СТАТЬИ

УДК 004.724.3

ОПТИМАЛЬНАЯ ИНТЕГРАЦИЯ УДАЛЕННЫХ КОМПОНЕНТОВ СИСТЕМ В ПРИЛОЖЕНИЯХ НА БАЗЕ АРХИТЕКТУРЫ КЛИЕНТ-СЕРВЕР

Драч В.Е., Косян Л.О., Лях А.М., Чукаев К.Е.

ФГБОУ ВО «Сочинский государственный университет», Сочи, e-mail: lyakh.lina.mail.com@mail.ru

В работе раскрываются общие аспекты взаимодействия компонентов информационных систем, приводится понятие интеграции систем, проводится анализ различных способов интеграции, подробно рассмотрен наиболее распространенный метод интеграции удаленных компонентов, обоснована фактическая эффективность данного метода на примере развернутой децентрализованной информационной системы. В результате выполнения данной работы была разработана система онлайн-кинотеатра, сделан выбор архитектурного стиля для интеграции компонентов. Статья содержит анализ различных методов интеграции, общий обзор наиболее популярного и эффективного метода интеграции, а также описание положительного опыта внедрения метода интеграции в коммерческую разработку через развернутую децентрализованную информационную систему. Описаны способы интеграции различных систем: через общую базу данных, передачу файлов, асинхронный поток сообщений или через API. Дается общее понятие API, приводятся популярные методы реализации API: протокол SOAP и архитектурный стиль REST; методы интеграции REST и SOAP имеют сходство в типичном использовании протокола HTTP, но содержат массу отличий, например в форматах запроса/ответа. Представлены результаты применения REST API в практической разработке системы онлайн-кинотеатра. Проанализированы преимущества выбора технологии REST перед протоколом SOAP в контексте удобства использования, простоты разработки, временных затрат разработчиков, предпосылок интеграции, времени обработки ответа, а также ряда других аспектов. Выполнено обсуждение теоретических и практических результатов исследования. Сделан вывод о преимуществе архитектуры REST перед методом интеграции SOAP по ряду интегральных показателей.

Ключевые слова: SOAP, API, REST, интеграция, протокол, запрос, клиент, сервер, микросервис, архитектура приложения

OPTIMAL INTEGRATION OF REMOTE SYSTEM COMPONENTS IN APPLICATIONS CLIENT-SERVER ARCHITECTURE

Drach V.E., Kosyan L.O., Lyakh A.M., Chukaev K.E.

Sochi State University, Sochi, e-mail: lyakh.lina.mail.com@mail.ru

General guidelines of remote system components interaction and system integration are being explained. The article contains analysis of different integration methods, general review of the most popular and efficient integration technique, as well as depiction of positive results of the integration method's implementation in commercial development via deployed decentralized information system. Presented article gives common introspection on microservices' functioning principles. In the introductory part it is also described how several varied systems can be integrated with one another: through common database, API, file transfer, or asynchronized message flow. The article gives a general concept of API, brings up popular API implementation methods: SOAP protocol and REST architectural style. In this paragraph it is explained that REST and SOAP integration methods have similarities in typical HTTP protocol usage and differences in request/response formats, as well as plenty others. The performance of the REST API in practical development of online cinema system is presented. The benefits of choosing REST technology over SOAP protocol are investigated in terms of user experience, usability, response processing time, developers' time consumption, simplicity of development, integration preconditions, as well as few others. After discussion of theoretical and practical researches' results, it is possible to note, that the positive aspects of REST technology has an advantage over SOAP integration method, based on the spectrum of integral parameters.

Keywords: SOAP, API, REST, integration, protocol, request, client, server, microservice, application architecture

Несмотря на то что однородные системы нетребовательны к комплексной поддержке, а также надежны и просты в исполнении, в настоящее время на передний план выходит иная парадигма разработки, базирующаяся на проектировании сложных систем как совокупности микросервисов. Микросервис (МС) – независимый сервис, отвечающий за один элемент логики в рамках конкретного функционала. Микросервисы взаимодействуют между собой через ряд интерфейсов, но не обладают информацией о внутреннем устройстве соседних сервисов. Каждый МС автономен и может функционировать отдельно от других частей системы. МС и другие элементы многозвенной информационной системы могут быть написаны на разных языках программирования и, как правило, не могут взаимодействовать между собой напрямую, пользуясь общими данными или переменными. Для взаимодействия разнородных систем, микросервисов и их элементов выполняется процесс, называемый интеграцией систем, – это организация взаимодействия по строго определенному протоколу.

