МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ВТ

КУРСОВАЯ РАБОТА

по дисциплине «Архитектура параллельных вычислительных систем»

Тема: Web-сервисы: SOAP-сервисы и REST-сервисы

Студентка гр. 0301 _____ Прохоров Б.В.

Студент гр. 0301 _____ Михайлов В.А.

Студентка гр. 0301 _____ Логунов О.Ю.

Преподаватель _____ Костичев С.В.

Санкт-Петербург
2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ 4
1.1 Теоретические аспекты веб-сервисов 4
1.2 SOAP-сервисы 5
1.3 REST-сервисы 7
1.4 Недостатки микросервисов 8
2 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ ДЛЯ ВЗАИМОДЕЙСТВИЯ С ВЕБ-СЕРВИСАМИ НА ОСНОВЕ SOAP И REST 12
2.1 Описание использованных инструментов 12
2.2 Описание реализованного приложения 12
ЗАКЛЮЧЕНИЕ 16
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 18
ПРИЛОЖЕНИЕ А. ЛИСТИНГ КОДА 20

ВВЕДЕНИЕ

Современные веб-технологии играют ключевую роль в интеграции информационных систем, позволяя приложениям обмениваться данными и взаимодействовать вне зависимости от платформ и языков программирования. Два основных подхода для реализации веб-сервисов – это SOAP (Simple Object Access Protocol) и REST (Representational State Transfer).

SOAP представляет собой протокол для обмена структурированными сообщениями, основанный на XML и поддерживающий строгие стандарты. REST, в свою очередь, является архитектурным стилем, который использует HTTP и предоставляет более гибкий способ взаимодействия, ориентированный на ресурсы.

Актуальность темы обусловлена широким использованием веб-сервисов в различных сферах, таких как электронная коммерция, банковские системы и мобильные приложения. Выбор подхода для разработки сервиса влияет на производительность, удобство интеграции и расширяемость системы.

Целью данной работы является анализ SOAP- и REST-сервисов с точки зрения их архитектуры, применимости и производительности.

Для достижения цели были поставлены следующие задачи:

1. Исследовать основные принципы работы SOAP- и REST-сервисов.

2. Провести сравнительный анализ подходов на основе практических примеров.

3. Оценить применимость каждого подхода в конкретных условиях и предложить рекомендации.

Работа состоит из теоретической части, в которой рассматриваются ключевые аспекты SOAP и REST, а также практической части, включающей реализацию простых примеров для демонстрации их особенностей.

1 ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Теоретические аспекты веб-сервисов

Веб-сервисы – это программные компоненты, которые обеспечивают взаимодействие между различными приложениями через интернет. Они играют роль мостов между разнородными системами, предоставляя стандартизированный интерфейс для обмена данными. Основное назначение веб-сервисов – интеграция приложений, работающих на разных языках программирования и платформах.

Основные характеристики веб-сервисов:

1. Веб-сервисы используют открытые протоколы, такие как HTTP, XML, JSON и SOAP, что обеспечивает совместимость между системами.
2. Сервисы могут быть разработаны на любой технологии и использоваться клиентами независимо от их среды исполнения.
3. Веб-сервисы легко адаптируются к изменяющимся требованиям бизнеса и могут обрабатывать большой объем запросов при правильной архитектуре.

Архитектурные принципы веб-сервисов:

1. Веб-сервисы построены по принципу взаимодействия клиента (отправляющего запрос) и сервера (обрабатывающего запрос и отправляющего ответ).
2. HTTP является наиболее популярным транспортным протоколом, позволяющим веб-сервисам обмениваться данными через интернет.
3. Веб-сервисы предоставляют интерфейсы (обычно в формате WSDL для SOAP или OpenAPI для REST), которые описывают доступные операции и способы их вызова.

Веб-сервисы делятся на две основные категории:

1. SOAP-сервисы, которые основываются на обмене XML-сообщениями, поддерживают сложные стандарты безопасности и надежности, что делает их подходящими для корпоративных систем.
2. REST-сервисы, которые используют легковесные протоколы, такие как JSON, и ориентированы на работу с ресурсами через стандартные HTTP-методы (GET, POST, PUT, DELETE).

