МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**НА ТЕМУ «АРХИТЕКТУРА И МЕТОДЫ РАЗРАБОТКИ WEB-СЕРВИСОВ»**

Выполнил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.В. Краснопер

(подпись, дата)

Факультет компьютерных технологий и прикладной математики,

3 курс, 36 группа  
Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Научный руководитель к.т.н., доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата)

Нормоконтролер к.т.н., доцент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.Е. Полупанова

(подпись, дата)

Краснодар, 2018

Оглавление

[Введение 3](#_Toc532497470)

[1 ТЕХНОЛОГИЯ WEB-СЕРВИСОВ 4](#_Toc532497471)

[1.1 Отличия SOAP от XML-RPC и REST 5](#_Toc532497472)

[1.2 Когда следует использовать SOAP 7](#_Toc532497473)

[1.3 Архитектура SOAP Web-сервисов 7](#_Toc532497474)

[1.4 Преимущества и недостатки Web-сервисов 12](#_Toc532497475)

[2 ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ SOAP WEB-СЕРВИСА 13](#_Toc532497476)

[2.1 Постановка задачи 13](#_Toc532497477)

[2.2 Реализация поставленной задачи 13](#_Toc532497478)

[Заключение 19](#_Toc532497479)

[Список использованных источников: 20](#_Toc532497480)

[Приложение А index.js 21](#_Toc532497481)

[Приложение Б client.js 23](#_Toc532497482)

[Приложение В Descr.wsdl 25](#_Toc532497483)

[Приложение Г package.json 29](#_Toc532497484)

Введение

Ранее в сети Интернет взаимодействия ограничивались лишь передачей текстовой и графической информации. Сегодня WWW предоставляет огромный набор возможностей, например, выбор необходимого товара среди большого набора магазинов с возможностью сравнения характеристик товара и его цены, а также можно заказать доставку прямо до дома или же просто прочитать свежие новости.

Во всем мире компании стараются использовать ресурсы сети Интернет по максимуму, чтобы достичь эффективного выполнения их бизнес-процессов и занять выгодное место на рынке. Однако WWW все еще не способна обеспечить необходимый уровень обслуживания клиентов и передача большого объема данных становится затруднительной. Для решения таких проблем необходимо составить такой способ взаимодействия приложений, который обеспечит простоту обмена данными между ними, а также выполняющий команды, которые раньше нужно было вводить вручную, автоматически. Здесь приходят на помощь Web-сервисы - обеспечивая взаимодействие приложений, они значительно увеличивают возможности Интернета, позволяя приложениям действовать непосредственно, будто они являются частью одной огромной информационной системы.

Безусловно, практическая польза Web-сервисов велика, так как они обеспечивают связь между приложениями, выполняемыми в любых местах нашей планеты, и позволяют обмениваться большими объемами данных между ними намного быстрее и продуктивнее, чем раньше.

1. ТЕХНОЛОГИЯ WEB-СЕРВИСОВ

В наше время World Wide Web терпит сильные изменения. Еще пару лет назад «всемирная сеть» представляла собой набор серверов со статическими документами и ссылками друг на друга, в то время как современный Web очень сложно представить без интерактивных Web-приложений, принимающих и обрабатывающих разнообразные запросы.

Однако развитие WWW не стоит на месте, что привело к созданию Web-сервисов, основанных на двух технологиях – XML и SOAP и имеющих интерфейсы в формате WSDL. Подразумевается, что обмен данными между серверами приложений таких сервисов происходит в формате XML по протоколу SOAP, а такой формат интерфейса обеспечивает автоматическую генерацию кода на клиентской стороне, нужного для связи с сервисом.

Web-сервисы (или Web-службы) призваны для согласования работы больших приложений, но кроме этого, они также могут использоваться в роли повторно-используемых компонент, предоставляющих разные сервисы, например, прогноз погоды или перевод текста с одного языка на другой.

Помимо вышеназванной группы сервисов, основанной на XML и SOAP, существуют еще две немало важные группы, приведенные ниже.

