Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

**(ПНИПУ)**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

**о научно-исследовательской работе на тему**

Разработка веб-сервера, используя архитектурный стиль REST

Выполнил: студент группы РИС-17-1б

Ерохин Николай Владимирович

Проверил: доцент кафедры ИТАС

Курушин Даниил Сергеевич

**г. Пермь, 2021**

**РЕФЕРАТ**

Отчёт 16 с., 7 источн.

REST, ВЕБ-СЕРВЕР, ВЕБ-СЕРВИС, ПРОТОКОЛ, КЛИЕНТ, РАСПРЕДЕЛЁННАЯ СЕТЬ.

Целью научно-исследовательской работы является выбор средств и технологий разработки сервиса подачи заявок на куплю/продажу для возможности реализовать приложение-клиент данного сервиса.

Объектом исследования в рамках данной работы выбор средств и технологий разработки сервиса подачи заявок на куплю/продажу.

Предметом исследования является архитектурный стиль REST.

В процессе работы будет произведена попытка рассмотрения альтернатив для реализации веб-сервера для сервиса подачи заявок на куплю/продажу, а также выбор средств для данной реализации.

В результате работы были выявлены преимущества архитектурного стиля REST и его наилучшая область применения.

**СОДЕРЖАНИЕ**

[ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ 4](#_Toc70593325)

[ВВЕДЕНИЕ 5](#_Toc70593326)

[1 Архитектурный стиль REST 6](#_Toc70593327)

[2 Cмежные с архитектурным стилем REST технологии 8](#_Toc70593328)

[3 Описание веб-сервиса и выбор технологий 10](#_Toc70593329)

[ЗАКЛЮЧЕНИЕ 15](#_Toc70593330)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc70593331)

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

REST (Representational State Transfer) – архитектурный стиль взаимодействия компонентов распределённого приложения в компьютерной сети.

Клиент-сервер – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемых серверами, и заказчиками услуг, называемых клиентами.

Веб-сервер (в контексте разработки программного обеспечения) – программное обеспечение, выполняющее функции поставщика услуг, выполняя приём и обработку запросов, а также выдачу ответов на них.

Веб-сервис – комплекс программного обеспечения, предоставляющий доступ к данным и функциям системы в распределённой компьютерной сети.

ВВЕДЕНИЕ

В настоящее время активно заполняется ниша программных приложений, которые используют сетевую архитектуру «клиент-сервер». По данным отчёта аналитиков Sensor Tower, количество приложений в 2020 году в Google play достигло 2,7 миллиона приложений с ростом 5,85% за второй квартал года. [4] Данная архитектура позволяет создавать системы, сложность которых не ограничена лишь мощностями клиентской рабочей станции, поэтому знание основ и методологий построения таких систем позволит успешно конкурировать на рынке программных приложений. Хотя протоколы и архитектурные стили разработаны относительно давно, многие разработчики программного обеспечения в полной мере не знают или не понимают их, ошибочно принимают те или иные решения, и, как следствие, получают растущую сложность своих продуктов с точки зрения разработки и поддержки, что отрицательно сказывается на их конкурентоспособности. [7]

**Объектом исследования** в рамках данной работы выбор средств и технологий разработки сервиса подачи заявок на куплю/продажу.

**Предметом исследования** является архитектурный стиль REST.

**Задачи**, которые необходимо решить для достижения цели:

* выявить условия, при которых наиболее подходящим архитектурным стилем является REST;
* сделать обзор инструментов, с помощью которых можно разработать веб-сервер;
* выбрать наиболее подходящий для разработки инструмент.

1 Архитектурный стиль REST

REST – архитектурный стиль взаимодействия компонентов распределённого приложения в компьютерной сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой медиа-системы. В определённых случаях, таких как интернет-магазины, поисковые системы и прочие системы, основанные на данных, это приводит к повышению производительности и упрощению архитектуры. [1]

Свойства архитектуры REST:

* 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. [2]

Как и любой архитектурный стиль, REST накладывает ограничения для построения распределённых систем. Выполнение этих ограничений обязательно. Накладываемые ограничения определяют работу сервера в том, как он может обрабатывать и отвечать на запросы клиентов. Действуя в рамках этих ограничений, система приобретает свойства архитектуры REST. Любое нарушение этих ограничений не позволяет считать систему как построенную по архитектуре REST:

* Модель «клиент-сервер». Разграничение потребностей является принципом, лежащим в основе данного ограничения.
* Отсутствие состояния. Протокол взаимодействия между клиентов и сервером требует соблюдения следующего условия: в период между запросами клиента никакая информация о состоянии клиента на сервере не хранится. Все запросы от клиента должны быть составлены так, чтобы сервер получил всю необходимую информацию для выполнения запроса. Состояние сессии сохраняется на стороне клиента.
* Кэширование. Клиенты могут выполнять кэширование ответов сервера. Ответы сервера, в свою очередь, должны иметь явное или неявное обозначение как кэшируемые или некэшируемые с целью предотвращения получения клиентами устаревших или неверных данных в ответ на последующие запросы.
* Единообразие интерфейса. Наличие унифицированного интерфейса является фундаментальным требованием дизайна REST-сервисов. Унифицированные интерфейсы позволяют каждому из сервисов развиваться независимо. К унифицированным интерфейсам предъявляются следующие ограничительные условия: идентификация ресурсов (все ресурсы идентифицируются в запросах), манипуляция ресурсами через представление, «самоописываемые» сообщения (каждое сообщение содержит достаточно информации, чтобы понять, каким образом его обрабатывать).
* Слои. Клиент обычно не способен (и не должен) точно определить, взаимодействует он напрямую с сервером или же с промежуточным узлом. Применение промежуточных сервером способно повысить масштабируемость за счёт балансировки нагрузки и распределённого кэширования.
* Код по требованию (необязательное ограничение). [2]

В распределённых системах используется понятие «Удалённый вызов процедур» (RPC). Это класс технологий, позволяющих программам вызывать функции или процедуры в другом адресном пространстве – на удалённых узлах, либо в независимой сторонней системе на том же узле. Обычно реализация RPC-технологии включает два компонента: сетевой протокол для обмена в режиме клиент-сервер и язык сериализации объектов. Архитектурный стиль REST можно считать альтернативной заменой RPC. Следует заметить, что архитектурный стиль REST не включает в себя компоненты «сетевой протокол» и «сериализация объектов», а использует их.

2 Cмежные с архитектурным стилем REST технологии

Так как архитектурный стиль REST использует компоненты сетевого протокола и сериализации объектов, необходимо определиться с этими понятиями, а также рассмотреть альтернативы, которые применимы в рамках REST.

Сетевой протокол – набор правил и действий, позволяющий установить соединение и обмен данными между двумя и более включенными в сеть устройствами. Примеры сетевых протоколов:

* HTTP (Hyper Text Transfer Protocol) – протокол передачи гипертекста. Позволяет передавать различные данные в текстовом виде. [5]
* FTP (File Transfer Protocol) – протокол передачи файлов со специального файлового сервера на компьютер пользователя.
* POP3 (Post Office Protocol) – стандартный протокол почтового соединения.
* SMTP (Simple Mail Transfer Protocol) – протокол, который задаёт набор правил для передачи почты.
* TELNET – протокол удалённого доступа, который даёт возможности абоненту работать на любой ЭВМ, находящейся с ним одной сети, как на своей собственной.

Для передачи различных типов данных по сети необходимо использовать процесс сериализации объектов – это процесс перевода структуры данных в последовательность байтов. Обратной к операции сериализации является операция десериализации – создание структуры данных из битовой последовательности.

Протокол HTTP накладывает ограничение на структуры данных – это должны быть данные в текстовом формате, поэтому необходимо рассмотреть, какие именно существуют текстовые форматы:

* Простой текст (plain text) – обычный набор символов, который воспринимается как текст. Не позволяет структурирование данных
* Двоичные данные (binary data) – последовательность символов, в которой закодированы двоичные данные. Как правило, используется для передачи файлов в виде текста
* XML (extensible markup language) – расширяемый язык разметки. Позволяет структурирование. С физической точки зрения XML-документа состоит из сущностей, из которых каждая может ссылаться на другую сущность. Единственный корневой элемент – документная сущность. Содержание сущностей – символы. С логической точки зрения документ состоит из комментарием, объявлений, элементов, ссылок на сущности и инструкций обработки. Всё это в документе структуризуется разметкой.
* JSON – текстовый формат обмена данными, основанный на JavaScript.
* Protocol Buffers – протокол сериализации (передачи) структурированных данных, предложенный Google, как эффективная бинарная альтернатива текстовому формату XML. Разработчики сообщают, что Protocol Buffers проще, компактнее и быстрее, чем XML, поскольку осуществляется передача бинарных данных, оптимизированных под минимальный размер сообщения.
* YAML – «дружественный» формат сериализации данных, концептуально близкий к языкам разметки, но ориентированный на удобство ввода-вывода типичных структур данных многих языков программирования.

