-07-01,Air France,1045.2
2018-07-01,Nashville hotel,334.6
406 Not Acceptable
Content-Type: application/json
{
"code": "UNSUPPORTED_MEDIA_TYPE"
"message": "unsupported media type"
}
Сервер отвечает запрошенным типом содержимого
Ресурс недоступен как аудио/mp3

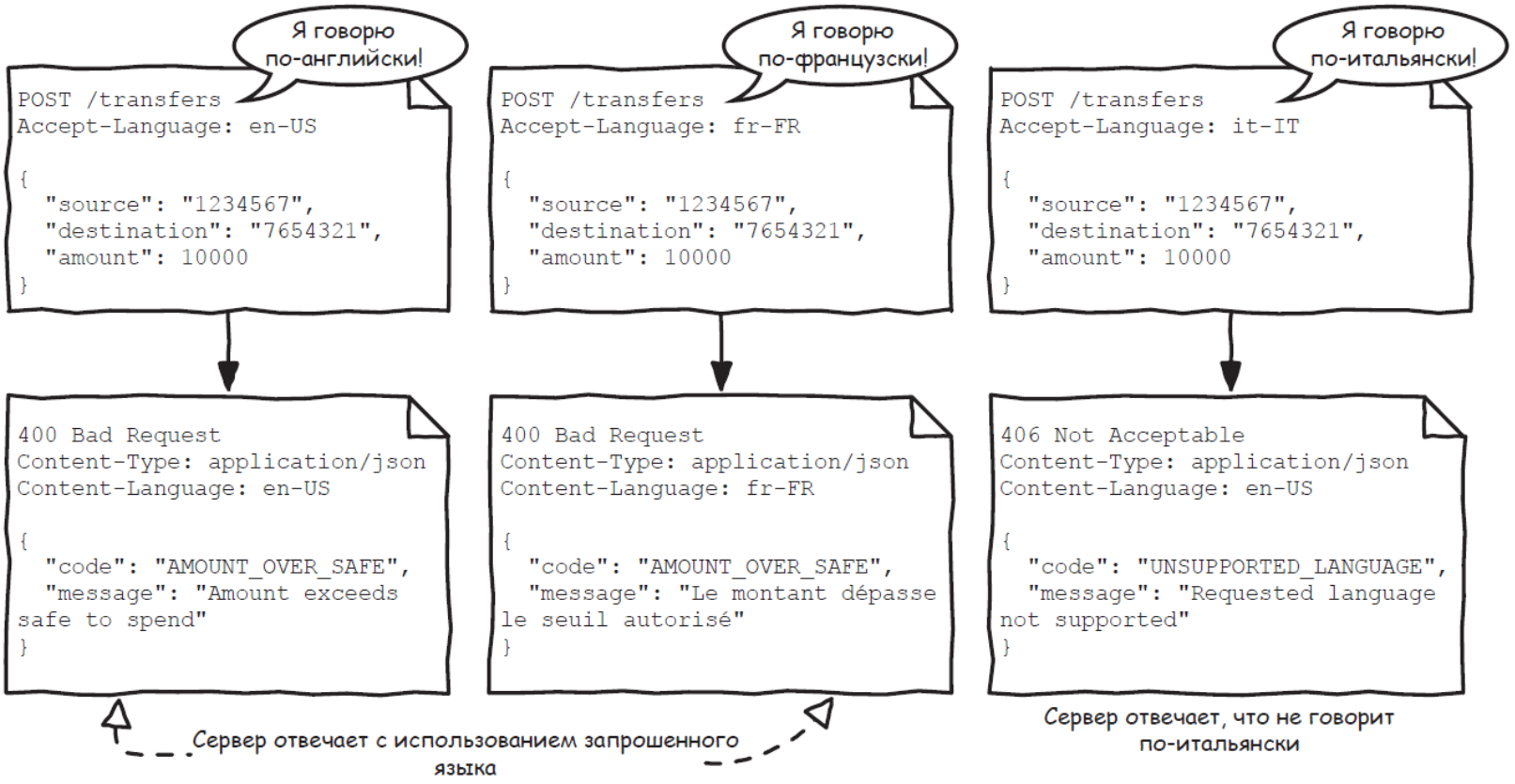
Ограничения REST: единообразие интерфейса

- Архитектурный стиль REST гласит, что все взаимодействия должны руководствоваться концепцией идентифицированных ресурсов, которыми манипулируют через представления состояний ресурсов и стандартные методы и которые предоставляют все метаданные, необходимые для понимания представлений и знания того, что можно делать с этими ресурсами.
- Один ресурс может быть предоставлен потребителем и возвращен поставщиком во многих разных форматах с помощью, например, различных представлений, таких как JSON или CSV. Это обеспечивает мощный механизм, который позволяет REST API адаптироваться к своим потребителям и, следовательно, быть предсказуемым. Возвращенные заголовки также предоставляют информацию о фактическом формате возвращаемого представления.