- XML-RPC Web-сервисы – такие сервисы ориентированы на сообщения и обеспечивают низкоуровневую обработку XML-сообщений. Они могут передавать и получать сообщения как в чистом XML-формате, так и в SOAP.

- RESTful Web-сервисы – такие сервисы представляют удаленные ресурсы, доступ к которым обеспечивается HTTP-запросами. Технология REST тоже может использовать WSDL описание интерфейсов и SOAP-протокол, но также может обходиться и без них.

На данный момент также существует несколько технологий для создания распределенных систем, являющиеся альтернативой применения технологии Web-сервисов: CORBA, JAVA RMI, DCOM.

Наиболее известные примеры Web-сервисов: интернет поисковики - Google, Яндекс, Yahoo; сервисы облачного хранения и передачи данных – Dropbox, Google.Drive, Box; почтовые сервисы – gmail.com, mail.com и так далее.

* 1. Отличия SOAP от XML-RPC и REST

Приведем сравнение SOAP и XML-RPC.

- SOAP более многословный, но при этом более мощный и эффективный, знание SOAP приветствуется работодателями намного больше, нежели XML-RPC;

- SOAP поддерживает передачу на уровне документа, в то время как XML-RPC больше относится к передаче значений;

- XML-RPC в сравнении с SOAP более универсален, так как он позволяет развертывать достаточно просто и быстро качественные веб-сервисы даже при использовании текстового редактора терминала, а в случае SOAP нам необходима действительно хорошая IDE, чтобы это осуществить;

- У XML-RPC есть некоторые проблемы с наборами символов, не являющимися ASCII.

Считается, что XML-RPC уже достаточно старая технология. На данный момент лучшими Web-сервисами являются SOAP и REST. SOAP был разработан до REST и основная его идея заключалась в том, чтобы программы, созданные на разных платформах и языках программирования, могли легко обмениваться данными между собой.

REST же был разработан для работы с такими компонентами, как медиа-компоненты, файлы и даже объекты на определенном аппаратном устройстве.

Ниже приведены основные различия между SOAP и REST.

- SOAP – протокол, REST – архитектурный стиль.

- SOAP был разработан со спецификацией, он включает в себя файл WSDL, который содержит в себе информацию о том, что делает веб-сервис. В случае REST веб-сервис является RESTful-сервисом только тогда, когда он соответствует ограничениям: наличие клиентского сервера, независимость относительно платформ реализации, наличие кэширование, многоуровневой системы и единого интерфейса.

- SOAP не может использовать REST, в то время как REST может использовать SOAP в качестве основного протокола для веб-сервисов, потому что REST все же является архитектурным стилем.

- SOAP поддерживает SSL и WS-security, в то время как REST – только SSL.

- REST работает только по HTTP(S), а SOAP может работать с различными протоколами.

- В сравнении с REST, SOAP требуется большая пропускная способность, так как в SOAP-сообщениях содержится много информации о содержании и прочем, а REST сообщения в основном состоят из сообщений JSON.

- SOAP может работать только с форматом XML. REST допускает различные форматы данных, такие как обычный текст, HTML, XML, JSON, но наиболее предпочтительным является JSON.

Исходя из сравнения, можно сказать, что RESTful Web-сервисы – это простота, скорость, расширяемость и поддержка многих форматов, из-за чего они могут оказаться более производительными, а также в ней отсутствуют затраты на разбор сложных XML команд. REST рекомендуется использовать в простых архитектурах, где не нужны более сложные отношения, чем создать, прочитать, изменить и удалить. Однако SOAP Web-сервисы предоставляются более надежными и безопасными, почему и рекомендуются для использования в сложных структурах.

* 1. Когда следует использовать SOAP

- Если существует такое требование, что клиенту требуется гарантированный уровень надежности и безопасности, тогда стандарт SOAP предоставляет множество дополнительных функций, особенно когда речь идет о безопасности.

- Если клиент и сервер имеют соглашение о формате обмена, тогда SOAP дает жесткие спецификации для такого типа взаимодействия. Например, сайт онлайн-покупок, на котором пользователи добавляют товары в корзину до совершения платежа. Предположим, у нас есть веб-сервис, который выполняет окончательный расчет. Может быть такое соглашение, что веб-сервис будет принимать только наименование корзины, цены за единицу и количество. Если такое соглашение есть, то лучшим выбором будет SOAP.