Эти подходы имеют свои преимущества и ограничения, которые зависят от контекста их применения. SOAP лучше подходит для сложных бизнес-систем, где важны надежность и стандарты, а REST эффективнее в сценариях, требующих высокой производительности и простоты разработки.

1.2 SOAP-сервисы

SOAP (Simple Object Access Protocol) – это протокол для обмена структурированными сообщениями между приложениями в распределённых вычислительных системах. Он разработан консорциумом W3C и является стандартом для веб-сервисов, требующих надежности, безопасности и совместимости.

Основные особенности SOAP:

1. SOAP-сообщения представляют собой XML-документы, которые состоят из трёх основных частей:
   - Envelope – обёртка, которая определяет границы сообщения.
   - Header – необязательная секция, содержащая информацию о маршрутизации и настройках.
   - Body – основная часть сообщения, содержащая данные, которые передаются.
2. SOAP-сервисы описываются с помощью языка WSDL (Web Services Description Language), который содержит информацию о методах, параметрах и способах взаимодействия с сервисом.

3. SOAP поддерживает передачу данных через различные транспортные протоколы, включая HTTP, SMTP и TCP. Однако наиболее часто используется HTTP.
4. SOAP поддерживает дополнительные спецификации, такие как WS-Security для обеспечения безопасности, WS-ReliableMessaging для надежной доставки сообщений и WS-AtomicTransaction для поддержки транзакций.

SOAP имеет следующие преимущества:

- Поддержка WS-Security обеспечивает высокий уровень защиты данных.
- SOAP строго следует спецификациям, что делает его подходящим для корпоративных приложений.
- Возможность добавления кастомных элементов в заголовки позволяет адаптировать протокол под сложные сценарии.

SOAP имеет следующие ограничения:

- Использование XML увеличивает размер сообщений и усложняет их обработку.
- SOAP-сообщения более тяжелы по сравнению с REST, что приводит к большему времени обработки и нагрузке на сеть.
- SOAP-сервисы сложнее в настройке и интеграции, что может увеличить затраты на их использование.

SOAP-сервисы активно применяются в банковской сфере, страховании и крупных корпоративных системах, где важно соблюдать строгие стандарты безопасности и надежности. Например, платёжные системы и службы авторизации часто используют SOAP для обеспечения гарантированной доставки данных.

SOAP-сервисы остаются важным инструментом для создания веб-сервисов, несмотря на их сложность и конкуренцию со стороны REST. Они

особенно актуальны в ситуациях, где критичны надежность и соблюдение стандартизированных спецификаций.

1.3 REST-сервисы

REST (Representational State Transfer) – это архитектурный стиль для проектирования распределённых систем, основанный на работе с ресурсами через стандартные протоколы, такие как HTTP. REST-сервисы представляют собой лёгкую и гибкую альтернативу SOAP, позволяя разработчикам сосредоточиться на простоте и производительности.

Основные принципы REST:

1. В REST каждый ресурс (например, пользователь, документ или транзакция) имеет уникальный URI (Uniform Resource Identifier). Например, ресурс может быть представлен в виде строки «https://api.example.com/users/123».
2. REST опирается на методы HTTP для работы с ресурсами:
   - GET – получение ресурса.
   - POST – создание нового ресурса.
   - PUT – обновление существующего ресурса.
   - DELETE – удаление ресурса.
3. Каждый запрос к REST-сервису содержит всю необходимую информацию, что упрощает масштабирование системы.
4. REST-сервисы могут передавать данные в разных форматах, таких как JSON, XML, YAML или plain text. JSON является наиболее популярным благодаря своей легковесности и простоте.
5. REST поддерживает кэширование ответов сервера для повышения производительности.

REST имеет следующие преимущества:

- REST-сервисы легко проектировать, разрабатывать и поддерживать.

- Использование лёгких форматов данных, таких как JSON, уменьшает задержки и снижает нагрузку на сеть.
- REST может быть использован практически в любой среде, поддерживающей HTTP.
- REST легко интегрируется с веб-приложениями и мобильными устройствами.