Интернационализация и локализация

- Интернационализация – это механизм, позволяющий программному обеспечению, приложению или API выполнять локализацию.
- Локализация – это способность справляться с адаптациями к локали, которая в основном состоит из языка и региона или страны.

Интернационализация и локализация



Интернационализация и локализация

- Для REST API *интернационализация означает* понимание того, что заголовок Accept-Language: fr-FR указывает, что потребитель хочет получить локализованный ответ с использованием французского языка и соглашений.
- На стороне сервера это означает, что,
  - если запрашиваемая локализация поддерживается, содержимое будет возвращено в локализованном виде вместе с заголовком Content-Language: fr-FR.
  - tсли она не поддерживается, сервер возвращает код состояния 406 Not Acceptable.

Интернационализация и локализация

- Для банковского API *локализация означает возможность* обрабатывать локаль fr-FR.
- Возвращаемые данные должны быть на французском языке с использованием метрической системы, а документ PDF (если задан этот тип формата для ответа) должен быть создан с использованием размера A4, а не формата, принятого, например, в США.

Фильтрация, разбиение на страницы и сортировка

- Банковский счет, который был открыт в течение многих лет, может иметь тысячи транзакций. Клиент, который хочет получить транзакции по счету с помощью банковского API, вряд ли горит желанием увидеть все эти транзакции сразу и предпочитает получить определенное количество.

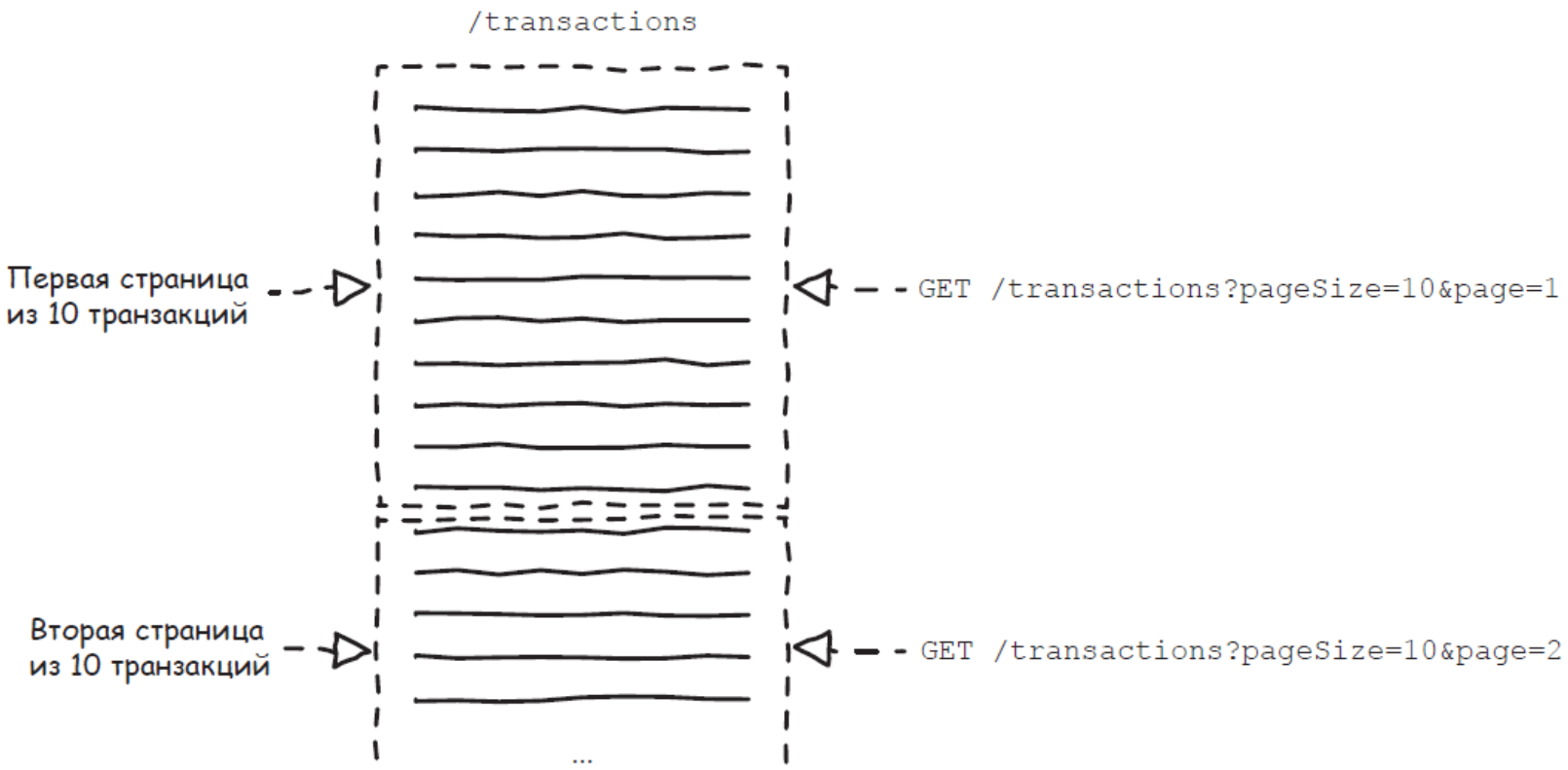
Фильтрация, разбиение на страницы и сортировка