- Если у приложения есть требование о том, что состояние должно поддерживаться от одного запроса к другому, тогда стандарт SOAP предоставляет структуру WS\* для поддержки таких требований.

* 1. Архитектура SOAP Web-сервисов

SOAP Web-сервисы основываются на четырех технологиях: XML, SOAP, WSDL и UDDI. Эти технологии обеспечивают работоспособность сервисов, и связаны между собой так, как показано на рисунке 1:

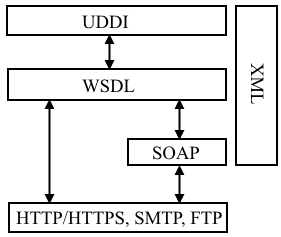


Рисунок 1 – Структура SOAP Web-сервисов

В нижней части рисунка 1 указан так называемый протокольный уровень, который содержит стандартные протоколы вызова Web-сервисов по сети. Изначально в нем используется HTTP/1.1 – текстовый протокол вида «запрос-ответ», который подразумевает следующую последовательность действий: клиент устанавливает соединение с сервером и отсылает запрос в специальном для протокола формате, затем сервер принимает и обрабатывает данный запрос, после чего отвечает на него. Однако также можно использовать SMTP и FTP.

XML выполняет важнейшую роль в Web-службах. Помимо того, что он является основополагающей SOAP и WSDL, XML задает кроссплатформенный и модифицируемый формат данных для обмена информацией между клиентом и сервисом. В отличие от HTML, XML дает возможность отображать один и тот же информационный документ различными способами. В общем смысле, XML – язык разметки документов, обеспечивающий текстовый формат хранения данных.

Протокол сервис-ориентированной архитектуры, основанный на XML, предназначен для взаимодействия клиентов с сервисами посредством обмена XML-сообщениями. Чаще всего SOAP используется поверх HTTP, однако протоколы FTP, SMTP и другие также могут его использовать.

Гибкость структуры SOAP предоставляет возможность адаптировать SOAP-сообщения разнообразным требованиям распределенной среды (например: безопасность, надежность), что делает SOAP независящим от платформы протоколом.

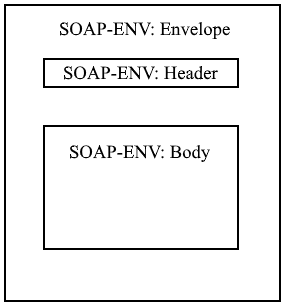


Рисунок 2 – «Структура SOAP-сообщения»

Структура сообщения SOAP (см. рисунок 2) состоит из четырех элементов <Envelope> (конверт), <Header> (заголовок), <Body> (тело) и <Fault> (неисправность).

1) Корневой элемент, играющий роль идентификатора SOAP-сообщения, указывается с использованием префикса пространства имен ENV и элемента Envelope. В общем можно сказать, что конверт SOAP решает проблему, когда мы знаем, когда именно мы получили сообщение и готовы его обработать. То есть конверт SOAP - это в основном механизм упаковки сообщения. Конверт является обязательной частью сообщения SOAP и каждый такой конверт должен содержать ровно один элемент Body.

2) Необязательный элемент, содержащий информацию, которая влияет на обработку сообщения после получения приложением SOAP-сообщения. Элементы заголовка могут встречаться в сообщении несколько раз <Header> предназначены для указания дополнительных функций и возможностей приложения, например, он может использоваться для указания цифровой подписи служб, защищенных паролем. Заголовок также может содержать элементы, несущие данные о безопасности, маршрутизации и др., называемые блоками заголовка.

Блоки заголовка в свою очередь могут иметь такие атрибуты, как:

- encodingStyle – хранит в себе информацию о способе преобразования данных в формат SOAP-сообщения;

- role – задает SOAP-роли для SOAP-обработчика;

- mustUnderstand – в зависимости от значения true или false устанавливается обязанность проверки данного блока заголовка обработчиком;

- relay – при значении true указывает, что данный блок заголовка остается в SOAP-сообщении, если он не был до этого обработан промежуточным SOAP-обработчиком.