REST имеет следующие ограничения:

- Требование полного указания контекста в каждом запросе может усложнить реализацию некоторых операций, таких как транзакции.
- В отличие от SOAP, REST не имеет встроенных спецификаций для обработки безопасности или транзакций, что требует дополнительных усилий для их реализации.

REST-сервисы широко используются в разработке API для веб- и мобильных приложений, например:

- Социальные сети, такие как Twitter и Facebook, предоставляют RESTful API для работы с их платформами.
- Электронная коммерция, где REST используется для обработки заказов, управления каталогами и интеграции платёжных систем.
- IoT (Интернет вещей), где REST применяется для управления устройствами и обмена данными.

REST-сервисы идеально подходят для современных приложений, где важны производительность, лёгкость интеграции и масштабируемость. Благодаря своей простоте и гибкости они остаются доминирующим подходом в разработке веб- и мобильных API.

1.4 Недостатки микросервисов

SOAP (Simple Object Access Protocol) и REST (Representational State Transfer) – это два подхода к реализации веб-сервисов, которые имеют различия в архитектуре, применении и производительности. Для выбора

наиболее подходящего подхода важно учитывать особенности и ограничения каждого из них.

Архитектура и подход

SOAP основан на строгих стандартах и протоколах, таких как XML и WSDL. SOAP использует строго структурированные сообщения и поддерживает сложные спецификации, включая WS-Security и WS-ReliableMessaging. Его архитектура предназначена для обработки сложных транзакций и обеспечения безопасности.

REST ориентирован на работу с ресурсами через стандартные HTTP-методы. REST не имеет строгих ограничений на формат данных и предоставляет лёгкость и гибкость в разработке.

Формат данных

SOAP использует исключительно XML для обмена данными, что увеличивает размер сообщений и замедляет их обработку.

REST поддерживает различные форматы данных (JSON, XML, YAML и др.), что позволяет выбирать более эффективный формат, например, JSON для лёгкости и быстродействия.

Транспортные протоколы

SOAP может работать с разными транспортными протоколами (HTTP, SMTP, TCP), что делает его универсальным для сложных сценариев.

REST зависит от HTTP, что ограничивает его использование только этим протоколом.

Производительность

SOAP более ресурсоёмкий из-за использования XML и сложных спецификаций, что может привести к увеличению времени обработки запросов.

Легковесность REST и использование JSON делают его более быстрым и менее ресурсоёмким, что особенно важно для мобильных и веб-приложений.

Безопасность

Встроенная поддержка WS-Security делает SOAP подходящим для сценариев, требующих высокого уровня защиты (например, банковские системы).

Для обеспечения безопасности REST требует дополнительных инструментов (например, OAuth для авторизации), что усложняет его настройку.

Гибкость и масштабируемость

SOAP менее гибок из-за строгой структуры сообщений и стандартов. Масштабируемость может быть ограничена из-за сложности сообщений.

Высокая гибкость REST позволяет легко адаптироваться к изменениям в требованиях системы, а простота кэширования и распределения нагрузки способствует масштабируемости.

Применение

SOAP подходит для корпоративных приложений, где важны надёжность, безопасность и обработка сложных операций (например, финансовые системы, обработка платежей).

REST идеален для публичных API, мобильных и веб-приложений, где требуется высокая производительность и простота интеграции.

Таблица 1. Сводная таблица сравнения SOAP и REST

Критерий	SOAP	REST
Архитектура	Протокол с жёсткими стандартами	Лёгкий архитектурный стиль
Формат данных	Только XML	JSON, XML, YAML и другие форматы
Транспорт	HTTP, SMTP, TCP	HTTP
Производительность	Низкая из-за тяжёлых сообщений	Высокая благодаря лёгкости
Безопасность	Встроенные стандарты безопасности	Требует внешних инструментов

Применимость	Корпоративные системы	Веб- и мобильные приложения

Можно сделать следующий вывод. Если требуется строгая безопасность и поддержка сложных операций, предпочтителен SOAP. Если важны производительность, гибкость и простота, REST будет лучшим выбором. REST чаще применяется в современных приложениях, но SOAP остаётся востребованным в корпоративной среде, где критичны стандарты и надёжность.

