ПОСТРОЕНИЕ КОМПОЗИТНЫХ RESTFUL-СЕРВИСОВ С ДЕЦЕНТРАЛИЗОВАННЫМ УПРАВЛЕНИЕМ, РЕАЛИЗУЮЩИХ БИЗНЕС-ПРОЦЕССЫ С ДЛИТЕЛЬНЫМИ ОПЕРАЦИЯМИ

Поляков А.А., Федорченко В.Н.

Харьковский национальный экономический университет имени Семена Кузнеца, Харьковский национальной университет радиоэлектроники

Современное приложение ориентировано на Web и мобильного клиента. Такой тренд изменил архитектуру приложения, в которой вся логика представления перенесена на клиента (браузер, мобильное приложение), т.е. реализация MVC вынесена на клиента. В серверной части приложения сохраняется доступ к данным и бизнес-логика (уровень служб), а для взаимодействия с клиентами появляется слой RESTful Web API сервисов. Необходимые условия для построения распределённых REST-приложений по Филдингу[1]: клиент-серверная архитектура, отсутствие состояния, кэширование, слои системы, код по требованию, единообразие интерфейса. Основной элемент REST – ресурс, адресуемый по ссылке через URI, а состояние которого, передается через представления HTML и другие медиатипы [2]. REST-сервис можно рассматривать как набор таких ресурсов, которые обеспечивают когерентный доступ к состоянию и функциональности программного компонента. Бизнес-процесс определяет набор взаимодействий между несколькими сервисами, которые необходимы для достижения цели; взаимодействия сервисов регулируются с помощью простых и составных шаблонов управления, которые определяют частичный порядок операций вызова сервиса.

Взаимодействие между архитектурными компонентами REST основано на обмене сообщениями по протоколу HTTP как основного, а управление в REST основано на кодах состояния. Коды стояния переадресации играют особую роль, позволяя серверу лучше контролировать взаимодействие, перенаправив клиента на дополнительный ресурс. Переадресация выполняется автоматически в случае запросов, инициированных методами GET или HEAD; в противном случае требуется подтверждение от пользователя для выполнения запроса. Если приложение работает в автоматическом режиме, то оно должен понимать последствия перенаправления на уровне домена или слепо следовать перенаправлению. Такое свойство эффективно для построения синхронных блокирующих сервисов.

Синхронный блокирующий сервис подразумевает, что для продолжения взаимодействия клиенты должны ожидать ответа сервера, т.е. после отправки запроса клиент прерывает свою работу и переходит в состояние ожидания ответа сервера. Такое решение приемлемо для случая, если связь клиент-сервер остается открытой до тех пор, пока сервер формирует ответ, но это неприемлемо для сервисов с большой латентностью, используемых в длительных бизнес-процессах. Под длительными бизнес-процессами следует понимать процессы, выполняющиеся в течение длительного периода времени, использующие слабо связанные сервисы, которые могут взаимодействовать с различными контекстами (среда сервисов организации) другой организации, и взаимодействие координируется третей стороной [3].

Традиционно, в централизованном подходе к архитектуре композитные сервисы WSDL/SOAP и RESTful реализуются после проведения базовой оркестровки. Централизованный подход не только накладывает ограничения масштабируемости, производительности и доступности композитных сервисов, он также ограничивает взаимодействие компонентов сервиса, то есть, компонент разрабатывается с учетом различных сценариев использования, что приводит к снижению его доступности и соответствию REST-принципам. Централизованный подход требует, чтобы на сервере отслеживалось состояние взаимодействия между составными ресурса и их компонентами на стороне сервера [4]. Такой подход подразумевает стиль с отслеживанием состояния, который нарушает ограничения REST и оказывает негативное влияние на масштабируемость сервиса. Включение дополнительной (сложной) логики освобождения ресурсов сервера в композитный сервис позволит решить ситуацию, т.е. сервис закрывает активные соединения с композитным сервисом и периодически запрашивает доступность ответа. В случае получения отрицательного результата выполнения бизнес-процесса, необходимо будет выполнить несколько дополнительных транзакций для восстановления сервисов в их предыдущее состояние, или сервис может закрыть соединение с клиентом и разрешить сохранить идентификатор экземпляра бизнес-процесса, чтобы позже связаться с сервисом. Такой подход применяется в композитных WSDL/SOAP сервисах. Несмотря на то, что децентрализованный подход улучшает масштабируемость и снижает связанность компонентов, остается вопрос поддержки состояния композитного сервиса. Если предположить, что контроль управления распределен между компонентами сервиса, то остается необходимость в координации их поведения для решений целей бизнес-процесса.

Альтернативный подход полагаться на полностью децентрализованный стиль, основанный на асинхронных сообщениях и сообщениях обратного вызова. В сообщения с кодом состояния 303 добавляется заголовок *Callback* с указанием адреса обратного вызова на соответствующий ресурс композитного сервиса, к которому должен обратиться клиент после завершения работы с ресурсом, указанным в поле *Location*. Обработка заголовка *Callback* на стороне клиент в браузерах выполняться с помощью скриптов *JavaScript* или платформы-зависимого кода мобильного приложения. В этом случае поддержка состояния переносится на клиента, и сервис соответствует всем требованиям REST.

1. Fielding R. T. Architectural styles and the design of network-based software architectures : Ph.D., University of California, Irvine. – 2000. – 162 с.

2. Richardson L. RESTful Web Services / L. Richardson, S. Ruby. – O'Reilly, 2007. – 448 p.

3. Dayal U. Business Process Coordination: State of the Art, Trends, and Open Issues / U. Dayal, M. Hsu, R. Ladin // In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01) – 2001, – № 1. – 3-13 с.

4. Pautasso C. On composing RESTful services // In Software Service Engineering. Dagstuhl Seminar Proceedings – 2009, – № 09021. – 1-7 с. URI: http://drops.dagstuhl.de/opus/volltexte/2009/2043