3) Обязательный элемент, который транспортирует основные данные конечному SOAP-получателю. Тело SOAP должно содержаться в конверте и должно следовать всем заголовкам, которые могут быть определены для сообщения.

4) Если во время обработки возникает ошибка, то ответом на SOAP-запрос является элемент ошибки в теле сообщения. Отказ возвращает конкретную информацию об ошибке, включая предопределенный код, описание и адрес процессора SOAP, который сгенерировал ошибку. Сообщение может содержать только один блок отказа. Ошибка SOAP является необязательной частью сообщения и связана с диапазоном кодов состояния от 500 до 599. Ошибка SOAP имеет следующие элементы:

- <faultCode>: В данном элементе выводится класс ошибки: SOAP-ENV: VersionMismatch, SOAP-ENV: MustUnderstand, SOAP-ENV: Client, SOAP-ENV: Server.

- <faultString>: Сообщение, объясняющее ошибку.

- <faultActor>: Текстовая строка, указывающая на то, кто вызвал ошибку.

Пригодится в случае, когда SOAP-сообщение проходит через несколько узлов. Все промежуточные узлы также должны содержать в себе этот элемент.

- <detail>: Элемент, используемый для передачи сообщений об ошибках приложения.

Из структуры SOAP-сообщения можно сделать вывод, что оболочка сообщения независима относительно протокола, по которому передаются сообщения.

WSDL обеспечивает доступ к Web-сервису и хранит в себе полную информацию, необходимую для допуска к данному сервису. Видит Web-сервис как список портов, которые являются набором абстрактных операций и сообщений. Такие операции дают возможность использовать их с SOAP, HTTP, MIME или GET/POST.

WSDL описывает среду как конечный набор точек, обменивающихся сообщениями друг с другом, и также не зависит от платформы и может применяться в Web-сервисах различной структуры.

WSDL описывает сервис двумя уровнями:

1. абстрактный – описание в терминах получаемых и отсылаемых сообщений;
2. детальный – уточняются детали транспортного протокола, формата сообщений, функциональность и адрес сервиса.

Подробнее WSDL рассмотрим в практической части.

UDDI – это реестр на основе XML, позволяющий находить Web-сервисы по WSDL-описанию и не зависящий от платформы. Его цель – упростить онлайн-транзакции, позволяя компаниям (и не только) находить друг друга в Интернете и обеспечивать совместимости своих систем для электронной коммерции.

По сути, UDDI предоставляет пользователям детальную информацию о деталях реализации Web-сервисов, а также их функциональным возможностям. Для управления такими реестрами создается определенный сервер, предоставляющий специальное приложение, позволяющее управлять информацией и сервисах в различного рода базах данных.

Такой реестр можно логически разделить на три категории:

- информация о поставщике Web-сервиса;

- классификация Web-сервиса;

- детали реализации Web-сервиса.

* 1. Преимущества и недостатки Web-сервисов

Преимущества:

1. web-сервисы могут работать на любой ОС, достаточно иметь установленный браузер и наличие подключения к Интернету;
2. благодаря резервному копированию, базы данных Web-сервиса надежно защищены от потери какой-либо информации;
3. в сравнении с CORBA или JAVA RMI, в Web-сервисах HTTP-взаимодействие программных систем реализуется через межсетевой экран, который фильтрует проходящий через него сетевой трафик по определенным правилам (также межсетевой экран имеет названия - Брандмауэр и Файрвол).

Недостатки:

1. из-за использования XML-сообщений понижается производительность и увеличивается размер сетевого трафика;
2. для обеспечения безопасности, качественные Web-сервисы вынуждены прибегать к дополнительному кодированию и, возможно, потребовать пользователя пройти аутентификацию.
3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ SOAP WEB-СЕРВИСА
   1. Постановка задачи

Реализация сервиса на данный момент не ставит задачу создать крупный проект, обладающий широким функционалом. Цель практического исследования состоит в понимании создания, описания и работы SOAP Web-сервиса. Для выполнения поставленной цели создадим достаточной простой сервис, который будет получать SOAP-запросы, содержащие информацию для выполнения заданных вычислительных функций, а именно сложение, вычитание, умножение и деление двух чисел. Отвечать программа будет также SOAP-сообщением.