2 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ ДЛЯ ВЗАИМОДЕЙСТВИЯ С ВЕБ-СЕРВИСАМИ НА ОСНОВЕ SOAP И REST

2.1 Описание использованных инструментов

Программное окружение при выполнении работы:

1. Операционная система: Windows 10 Pro 64bit.
2. Программа выполняется в среде WSL (Windows Subsystem for Linux), что позволяет запускать Linux-программы в Windows.
3. На WSL установлена версия дистрибутива Linux (Ubuntu 20.04).
4. Серверная платформа для выполнения JavaScript-кода Node.js.
5. Менеджер пактов зависимостей npm (Node Package Manager).
6. Пакет soap для взаимодействия с SOAP-сервисами.
7. Пактет axios для выполнения HTTP-запросов.
8. IDE для разработки – Visual Studio Code с подключением к WSL.
9. Управление компиляцией и запуском программ осуществляется через командную строку WSL.

Аппаратное окружение:

1. Процессор 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz.
2. Установленная память (ОЗУ) 48 ГБ (47,7 ГБ доступно).
3. Тип системы 64-разрядная операционная система, процессор x64.

Чтобы запустить приложение, надо в корне проекта:

1. Установить зависимости с помощью команды «npm i».
2. Запустить проект с помощью команды «npm run services».

2.2 Описание реализованного приложения

В рамках реализации были созданы два модуля, каждый из которых выполняет задачу взаимодействия с конкретным типом веб-сервисов: SOAP и REST.

SOAP-запрос

Для взаимодействия с SOAP-сервисом был выбран публичный API, предоставляющий данные о числах в текстовом формате. Библиотека soap была использована для создания клиента, подключения к сервису и отправки запросов в формате XML. Реализация включала:

- Подключение клиента к WSDL-документу сервиса.
- Формирование запроса с необходимыми аргументами.
- Обработку ответа, извлечение данных и вывод результата в читаемом виде.

REST-запрос

Для REST-запроса использовался публичный API, предоставляющий данные в формате JSON. В реализации применялась библиотека axios, которая упрощает выполнение HTTP-запросов. Реализация включала:

- Отправку GET-запроса к REST API.
- Обработку JSON-ответа с помощью встроенных методов axios.
- Вывод данных из ответа в удобном формате для демонстрации.

Каждый из модулей был интегрирован в общий файл index.js, где обеспечивалась последовательная работа обеих функций. Таким образом, пользователь мог протестировать взаимодействие с SOAP- и REST-сервисами, запустив код одной командой.