Вид взаимодействия компонентов информационной системы определяется ее архитектурой – моделью системы, учитывающей роли различных элементов, которые концептуально группируются в слои или уровни. Наибольшее распространение получило количество уровней, равное трем. В трехуровневой архитектуре части системы разделены на три физических и логических уровня: уровень данных (предназначен для хранения и управления данными, заключающими в себе информацию о компонентах и пользователях приложения [1]), уровень приложения (на котором осуществляется обработка данных) и уровень представления (пользовательский интерфейс). В качестве уровня представления может выступать браузер, в качестве уровня приложения – сервер, а в качестве уровня данных – непосредственно база данных системы.

Для того чтобы связать эти уровни в общее информационное пространство, существуют четыре основных интеграционных подхода: асинхронный обмен сообщениями, обмен на уровне файлов [2], общая база данных [3, 4] или через удаленный вызов процедур по API.

Целями исследования являются нахождение оптимальной интеграции удаленных компонентов информационных систем в приложениях на базе архитектуры клиент-сервер, а также определение наиболее эффективной архитектуры при сравнении REST и SOAP. Как правило [5, 6], SOAP встречается в крупных разработках, проводимых в строгом соответствии с ГОСТ 34.602-89, но в коммерческой разработке малого и среднего звена архитектура REST реализуется чаще. Во многом это обусловлено тем, что при разработке, не подпадающей под действие ГОСТ, SOAP значительно усложняет [7, 8] работу команды тестировщиков и команды аналитиков, это ощутимо сказывается на сроках средних и малых проектов, в которых на разработку отводится от полугода до года. Один из подобных реализованных нашей командой проектов – система онлайн-кинотеатра, доступного пользователям через любой веб-браузер. Для достижения цели исследования была проанализирована разработка онлайн-кинотеатра, выполненная в коммерческих целях.

Материал и методы исследования

Разработка онлайн-кинотеатра потребовала участия команды специалистов. В состав команды разработчиков вошли проект-менеджер, системный аналитик, разработчик-программист, инженер для развертывания серверного оборудования и ПО и тестировщик. Для оценки проекта (включая предварительную оценку и согласование стоимости на разработку приложения) на предмет трудозатрат в человеко-часах было выделено полмесяца. Этап проектирования интеграционного взаимодействия через сервисы REST был оценен в одну неделю (описание методов, их расположение и структурирование, тестирование запросов и ответов).

Документирование спецификации к методам в сервисе Swagger было сделано уже после приемки проекта. Использование SOAP-протокола предполагало бы написание WSDL-документа с описанием каждого запроса и ответа для каждой из функций, проектирование XSD-схем для форматирования XML-запросов и определения XML-ответов в различных инстанциях, что уже на стадии проектирования и аналитики увеличивает сроки работы команды с одной недели до четырех недель.

Оценка при этом получается менее достоверной, поскольку при итерационной разработке требования к проекту обновляются каждые 2 недели, и управление изменениями в WSDL-документации обходится гораздо дороже, чем в спецификации к REST через OpenAPI.

Пользовательский интерфейс выполнен с помощью фреймворка Angular языка TypeScript, который является разновидностью языка JavaScript. Поскольку язык TypeScript строго типизированный, интеграция частей REST API непосредственно в программный код интерфейса также упростила (а значит, удешевила и ускорила) разработку.

Необходимо было организовать взаимообмен данными между веб-браузером, включенными в него МС пользовательского приложения, панелью администрирования, несколькими базами данных для каждого из МС и общим сервером сайта. С помощью REST-сервиса было организовано взаимодействие четырех компонентов в архитектуре клиент-сервер: прокси-серверов, клиентов, сетевых шлюзов и основных серверов приложения.