В качестве языка программирования выберем JavaScript, а в качестве среды выполнения кода JavaScript выберем Node.js – серверная платформа, основанная на V8. Версия SOAP будет рассматриваться SOAP 1.1.

* 1. Реализация поставленной задачи

Наша программа состоит из пяти частей:

- Index.js

- Client.js

- Descr.wsdl

- Package.json

- Package-lock.json

Рассмотрим создание, свойства и роль каждой части чуть подробнее.

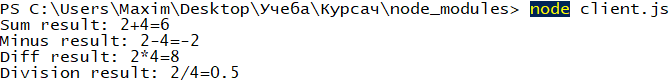
1) Index.js поднимает сервер и по описанию WSDl-файла Descr.wsdl создает наш SOAP-сервис. Для реализации данных задач необходимо подключить два модуля http и soap, где первый отвечает за сервер, который мы будем поднимать, а второй за организацию SOAP-сервиса.

Далее определим наш сервис var service, назовем его Some\_service и зададим порт Some\_Port, который будет содержать четыре вычислительных функции: sumNums, minusNums, differenceNums и divisionNums. Установим синхронизацию с WSDL-файлом readFileSync(‘.Descr.wsdl’, ‘utf8’) в переменную «xml», укажем порт 3030 и, наконец, создадим сервер при помощи server = http.createServer, дополнительно описав ситуацию 404: Not Found.

В конце задаем наш SOAP Web-сервис на данный сервер выражением soap.listen(server, ‘/’, service,xml). Теперь, если задание сервера, работоспособность заданных функций и описание Web-сервиса корректны, мы можем поднять наш сервис в среде выполнения командой node index.js и проверить его работоспособность, перейдя в браузере по ссылке http://localhost:3030/.

2) Client.js отвечает за вызов функций, описанных в нашем сервисе. Здесь также нужно подключить модуль «soap» для создания клиента «soapClient» функцией soap.createClient, где опишем вывод ошибочного сообщения при некорректности запроса или целостности и работоспособности проекта в целом, а также вызов наших функций в ином случае. Для каждой функции также отдельно зададим вывод ошибки в ошибочной ситуации.

Теперь, при вызове данного файла командой node client.js (в это время сервер index.js должен быть запущен), мы увидим такой ответ:

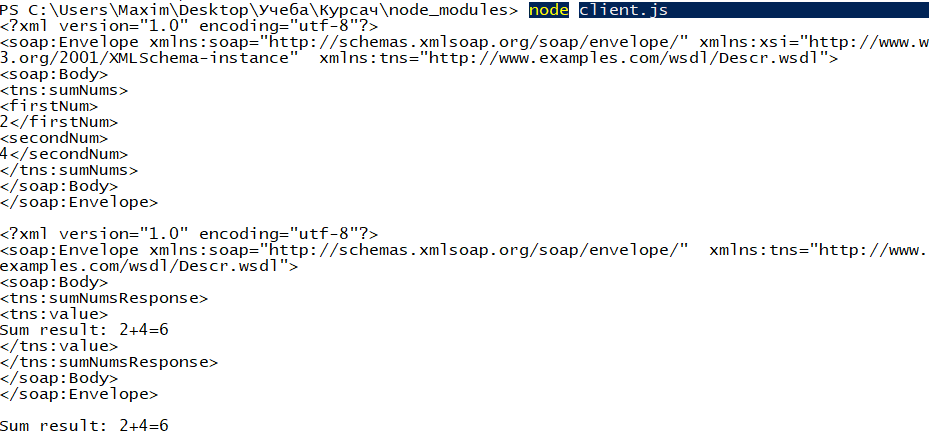


Для того, чтобы убедиться в работе SOAP, добавим вывод тела SOAP-запросов и SOAP-ответов наших функций, добавив две такие строки:

client.on('request',(body)=>{console.log(body.replace(/>/g, ">\n"))}) – вывод тела запроса; replace помогает преобразовать вывод в более пригодный для показа вид.

client.on('response',(body)=>{console.log(body.replace(/>/g, ">\n"))}) – вывод тела ответа.