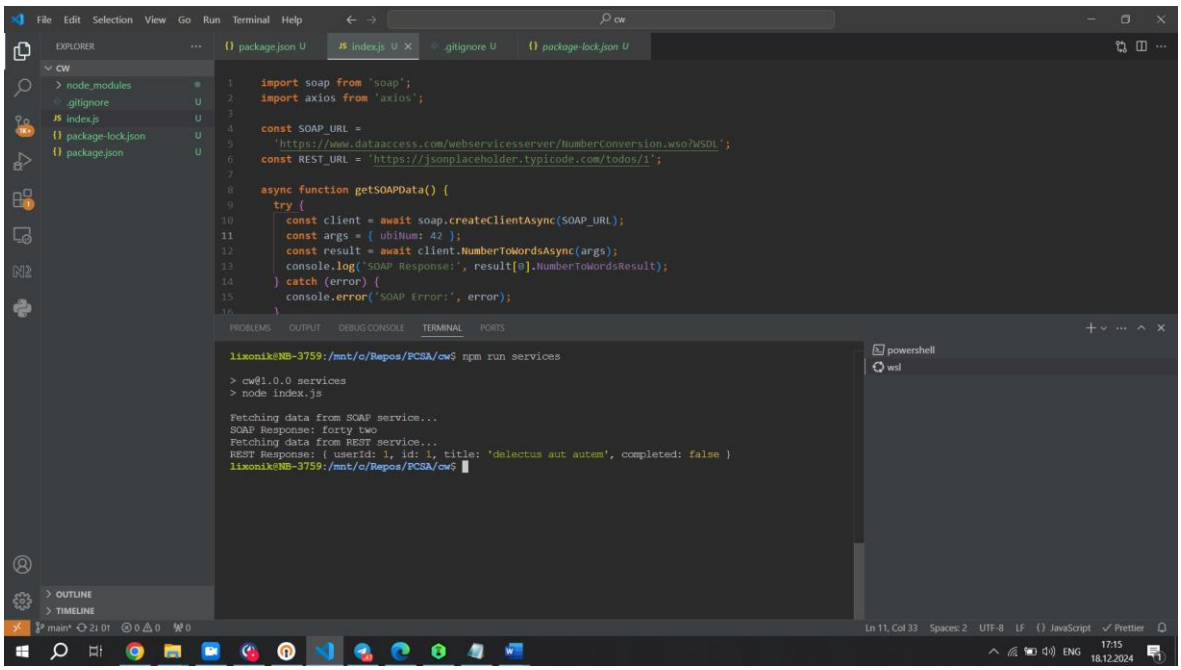

Рисунок 1 – Демонстрация работы реализованного приложения

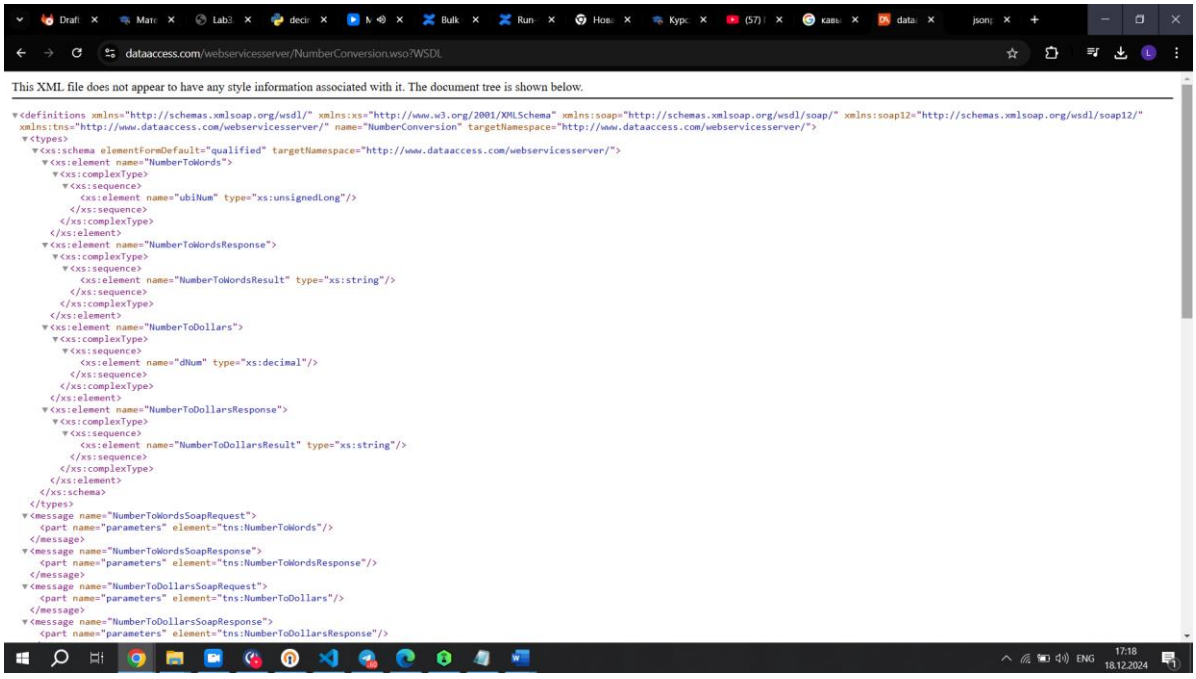

Рисунок 2 – Проверка корректности поставляемых данных. SOAP

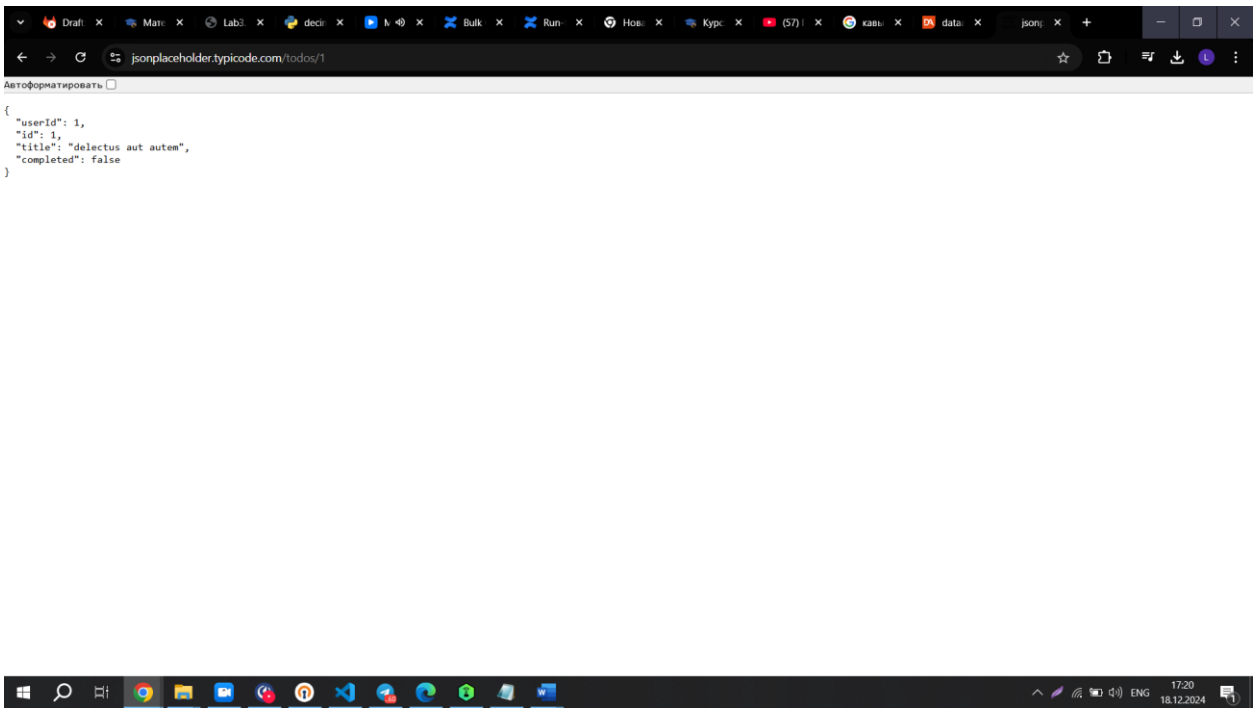

Рисунок 3 – Проверка корректности поставляемых данных. REST

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была проведена теоретическая и практическая оценка подходов к разработке веб-сервисов на основе SOAP и REST.

Во введении был обоснован выбор темы и определены основные цели и задачи исследования. В рамках основной части были рассмотрены теоретические аспекты веб-сервисов, ключевые особенности SOAP и REST, проведён их сравнительный анализ. В практической части разработаны примеры взаимодействия с веб-сервисами, которые продемонстрировали различия в подходах к реализации запросов и обработке ответов.

Полученные результаты показывают, что:

1. SOAP подходит для систем с высокими требованиями к безопасности, надёжности и стандартизации (например, банковские приложения и корпоративные решения).
2. REST является более гибким и производительным решением, особенно популярным в веб- и мобильной разработке благодаря поддержке JSON и простоте интеграции.

Практическая часть позволила наглядно увидеть различия между SOAP и REST, а также получить опыт работы с библиотеками soap и axios. Эти инструменты показали, что несмотря на сложность реализации SOAP-запросов, современные библиотеки значительно упрощают работу с этим протоколом.