Общая структурная схема архитектуры МС представлена на рисунке 1.

При этом, поскольку данный стиль архитектуры придерживается принципов максимальной доступности и простоты, инструментарий REST оказал положительное воздействие на субъективное восприятие пользователей приложения.

Архитектурный стиль интерфейса REST позволил кратно уменьшить нагрузку на аппаратные ресурсы конечного пользователя.

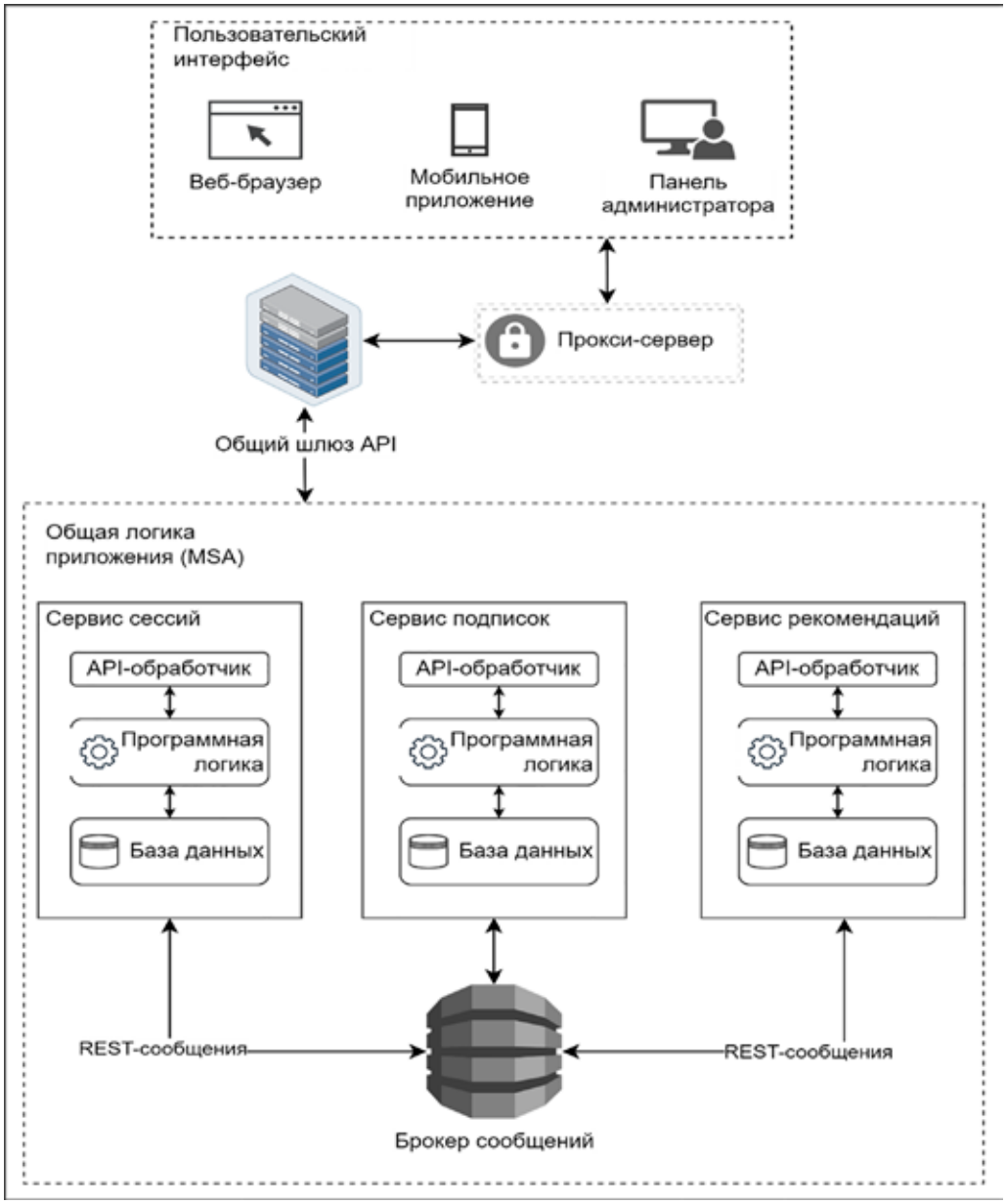

Рис. 1. Общая структурная схема архитектуры МС онлайн-кинотеатра

Генерация запросов и ответов была выполнена в рамках протокола передачи HTTP без предопределенной структуры и зависимости от предустанавливаемых трассировщиков запросов-ответов API. Интеграция была проведена без дополнительных компонентов обеспечения взаимодействия, что позволило оптимизировать скорость получения и обработки данных (отдельно следует отметить вклад языка разметки XML).