3 Описание веб-сервиса и выбор технологий

Сервис приёма заявок купли/продажи предоставляет возможность различным пользователям системы создавать объявления о продаже своего имущества, а также просматривать объявления на продажу других пользователей системы. Пользователи могут находиться в разных городах, поэтому единственной объединяющей их компьютерной сетью является сеть Интернет. Все пользователи должны иметь доступ к данным сервиса, таким как:

* информация об объявлениях на продажу;
* информация о заявителях объявлений;
* прочая информация, необходимая для совершения сделки купли/продажи (цена, адрес и другое).

Из определения архитектурного стиля REST становится очевидна наилучшая область применения данной архитектуры – системы, основанные на данных. Это означает, что если необходимо обеспечить централизованное хранение данных и распределённый доступ к ним, то выбор архитектурного стиля REST можно считать наилучшим решением.

Говоря о преимуществах самого архитектурного стиля REST, можно выделить следующие:

* надёжность, за счёт отсутствия необходимости сохранять информацию о состоянии клиента, которая может быть утеряна;
* производительность, за счёт использования кэширования;
* масштабируемость;
* простота системы взаимодействия;
* простота и унификация интерфейсов;
* портативность компонентов;
* лёгкость внесение изменений и поддержки системы;
* способность эволюционировать, приспосабливаясь к новым требованиям. [2]

Альтернативой архитектурному стилю REST является протокол SOAP (Simple Object Access Protocol) – расширение протокола XML-RPC, который, в свою очередь, является одним из стандартов вызова удалённых процедур, использующий XML для кодирования сообщения и HTTP в качестве транспортного механизма. Согласно исследованию производительности приложений, построенных по архитектурному стилю REST с приложениями, использующими SOAP, первые работают эффективнее.

|  |  |  |  |
| --- | --- | --- | --- |
| Тип запроса | Количество запросов | Добавление запроса, мс | Получение запроса, мс |
| REST | 50 | 1084 ± 97,1 | 1380 ± 111,8 |
| 100 | 1294 ± 104,7 | 1881 ± 167,6 |
| 250 | 1852 ± 126,6 | 2974 ± 255,8 |
| 500 | 2301 ± 197,4 | 4598 ± 343,2 |
| 1000 | 3045 ± 292,9 | 7337 ± 696,4 |
| SOAP | 50 | 1174 ± 106,8 | 1512 ± 132,7 |
| 100 | 1378 ± 117,2 | 1974 ± 183,4 |
| 250 | 1997 ± 145,1 | 3442 ± 290,7 |
| 500 | 2568 ± 212,5 | 5174 ± 378,4 |
| 1000 | 3289 ± 313,7 | 8217 ± 710,5 |

Таблица 1. Результаты множественного запроса [5]

Из обзора сетевых протоколов и их возможностей можно сделать вывод, что наиболее общим по назначению является протокол HTTP, так как можно передавать произвольные данные. Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом. Основным объектов манипуляции в HTTP является ресурс. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представление одного и того же ресурса по различным параметрам: формату, кодировке, языку. Благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениться двоичными данными, хотя данный протокол является текстовым.

При ближайшем рассмотрении протокола HTTP становится понятно, что именно его лучше всего выбрать в качестве сетевого протокола при использовании архитектурного стиля REST, так как и там, и там:

* основой является технология «клиент-сервер»;
* имеется возможность работы над произвольными данными;
* имеется возможность организовывать произвольные запросы и ответы на них.

Таким образом, в качестве вызова удалённой процедуры будет выступать обычный HTTP-запрос, в котором необходимые данные будут передаваться в качестве параметров запроса. Интересно заметить, что термин «REST» был введён одним из создателей протокола «HTTP» Роем Филдингом в своей диссертации «Архитектурные стили и дизайн сетевых программных архитектур», в которой он подвёл теоретическую основу под способ взаимодействия клиентов и сервером, абстрагировав его и назвав «передачей представительного состояния». 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.