Для наглядности вызовем node client.js только для сложения, так как для всех четырех функций результат будет объемным:



Сначала идет запрос, где мы видим две наши переменные firstNum и secondNum, равные 2 и 4 соответственно, вызванные вместе с функцией sumNums (мы можем это увидеть, посмотрев на <ths: sumNums>).

Далее мы вывели SOAP-ответ, где «value» несет результат выполнения функции сложения двух чисел, который можно увидеть под ответом. «Value» является частью сообщения sumNumsResponse, которое является шаблоном ответа на запрос функции сложения двух чисел и описано в WSDL-файле. Об описании сообщений рассмотрим далее.

3) Descr.wsdl - WSDL-файл, полностью описывающий наш Web-сервис, его работу и доступ к нему извне. По сути, он является основополагающей SOAP Web-сервиса и работать с ним нужно аккуратно. Основное предназначение описания было рассмотрено выше, теперь подробнее рассмотрим элементы WSDL, их назначение и опишем наш сервис.

Документ содержит в себе такие элементы, как: definitions, types, message, operation, portType, binding, port и service. Структура WSDL-файла указана на рисунке 3. Теперь рассмотрим каждый элемент подробнее.

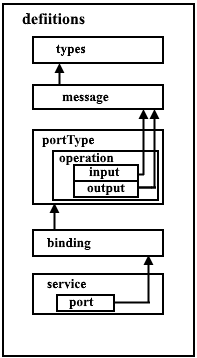


Рисунок 3 – Структура WSDL документа

- definitions.

Содержит используемые для сообщений определения XML. Поле name является необязательным, xmlns: tns обозначает пространство имен, xmlns: soap определяет SOAP протокол, xmlns:xsd является стандартным определением пространства имен, которое используется для указание специфичной информации для SOAP. TargetNamespace является «логическим именем» документа WSDL, так как WSDL могут использовать другие WSDL, а targetNamespace поможет не конфликтовать им между собой.

- types.

Предоставляет информацию о любых сложных типах данных, используемых в WSDL. Если сложных типов в описании нет, то WSDL документ не должен иметь раздел типов, как в нашем сервисе.

- message.

Абстрактное определение передаваемых данных. В нашей программе мы описали восемь сообщений по два на каждую функцию, где одно сообщение создано для запроса, другое для ответа.

Рассмотрим два таких сообщения для функции сложения двух чисел.

<message name = "sumNumsRequest">

<part name = "firstNum" type = "xsd:string"/>

<part name = "secondNum" type = "xsd:string"/>

</message>

Данное сообщение отвечает за построение SOAP-запроса, его имя sumNumsRequest. Тип наших переменных указан string, однако при вычислении мы проводим конвертацию string в number.

<message name = "sumNumsResponse">

<part name = "value" type = "xsd:string"/>

</message>

Данное сообщение отвечает за построение SOAP-ответа, его имя sumNumsResponse мы можем увидеть в теле SOAP-запроса. Значением сообщения является переменная value типа string, которая содержит в себе ответ запрошенной функции (в нашем случае – сложения).

- operation.

Абстрактное описание действия, поддерживаемого сервисом. Содержит в себе два сообщения, одно из которых отвечает за вход данных input, другое за выход output. Возвращаясь к нашему примеру, input – sumNumsRequest, output – sumNumsResponse. Раздел operation содержится в portType и их может быть несколько.

- portType.

Набор операций, поддерживаемых одной или несколькими точками (портами). В нашем Web-сервисе portType содержит в себе четыре элемента operation. Для наглядности ниже приведен пример portType для одной только функции сложения, где имя порта Some\_PortType.

<portType name = "Some\_PortType">

<operation name = "sumNums">

<input message = "tns:sumNumsRequest"/>

<output message = "tns:sumNumsResponse"/>

</operation>

</portType>

- binding.