- Например, вывести только транзакции, которые были отнесены к категории *ресторанных транзакций*.
- Чтобы получить эти конкретные транзакции, потребитель может отправить запрос

GET /account/1234567/transactions?category=restaurant.

Параметр запроса category используется здесьдля фильтрации транзакций и возврата только тех транзакций, которыеотносятся к категории «ресторан».

Фильтрация, разбиение на страницы и сортировка



Фильтрация, разбиение на страницы и сортировка

- По умолчанию транзакции идут по порядку от самых последних до самых старых
- Потребители могут также захотеть отсортировать транзакции в порядке убывания (сначала идут самые крупные суммы) и в хронологическом порядке (от самых старых до самых последних).
- Чтобы получить такой список, они могут отправить запрос:

  GET /accounts/1234567/transactions?sort=-amount,+date

- Параметр запроса sort определяет способ сортировки списка транзакций. Он содержит список пар типа «направление–свойство». Направление + (плюс) – по возрастающей и – (минус) – по нисходящей. Значения -amount и +date указывают серверу сортировать транзакции по сумме в порядке убывания и по дате в порядке возрастания.

Фильтрация, разбиение на страницы и сортировка

- Пример совместного использования фильтрации, разбиения на страницы и сортировки

GET /accounts/1234567/transactions?
category=restaurant&sort=-amount,+date&page=3