Интеграция всех модулей по общему API посредством технологии REST снизила объем расходования трафика, поскольку форматы данных в REST не имеют значения, а каждый информационный ресурс определялся уникальным адресом. Генерация файлов в формате XML требовательна к ресурсам и времени выполнения, если сравнивать с передачей информации без разметки и форматирования, поэтому пропускная способность общей сети оказалась выше, чем в аналогичной информационной системе с архитектурным стилем SOAP.

Общая разработка информационной системы от момента оценки до момента введения в эксплуатацию заняла восемь месяцев, во многом благодаря простоте интеграции компонентов. Сравнительный анализ интеграционных технологий показывает, что использование REST является предпо-

чтительным при построении приложений на базе МС с удаленными модулями системы, а SOAP нельзя рекомендовать для новых разработок.

Результаты исследования и их обсуждение

В результате выполнения работы удалось наглядно сравнить методы интеграции и их эффективность на примере разработки реального коммерческого проекта. Данное исследование показывает, что методы интеграции систем и их компонентов посредством технологий REST и SOAP в корне отличаются друг от друга и применимы для разных целей, однако сервисы REST явно предпочтительнее за счет значительного снижения нагрузки на сервер, экономии во времени на проектирование интеграции и написание спецификаций, большей гибкости, передачи данных без дополнительных внутренних прослоек, а также более простого формата отправки запросов и анализа ответов.

Сравнение эффективности архитектурных стилей SOAP и REST на разных стадиях жизненного цикла продукта на примере разработки информационной системы онлайн-кинотеатра представлено на рисунке 2. Сбор и анализ данных (рис. 2) позволяют констатировать уверенное превосходство технологии REST.

В результате выполнения данной работы была разработана информационная система онлайн-кинотеатра, интеграция компонентов в которой была осуществлена на основе архитектурного стиля REST. Взаимодействие каждого МС системы с другими элементами выполнено по единому контракту API, передача реализована без дополнительного форматирования, а прогнозируемая дополнительная нагрузка не повлияет на пропускную способность системы за счет установки балансировщика нагрузки (рис. 3).

По итогу внедрения данной разработки достигнуты практические результаты коммерческого пользования за минимальные сроки; информационная система была протестирована вручную, без автоматических A|B тестов, однако тестирование было завершено в течение нескольких дней ввиду простоты тестирования функционала системы. Тестировщиками и конечными пользователями были отмечены интуитивно понятное взаимодействие с программными модулями системы и обработка ошибок различных категорий (возникавших на стороне сервера, клиента, прокси-сервера и т.д.).

Сравнительный анализ двух технологий REST и SOAP приведен в таблице.

В настоящее время выполняется поисковая оптимизация веб-сайта [9, 10], а процесс разработки информационной системы успешно завершен.