Произведём сравнительный анализ форматов передачи данных.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Формат передачи данных | Стандартизация | Двоичный | Человекочитаемый | Стандартные API |
| JSON | Есть | Нет | Да | Частично |
| XML | Есть | Частично | Да | Да (DOM, SAX, XQuery, XPath) |
| Google protobuf | Нет | Да | Частично | Реализации на многих языках программирования |
| YAML | Нет | Нет | Да | Нет |

Таблица 2. Сравнение форматов передачи данных. [6]

При разработке приложений важна человекочитаемость сообщений, поэтому google protobuf подходит лишь тогда, когда акцент делается на производительность передачи больших сообщений. Впрочем, нет убедительных доказательств того, что google protobuf действительно эффективные передаёт данные в сравнении с другими форматами передачи данных. Стандартизация формата определяет одинаковое поведение инструментов, работающих с данным форматом, поэтому формат данных YAML лучше не использовать в качестве промышленного решения. Необходимость передачи данных различных структур ограничивает выбор текстовым форматов теми, которые поддерживают структурирование – это форматы XML и JSON. Целесообразно сделать выбор в пользу популярного в настоящее время формата JSON, так как у него есть ряд преимуществ перед форматом XML: простота, читабельность, компактность. Рассмотрим, как выглядит одно и то же сообщение в формате JSON и в формате XML.

JSON:

{

"firstName": "Иван",

"lastName": "Иванов",

"address": {

"streetAddress": "Московское ш., 101, кв.101",

"city": "Ленинград",

"postalCode": 101101

},

"phoneNumbers": [

"812 123-1234",

"916 123-4567"

]

}

XML:

<person>

<firstName>Иван</firstName>

<lastName>Иванов</lastName>

<address>

<streetAddress>Московское ш., 101, кв.101</streetAddress>

<city>Ленинград</city>

<postalCode>101101</postalCode>

</address>

<phoneNumbers>

<phoneNumber>812 123-1234</phoneNumber>

<phoneNumber>916 123-4567</phoneNumber>

</phoneNumbers>

</person>

Из этого простого примера видно, как формат JSON выгодно отличается от XML по количеству используемых символов в сообщении.

ЗАКЛЮЧЕНИЕ

В результате выполнения исследовательской работы был произведён обзор архитектурного стиля REST, выявление наилучших условий для его применения. Был произведён обзор смежных технологий и понятий, а также выбор альтернативных способов и технологий для возможной последующей реализации сервиса подачи заявок на куплю/продажу.

Результаты работы позволяют приступить к проектированию компонентов системы в соответствии с архитектурным стилем REST, их состав и обязанности, в соответствии с выбранными технологиями и методами вызова процедур и передачи данных. На основе проектирования можно будет приступить к выбору средств реализации сервиса подачи заявок на куплю/продажу.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Cтатья по архитектурному стилю REST. - URL: <https://ru.wikipedia.org/wiki/REST> (дата обращения: 18.04.2021)
2. Chapter 5 of Roy Fielding’s dissertation «Representational State Transfer (REST)» - URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm> (дата обращения: 18.04.2021)
3. Статья о протоколе HTTP - URL: <https://ru.wikipedia.org/wiki/HTTP> (дата обращения: 18.04.2021)
4. Статья о доле рынка приложений - URL: <https://www.safebettingsites.com/2020/10/13/google-play-hit-28-3bn-downloads-in-q3-2020-3x-more-than-app-store/> (дата обращения: 26.04.2021)
5. Статья о производительности протоколов доступа к облачным ресурсам - URL: <https://cyberleninka.ru/article/n/sravnenie-proizvoditelnosti-protokolov-dostupa-k-oblachnym-resursam/viewer> (дата обращения: 28.04.2021)
6. Статья о сравнении форматов передачи данных - URL: <https://360wiki.ru/wiki/Comparison_of_data-serialization_formats> (дата обращения: 28.04.2021)
7. Макконнелл С.М15 Совершенный код. Мастер#класс / Пер. с англ. — М. : Издательство «Русская редакция», 2010. — 896 стр. : ил.