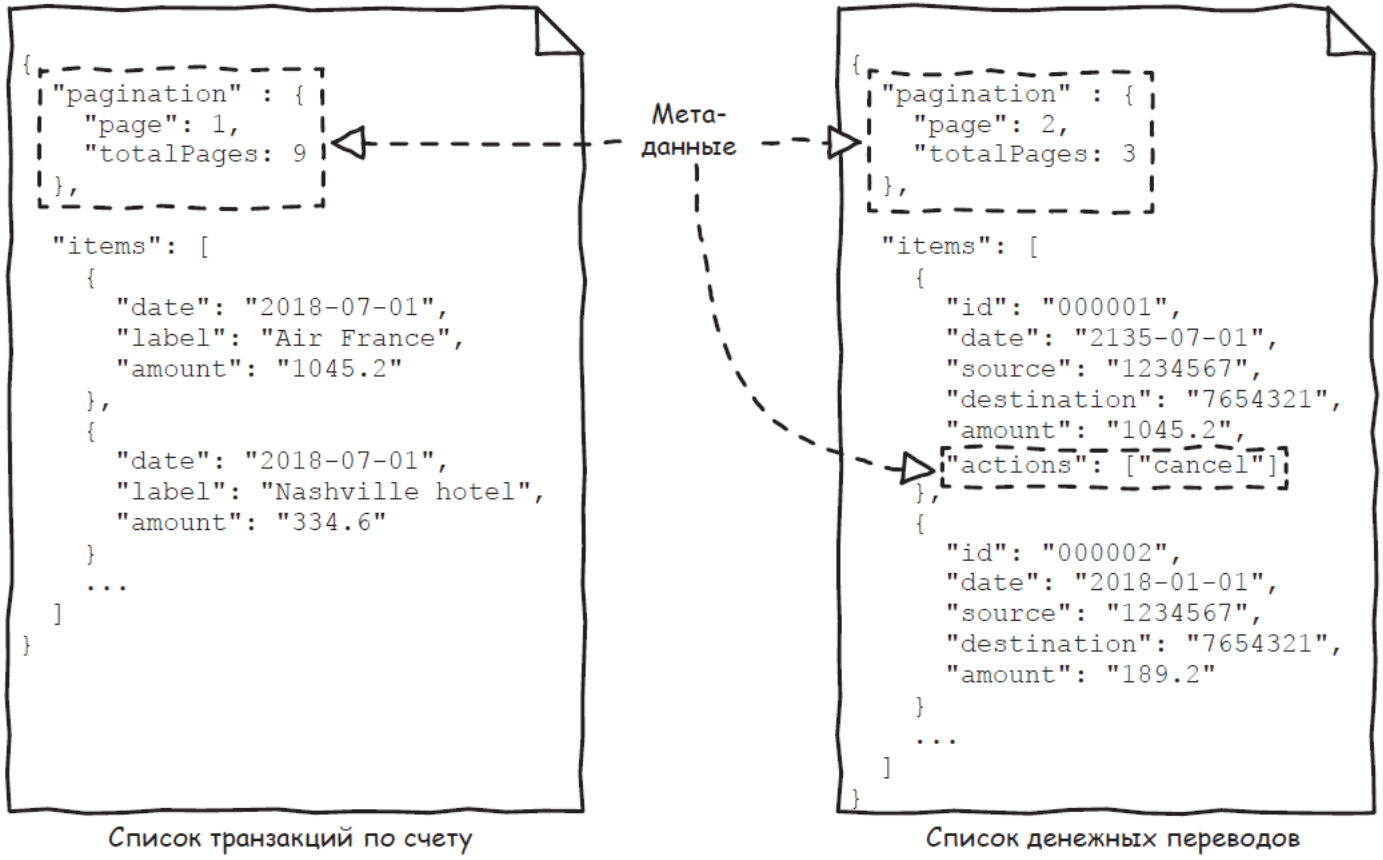
- Вернуть третью страницу транзакций по ресторанам с указанием суммы в порядке убывания и дат в порядке возрастания.

3. Видимость

Быть видимым

- Это делается путем предоставления дополнительных данных различными способами, но обнаруживаемость также можно улучшить с помощью преимуществ используемого протокола.
- У REST API эта особенность заложена, потому что они используют URL-адреса и протокол HTTP.