Данный элемент описывает вызов операций путем указания конкретного протокола (в нашем случае SOAP) и спецификаций формата данных сообщений. Необходимо также указать название порта, в котором находится описываемая функция и само имя элемента binding для дальнейшего указания в элементе port.

- port.

Указывается одна конечная точка в качестве адреса для привязки, определяя, таким образом, единственную конечную точку связи. Элемент port содержится в элементе service. Помимо этого, в port указывается адрес, где будет расположен сервис.

<port binding = "tns:Some\_Binding" name = "Some\_Port">

<soap:address location = "http://localhost:3030/" />

</port>

- service.

Определяет адреса порта привязки. Сервис представляет собой набор сетевых конечных точек или портов.

4) Package.json позволяет упростить установка зависимостей между пакетами данных вручную, так как это очень трудоемкое и некомфортное занятие. Здесь у нас указано название проекта, его версия, имя автор и указан файл с сервером index.js.

5) Файл package-lock.json создается автоматически при установке Node.js пакетов. Его цель состоит в отслеживании точных версий установленных пакетов, что позволяет гарантированно сделать продукт рабочим и корректным.

# Заключение

В ходе проведения курсовой работы были рассмотрены различные способы реализации Web-сервисов, принципы их работы, а также их преимущества и недостатки.

В рамках проделанной работы были подробно изучены SOAP Web-сервисы, их структура, возможности и предпочтительные ситуации для применения, составлена их сравнительная характеристика с XML-RPC и REST. Также отдельное внимание было уделено важности WSDL документа в описании SOAP-сервисов, мы также подробно рассмотрели каждый элемент его структуры, рассказаны правила и принципы реализации такого документа, чтобы SOAP Web-сервис работал корректно. В конечном результате был создан такой сервис, который вычисляет сумму, разность, произведение и деление двух чисел и для наглядности выводит тела SOAP-запросов и SOAP-ответов.

# Список использованных источников:

1 Машнин Т.С. Web-сервисы Java. – СПб.: БХВ-Петербург. - 2012.

2 SOAP vs REST (различия) [Электронный ресурс]. – URL: <http://qaru.site/questions/2543/soap-vs-rest-differences> (дата обращения 21 ноября 2018).

3 Web Services Tutorial [Электронный ресурс]. – URL: <https://www.guru99.com/web-service-architecture.html> (дата обращения 3 декабря 2018).

4 SOAP Tutorial [Электронный ресурс]. – URL: <https://www.tutorialspoint.com/soap/soap_message_structure.htm> (дата обращения 8 декабря 2018).

5 Web Services Description Language (WSDL) 1.1 [Электронный ресурс]. – URL: <https://www.w3.org/TR/2001/NOTE-wsdl-20010315> (дата обращения 11 декабря 2018).

6 WSDL Tutorial [Электронный ресурс]. – URL: <https://www.tutorialspoint.com/wsdl/wsdl_elements.htm> (дата обращения 11 декабря 2018).

7 REST vs XML-RPC vs SOAP – pros and cons [Электронный ресурс]. – URL: <https://maxivak.com/rest-vs-xml-rpc-vs-soap/> (дата обращения 11 декабря 2018).

# Приложение А

# index.js

var soap = require('soap');

var http = require('http');

var result = "";

var service = {

Some\_Service: {

Some\_Port: {

sumNums: function (args) {

return {

value: +args.firstNum + +args.secondNum + "\n"

};

},

minusNums: function (args) {

return{

value: +args.secondNum - +args.firstNum + "\n"

};

},

differenceNums: function (args) {

return{

value: +args.secondNum \* +args.firstNum

};

},

divisionNums: function (args) {

return{

value: +args.secondNum / +args.firstNum

};

}

}

}

}

var xml = require('fs').readFileSync('./Descr.wsdl', 'utf8');

var server = http.createServer(function(request,response) {

response.end('404: Not Found: ' + request.url);

});

server.listen(3030);

console.log("Запущено");

soap.listen(server, '/', service, xml);