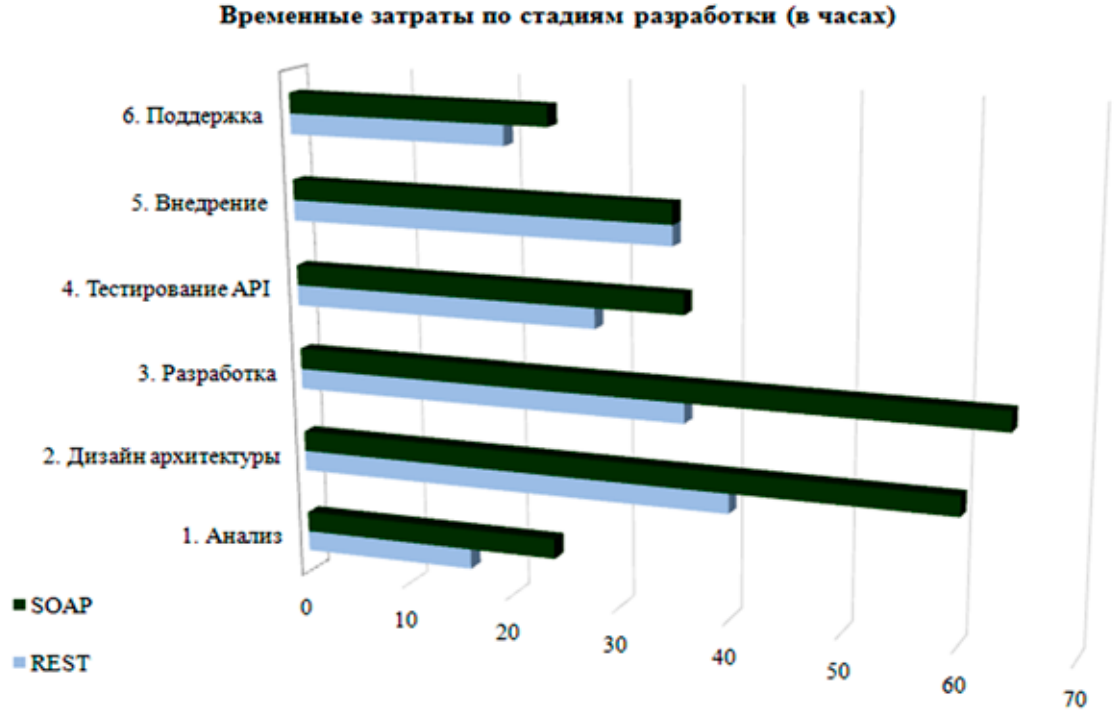

Рис. 2. Эффективность REST/SOAP в жизненном цикле разработки информационной системы