Предоставление метаданных



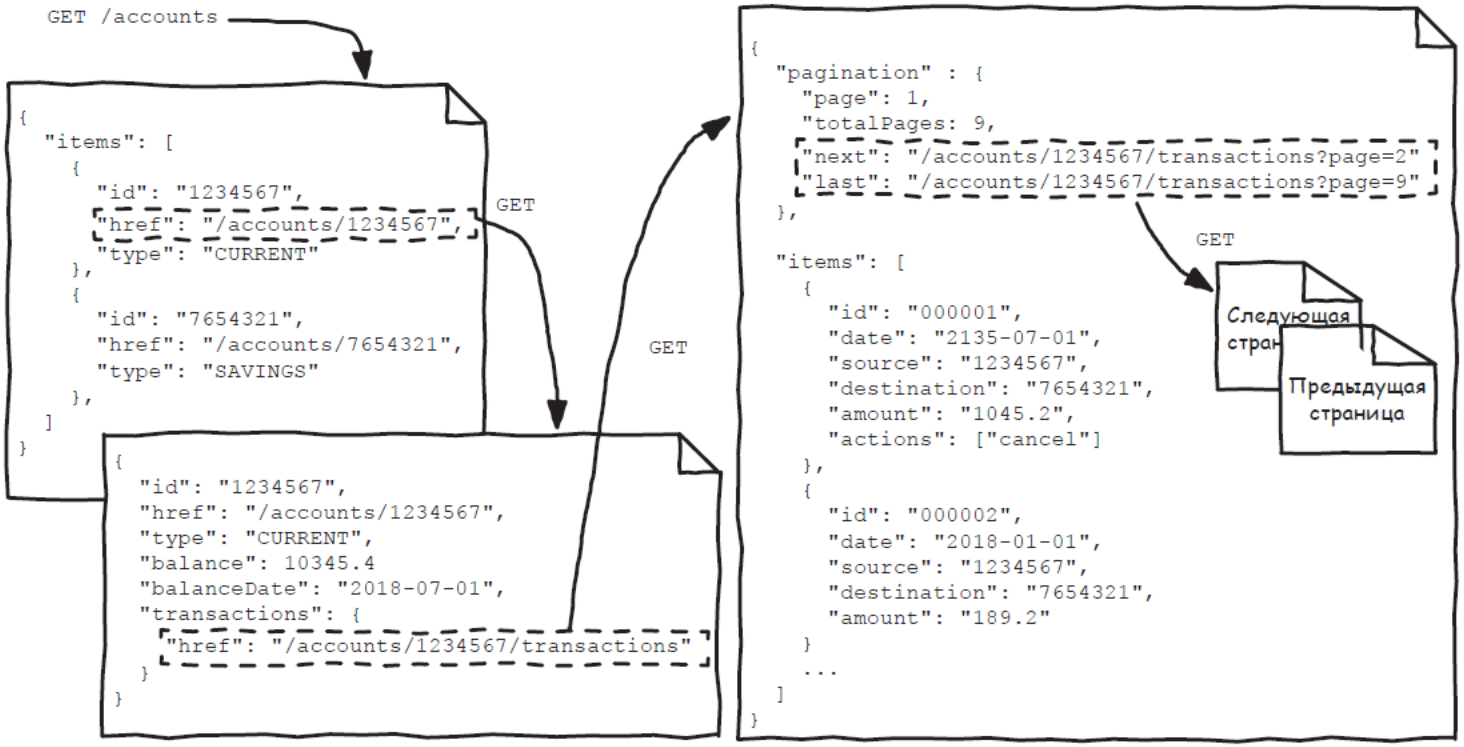
Список транзакций по счету

Список денежных переводов

Предоставление метаданных

- API может возвращать метаданные вместе с данными, чтобы помочь потребителям узнать, где они находятся и что они могут сделать.
- API можно использовать и без этой дополнительной информации, но метаданные значительно облегчают его использование.

Создание гипермедиа-API



Создание гипермедиа-API

- REST API предоставляют ссылки так же, как и веб-страницы. Это облегчает обнаружение API и, как вы увидите позже, обновление API.
- Стандартного способа предоставления этих гиперметаданных не существует, но есть общепринятые практики, по большей части основанные на том, как ссылки представлены на HTML-страницах и в протоколе HTTP.

Создание гипермедиа-API

- В гиперметаданных обычно используются такие имена, как href, links или _links. Хотя здесь нет стандарта, было определено несколько форматов.
- Наиболее известные из них – HAL, Collection+JSON, JSON API, JSON-LD, Hydra и Siren.
- Эти форматы поставляются с различными ограничениями относительно структуры данных.