# Приложение Б

# client.js

var soap = require('soap');

var url = "http://localhost:3030/?wsdl";

var args = {

firstNum: 2,

secondNum: 4,

}

var soapClient;

soap.createClient(url, function (err, client) {

if (err){

console.error(err);

} else {

soapClient = client;

client.on('request',(body)=>{console.log(body.replace(/>/g, ">\n"))})

client.sumNums(args, function (err, response) {

if (err)

console.error(err.Fault);

else {

console.log(response.value);

}

});

client.minusNums(args, function (err, response) {

if (err)

console.error(err.Fault);

else {

console.log(response.value)

}

});

client.differenceNums(args, function (err, response) {

if (err)

console.error(err.Fault);

else {

console.log(response.value)

}

});

client.divisionNums(args, function (err, response) {

if (err)

console.error(err.Fault);

else {

console.log(response.value)

}

});

client.on('response',(body)=>{console.log(body.replace(/>/g, ">\n"))})

}

});

# Приложение В

# Descr.wsdl

<definitions name = "functionalService"

targetNamespace = "http://www.examples.com/wsdl/Descr.wsdl"

xmlns = "http://schemas.xmlsoap.org/wsdl/"

xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:tns = "http://www.examples.com/wsdl/Descr.wsdl"

xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

<message name = "sumNumsRequest">

<part name = "firstNum" type = "xsd:string"/>

<part name = "secondNum" type = "xsd:string"/>

</message>

<message name = "sumNumsResponse">

<part name = "value" type = "xsd:string"/>

</message>

<message name = "minusNumsRequest">

<part name = "firstNum" type="xsd:string"/>

<part name = "secondNum" type="xsd:string"/>

</message>

<message name = "minusNumsResponse">

<part name = "value" type="xsd:string"/>

</message>

<message name = "differenceNumsRequest">

<part name = "firstNum" type="xsd:string"/>

<part name = "secondNum" type="xsd:string"/>

</message>

<message name = "differenceNumsResponse">

<part name = "value" type="xsd:string"/>

</message>

<message name = "divisionNumsRequest">

<part name = "firstNum" type="xsd:string"/>

<part name = "secondNum" type="xsd:string"/>

</message>

<message name = "divisionNumsResponse">

<part name = "value" type="xsd:string"/>

</message>

<portType name = "Some\_PortType">

<operation name = "sumNums">

<input message = "tns:sumNumsRequest"/>

<output message = "tns:sumNumsResponse"/>

</operation>

<operation name = "minusNums">

<input message = "tns:minusNumsRequest"/>

<output message = "tns:minusNumsResponse"/>

</operation>

<operation name = "differenceNums">

<input message = "tns:differenceNumsRequest"/>

<output message = "tns:differenceNumsResponse"/>

</operation>

<operation name = "divisionNums">

<input message = "tns:divisionNumsRequest"/>

<output message = "tns:divisionNumsResponse"/>

</operation>

</portType>

<binding name = "Some\_Binding" type = "tns:Some\_PortType">

<soap:binding style = "rpc" transport = "http://schemas.xmlsoap.org/soap/http"/>

<operation name = "sumNums">

<soap:operation soapAction = "sumNums"/>

<input>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</input>

<output>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</output>

</operation>

<operation name = "minusNums">

<soap:operation soapAction = "minusNums"/>

<input>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</input>

<output>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</output>

</operation>

<operation name = "differenceNums">

<soap:operation soapAction = "differenceNums"/>

<input>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</input>

<output>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</output>

</operation>

<operation name = "divisionNums">

<soap:operation soapAction = "divisionNums"/>

<input>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</input>

<output>

<soap:body

encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"

namespace = "urn:examples:someservice"

use = "encoded"/>

</output>

</operation>

</binding>

<service name = "Some\_Service">

<documentation>WSDL File for HelloService</documentation>

<port binding = "tns:Some\_Binding" name = "Some\_Port">

<soap:address location = "http://localhost:3030/" />

</port>

</service>

</definitions>

# Приложение Г

# package.json

{

"name": "soapboi",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC",

"dependencies": {

"express": "^4.16.4",

"soap": "^0.25.0"

}

}