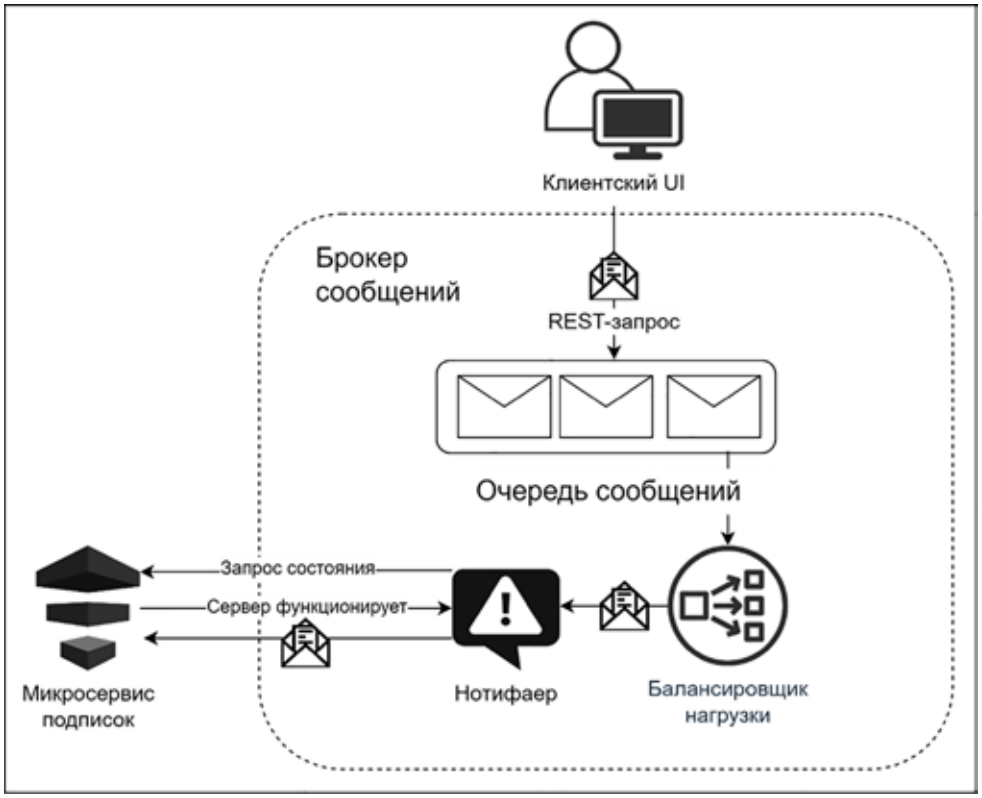

Рис. 3. Схема структурная информационной системы

Сравнительный анализ двух технологий – REST и SOAP

Параметр для сопоставления	Архитектурный стиль интерфейса	
	REST	SOAP
Ограничения структуры сообщений	Требуются только метод и адрес	Требуются заголовок, тело пакета и пространство имен, информация об ошибках не является обязательной
Язык разметки	JSON	XML
Протоколы передачи данных	Исключительно HTTP или HTTPS	HTTP(s), FTP, MQ, SMTP и др.
Спецификация	YAML (возможно, но не обязательно)	Требуется подключение WSDL с определением данных и метаданных
Шаблоны проектирования	CRUD (4 типа запросов)	Отсутствуют
Кэш на стороне сервера	На основе протокола	Отсутствует
Обработка ошибок	Опирается на коды ошибок протокола	Стандартизируется уникально по WSDL

В будущих исследованиях интересной с научной точки зрения задачей будет проведение аналогичного анализа эффективной интеграции компонентов, но уже на прикладной задаче, выполняемой в строгом соответствии с ГОСТ [11].

Заключение

В результате выполнения данной работы была разработана система онлайн-кинотеатра, причем для интеграции компонентов был обоснованно выбран архитектурный стиль REST. Коммерческое использование системы стало возможным сразу после разработки и тестирования, ограниченных жесткими временными рамками.

Опыт интеграции компонентов по архитектурному стилю взаимодействия REST можно считать положительным и вдохновляющим на предпочтительное использо-

вание технологии REST перед протоколом SOAP для решения интеграционных задач.

Список литературы

1. Копырина А.О., Копырин А.С. Концепция хранения данных в экономической экспертной системе с применением гибридных технологий // Управленческий учет. 2021. № 10-1. С. 46-52.

2. Лебедкина Т.В., Соколовский С.П. Модель функционирования защищенной технологии файлового обмена // Вопросы кибербезопасности. 2021. № 5 (45). С.52-62.

3. Драч В.Е., Родионов А.В., Чухраева А.И. Выбор системы управления базами данных для информационной системы промышленного предприятия // Электромагнитные волны и электронные системы. 2018. Т. 23, № 3. С.71-80.

4. Драч В.Е., Ильичев В.Ю. Анализ популярных реляционных систем управления базами данных // Системный администратор. 2021. № 12 (229). С.60-65.

5. Коптяев К.Р., Ходжатов Т.Б., Назаров И.В. Анализ интерфейса взаимодействия REST // Научные горизонты. 2019. № 3 (19). С.148-152.

6. Ильичев В.Ю., Драч В.Е., Кушнир А.С. Сравнительный анализ технологий REST и SOAP для решения интеграционных задач // Системный администратор. 2022. № 4 (233). С. 86-89.

7. Маквитти Л. REST как альтернатива SOAP // Сети и системы связи. 2007. № 1. С. 73-74.

8. Сидорова А.П., Афзалова А.Н. Анализ и распространенность протоколов SOAP, REST для сбора данных на территории Российской Федерации // Меридиан. 2020. № 8 (42). С. 66-68.

9. Дивина О.И. Возможности digital-маркетинга в системе управления компанией // Научные записки академии. 2023. № 1 (45). С. 64-68.

10. Буторин В.А. Основные подходы к продвижению товаров и услуг в цифровом маркетинге // Конкурентоспособность в глобальном мире: экономика, наука, технологии. 2023. № 4. С. 225-229.

11. Шеметов А.С., Кириенко О.В., Горчаков А.А. Программные комплексы сопровождения жизненного цикла РЗА и АСУ ТП // Электроэнергия. Передача и распределение. 2023. № 2 (77). С. 72-78.