Создание гипермедиа-API

Банковский счет в виде документа HAL

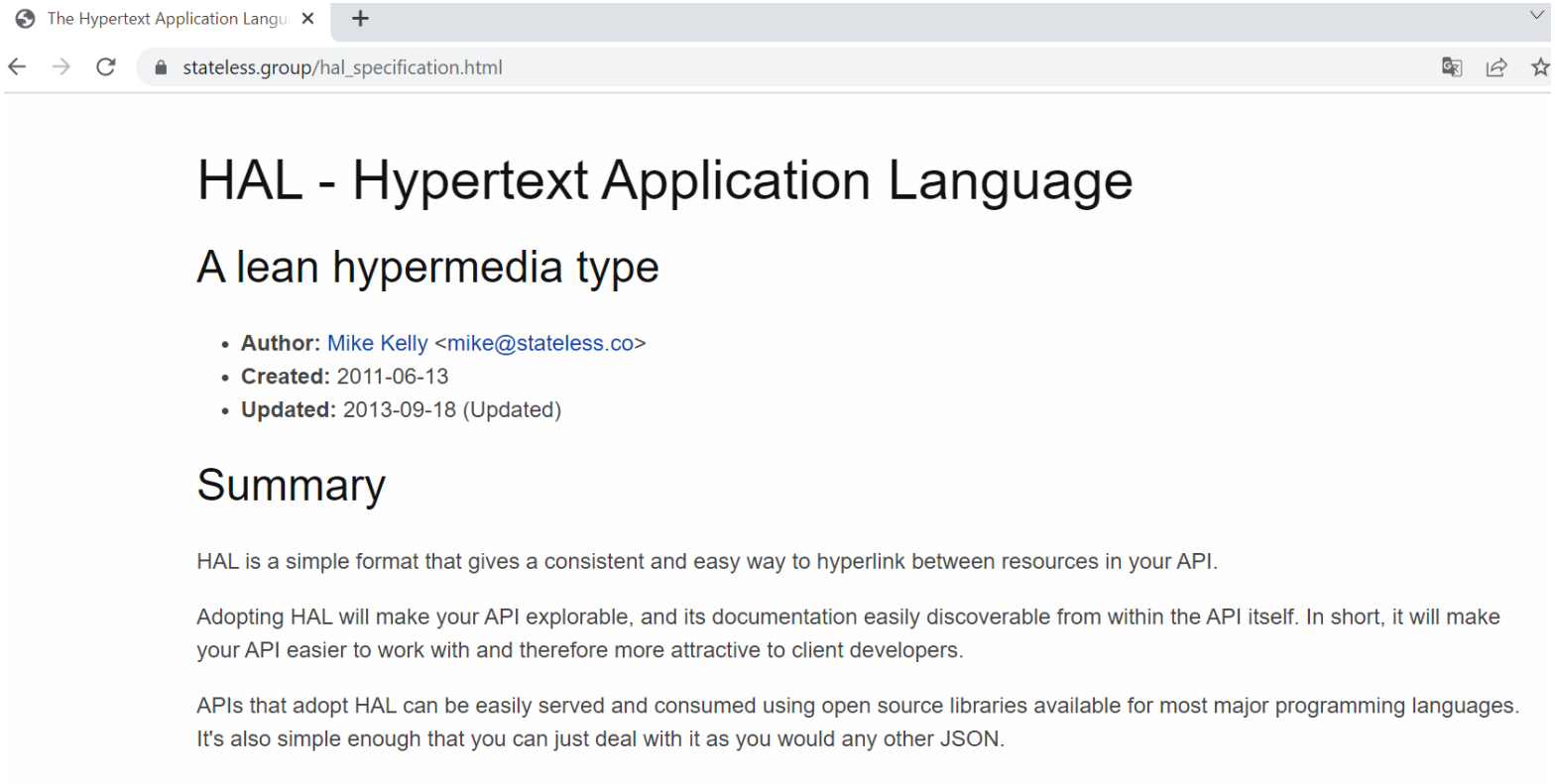
```
{
   "_links" : {
      "self": {
         "href": "/accounts/1234567" ①
      },
      "transactions": {
         "href":"/accounts/1234567/transactions" ②
      }
   }
   "id": "1234567",
   "type": "CURRENT",
   "balance": 10345.4
   "balanceDate": "2018-07-01"
}
```

① Ссылка на сам ресурс банковского счета.

② Ссылка на транзакции банковского счета.

Создание гипермедиа-API

- **Hypertext Application Language** (HAL, **Язык Гипертекстовых Приложений**) — это разрабатываемый (Internet Draft, или **ID**) стандарт для определения гипермедиа, таких как ссылки на внешние ресурсы в форматах JSON или XML.
- Был впервые предложен в июне 2012 года для использования с JSON и с тех пор стал доступен в двух вариантах, JSON и XML.



https://stateless.group/hal_specification.html

Создание гипермедиа-API

- Базовый документ HAL имеет свойство links, содержащее доступные ссылки.
- Каждая ссылка – это объект, идентифицируемый связями (или _rel) с текущим ресурсом. Связь self используется для ссылки на ресурс.
- Объект ссылки содержит как минимум свойство href с полным или относительным URL-адресом.

Ограничения REST: единообразие интерфейса

- Архитектурный стиль REST гласит, что все взаимодействия должны руководствоваться концепцией идентифицированных ресурсов, которыми манипулируют посредством представлений состояний ресурсов и стандартных методов, и предоставляет все метаданные, необходимые для понимания представлений и знания того, что можно сделать с этими ресурсами.
- API-интерфейсы REST – это гипермедиа-API, которые предоставляют все метаданные, необходимые, для того чтобы потребители могли путешествовать по ним, как по сайту, чтобы облегчить их обнаружение.
- Метаданные могут использоваться для описания не только связей между ресурсами, но и между доступными операциями. Эта часть ограничения унифицированного интерфейса архитектурного стиля REST носит название Hypermedia as the Engine of Application State (часто используется вариант в виде непроизносимой аббревиатуры HATEOAS).

Спасибо за внимание!

Муртазина Марина Шамильевна

Контакты

Для **быстрой связи** можно задавать вопросы в VK: https://vk.com/id232309638