Итоги работы подтверждают, что выбор между SOAP и REST зависит от конкретных требований системы. REST может быть предпочтительным для большинства приложений благодаря лёгкости и быстродействию, однако SOAP сохраняет свою актуальность в сложных корпоративных сценариях.

Перспективы дальнейшего изучения темы включают исследование гибридных подходов и современных стандартов API, таких как GraphQL,

которые сочетают преимущества REST и SOAP, а также изучение новых инструментов для их реализации.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Fielding, R. T. Architectural Styles and the Design of Network-based Software Architectures [Электронный ресурс]. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (дата обращения: 18.10.2024).
2. W3C. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [Электронный ресурс]. URL: https://www.w3.org/TR/soap12/ (дата обращения: 11.10.2024).
3. Paulraj Ponniah. Data Warehousing Fundamentals for IT Professionals, Second Edition [Электронный ресурс]. URL: https://www.wiley.com/en-us/Data+Warehousing+Fundamentals+for+IT+Professionals%2C+2nd+Edition-p-9780470462072 (дата обращения: 17.10.2024).
4. Stack Overflow. REST vs SOAP: Key Differences and Use Cases [Электронный ресурс]. URL: https://stackoverflow.com/questions/ (дата обращения: 23.10.2024).
5. Node.js Documentation [Электронный ресурс]. URL: https://nodejs.org/docs (дата обращения: 30.10.2024).
6. npm package soap. Node.js SOAP client for invoking web services and a mock-up SOAP server capability [Электронный ресурс]. URL: https://www.npmjs.com/package/soap (дата обращения: 02.11.2024).
7. npm package axios. Promise based HTTP client for the browser and node.js [Электронный ресурс]. URL: https://www.npmjs.com/package/axios (дата обращения: 10.11.2024).
8. JSONPlaceholder. Free Fake API for Testing and Prototyping [Электронный ресурс]. URL: https://jsonplaceholder.typicode.com/ (дата обращения: 24.11.2024).

9. Data Access. Number Conversion SOAP Service [Электронный ресурс]. URL: https://www.dataaccess.com/webservicesserver/ (дата обращения: 25.11.2024).
10. IBM. Understanding SOAP and REST Basics [Электронный ресурс]. URL: https://www.ibm.com/cloud/learn/soap-vs-rest (дата обращения: 30.11.2024).
11. Microsoft Docs. SOAP and REST API Design Best Practices [Электронный ресурс]. URL: https://learn.microsoft.com/ (дата обращения: 06.12.2024).

ПРИЛОЖЕНИЕ А. ЛИСТИНГ КОДА

Название файла: index.js

```
import soap from 'soap';
import axios from 'axios';

const SOAP_URL =

'https://www.dataaccess.com/webservicesserver/NumberConversion.wso?WSD
L';
const REST_URL = 'https://jsonplaceholder.typicode.com/todos/1';

async function getSOAPData() {
  try {
    const client = await soap.createClientAsync(SOAP_URL);
    const args = { ubiNum: 42 };
    const result = await client.NumberToWordsAsync(args);
    console.log('SOAP Response:', result[0].NumberToWordsResult);
  } catch (error) {
    console.error('SOAP Error:', error);
  }
}

async function getRESTData() {
  try {
    const response = await axios.get(REST_URL);
    console.log('REST Response:', response.data);
  } catch (error) {
    console.error('REST Error:', error);
  }
}

async function main() {
  console.log('Fetching data from SOAP service...');
  await getSOAPData();

  console.log('Fetching data from REST service...');
  await getRESTData();
```

```
}

main();
```

Название файла: package.json

```
{
  "name": "cw",
  "version": "1.0.0",
  "type": "module",
  "description": "",
  "main": "index.js",
  "scripts": {
    "services": "node index.js"
  },
  "keywords": [],
  "author": "Boris Prokhorov",
  "license": "MIT",
  "dependencies": {
    "axios": "^1.7.9",
    "soap": "^1.1.7"
  }
}
```