УДК 004.942

ПРЕИМУЩЕСТВА МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Полховский Е.С.
Научный руководитель - Попова Ю.Б., доцент, к.т.н.

Цель работы – обосновать преимущества разделения ответственности между внешним представлением и внутренней реализацией программного продукта за счет применения микросервисной архитектуры.

Микросервисная архитектура приложений реализует слабо связанные сервисы, взаимодействующие друг с другом для выполнения задач, относящихся к их бизнес-возможностям[1].По сравнению с монолитом в микросервисах есть несколько единиц развертывания (каждый сервис развертывается самостоятельно).

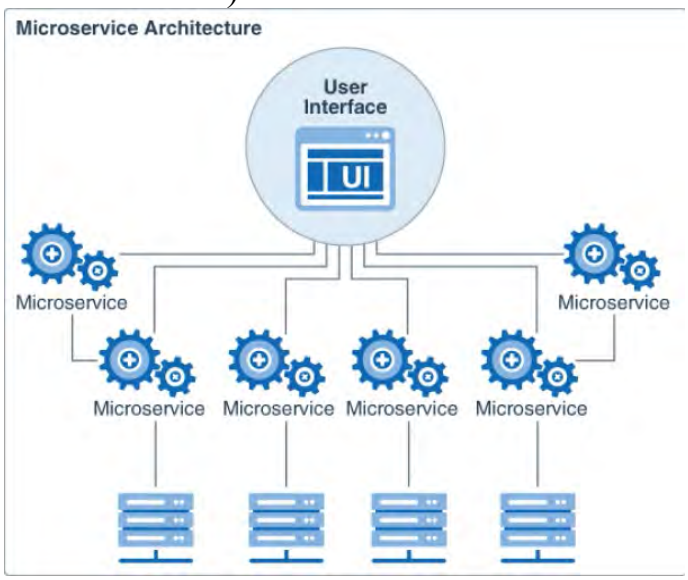

Рисунок 1 – Схема микросервисной архитектуры приложений [1]

Анализ литературных источников позволил выделить следующие преимущества микросервисной архитектуры приложений перед монолитной[1-2]:

1. Микросервисы легче реализовать модульными. Технически это обеспечивается жесткими границами между отдельными сервисами.

2. В больших компаниях разные сервисы могут принадлежать разным командам, но могут быть повторно использованы всей компанией. Нет необходимости координировать развертывание между командами.

3. Микросервисы имеют меньшие размеры, поэтому их легче сопровождать и проверять. Уменьшается время компиляции, время запуска и выполнения тестов. Все эти факторы влияют на производительность разработчика, так как позволяют затрачивать меньше времени на ожидание на каждом этапе разработки.

4. Микросервисы не привязаны к технологии, используемой в других сервисах, следовательно можно использовать лучшие технологии подгонки. Старые сервисы могут быть быстро переписаны для использования новых технологий.

5. В микросервисах изолируемые разломы лучше по сравнению с монолитным подходом. Хорошо спроектированная распределенная система переживет сбой одного сервиса.

Практическая реализация данного исследования проходила на программном модуле для тестирования знаний обучающей системы CATS [3].С использованием современных технологий фреймворкаAngular была разделена клиентская и сервернаялогика, т.е. архитектура стала микросервисной. Фреймворк Angular прост в понимании и изучении, подходит для разработки больших приложений. Angular позволяет привязывать данные к HTML с помощью выражений, а директивы Angular позволяют разработчикам расширять функциональности HTML и создавать новые конструкции. Манипуляции с DOM и код привязки данных обернуты в простые элементы, которые можно быстро и просто вставить в HTML шаблон.В итоге рассматриваемый программный модуль перестал быть зависимым от других модулей системы и может работать отдельно, т.е. в любой момент можно перейти на новую клиентскую часть, не касаясь при этом серверной части. При использовании нового фреймворка уменьшается объем загружаемых ресурсов в несколько раз. ИспользованиеAhead-of-time компилятора позволяет получить следующие плюсы:

1. Быстрая загрузка в браузере за счет того, что:

a) приложение компилируется до загрузки в браузер;

b) в сборку не включается компилятор Angular и, как следствие, конечные файлы имеют меньший размер;

c) выполняется меньше AJAX-запросов на получение исходных HTML- и CSS-файлов, поскольку они включаются в строковом виде в файлы JavaScript.

2. Обнаружение ошибок при сборке. Имеется возможность исправить все ошибки до запуска приложения в режиме эксплуатации.

3. Повышенная безопасность. Поскольку HTML- и CSS-файлы включаются в процессе Angular компиляции в файлы JavaScript, то нет возможности просмотреть шаблоны, что снижает риск осуществления атак.

Рассмотрим в качестве параметра сравнения время загрузки страницы с тестами старой и новой системы (т.е. монолитной и микросервисной архитектуры). В старой системеэто время составляло 2,78 секунды (Рисунок 2). В новой системескорость загрузки этой же страницы получается 1,34 секунды (Рисунок 3). Таким образом, применение микросервисной архитектуры приложения в совокупности с возможностями фреймворка Angular позволило увеличить быстродействие загрузки страниц в 2 раза.

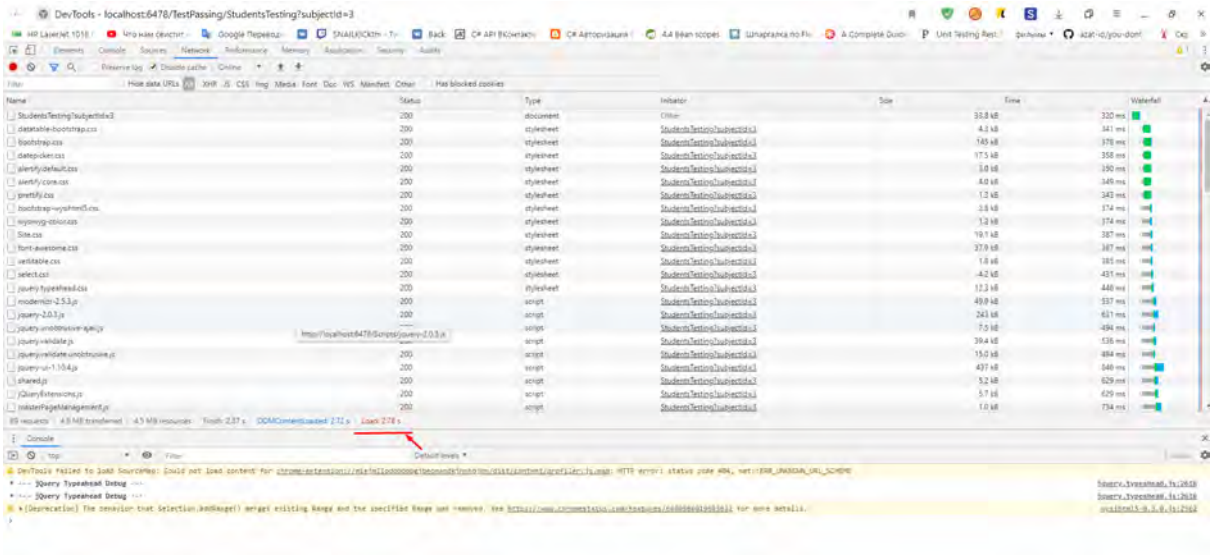

Рисунок 2 – Данные о загрузке страницы с тестами старой системы

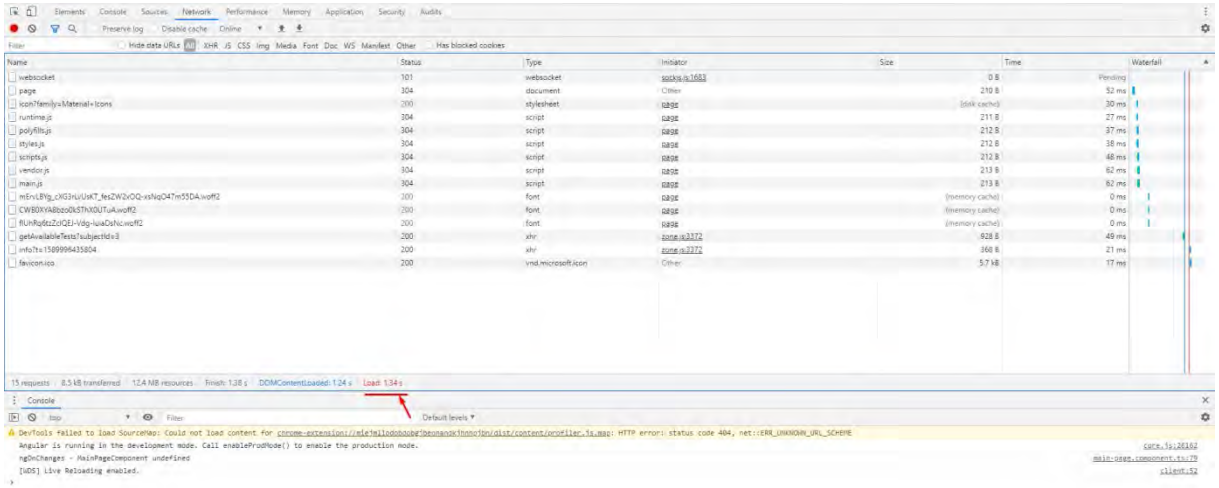

Рисунок 3 – Данные о загрузке страницы с тестами новой системы

Литература

1. Монолитная vsМикросервисная архитектура [Электронный ресурс]. – Режим доступа: https://dailycoding.io/article/4BABLqr0nR9E0gUU7Ko6 – Дата доступа: 19.05.2020.

2. Почему лучше разделить фронтенд и бэкенд [Электронный ресурс]. – Режим доступа: https://bureau.ru/soviet/20200116/ – Дата доступа: 19.05.2020.

3. Попова, Ю.Б. Автоматизированная система управления обучением CATS (CareAboutTheStudents) / Ю.Б. Попова // Наука и техника. – 2019. – №4 (18). – С. 339-349.