УДК 004

Микросервисная архитектура приложения

Голубь Илья Сергеевич
Приамурский государственный университет имени Шолом-Алейхема
Магистрант

Аннотация
В данной работе проведено исследование микросервисной архитектуры. Был использован метод сравнения, а так же приведены примеры недостатков микросервисной архитектуры. Были сделаны выводы о том, что микросервисная архитектура как облегчает разработку и поддержку, так и усложняет их.
Ключевые слова: Микросервис, архитектура приложений, языки программирования

Microservice Application Architecture

Golub Ilya Sergeevich
Sholom Aleichem Priamursky State University
Undergraduate

Abstract
In this paper, a study of microservice architecture. A comparison method was used, as well as examples of the disadvantages of the microservice architecture. It was concluded that microservice architecture both facilitates development and support, and complicates them.
Keywords: microservice, application architecture, programming languages

Данная тема актуальна, в связи с тем, что в данный момент все средние и крупные предприятия стараются уйти от монолитной архитектуры, к микросервисной.

Цель работы – исследование микросервисной архитектуры

В статье Опарина Г. А. Богдановой В. Г., Пашинин А. А. рассматриваются вопросы применения микросервисного подхода для конструирования и поддержки функционирования распределенных решателей вычислительных задач на модели предметной области[2].

В работе Джамшиди П., Льюис Д., Паль К., Мендонча Н., Тилков С. говорится о том, что каждый модуль-микросервис реализуется и работает как малая полностью независимая система, что способствует гибкости разработки, развертывания, эксплуатации, управления версиями и масштабирования[1].

Статья Стоянченко С. С. и Полищук М. А. посвящена разработке алгоритмов повышения быстродействия информационной системы

медицинского ассистанса при помощи RESTful микросервисов. Произведен анализ факторов, влияющих на использование микросервисной архитектуры. Предложена технология использования «метода хореографа» для повышения контроля над передаваемыми асинхронными запросами[5].

Программный комплекс NCRAWLER предназначен для сбора неструктурированной, открытой информации из разных источников сети интернет (товарах, их технические характеристиках, фотографии, цены, названия и иные заданные пользователем характеристики) и анализа полученной информации. Функциональные особенности: автоматическое извлечение необходимой информации со страницы HTML; автоматическая классификация и структурирование полученной информации; автоматическое переполучение/обновление имеющейся информации; автоматический поиск новых источников информации. Посредством программного комплекса пользователь может: осуществлять доступ ко всей информации собранной из сети интернет посредством веб-браузера в структурированном виде; задавать приоритеты и новые сегменты для индексации; выбирать интересующие товары с получением данных на источник информации; осуществлять автоматический контроль цен на заданные товары и получать разного рода оповещения при фиксации изменений, попадающие по заданные пользователем условия[4].

В работе Жевнерчук Д. В., Рассадин О. С. Реализованы алгоритмы синтеза моделей аппаратных ресурсов микросервисов, решающих задачи дискретной математики. Входными данными являются: ресурсоемкость микросервисов для монопольного режима функционирования, модели потоков пользовательских запросов, критериальные оценки режимов функционирования. Результатами работы программы являются параметры аппаратных ресурсов для поддержки режима работы микросервисов определяемого потоком запросов. Система представляет собой совокупность модулей, которые можно использовать как самостоятельный инструментарий, так и в качестве встраиваемой библиотеки. Комплекс должен быть нацелен на использование в учебном и промышленном применении[3].

Методом исследования будет сравнение микросервисного сервера с сервером для монолитного приложения. Микросервисная архитектура — это подход, при котором цельное приложение строится как набор сервисов, каждый из которых работает изолированно, и коммуницирует с остальными используя механизмы, как правило, HTTP. Эти сервисы построены вокруг бизнес-потребностей и разворачиваются независимо, используя полностью автоматизированную среду. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных. Понимание микросервисной архитектуры лучше всего воспринимать на примере сравнения монолитного и микросервисного приложения, монолитное приложение – приложение, построенное как единое целое. Приложения разработанные для предприятия, часто включают три основные части: пользовательский интерфейс (состоящий из HTML и javascript-а), база

данных (реляционной, со множеством таблиц) и сервер. Серверная часть обрабатывает HTTP запросы, выполняет доменную логику, запрашивает и обновляет данные в БД, заполняет HTML страницы, которые затем отправляются браузеру клиента. Любое изменение в системе приводит к пересборке и развертыванию новой версии серверной части приложения. Если говорить другим языком то один микросервис должен решать одну задачу, например сервис авторизации, который отвечает только за авторизацию пользователя на портале.

Монолитный сервер — довольно очевидный способ построения подобных систем. Вся логика запросов выполняется в единственном процессе, при этом вы можете использовать возможности вашего языка программирования для разделения приложения на классы, функции и namespace-ы. Вы можете запускать и тестировать приложение на машине разработчика и использовать стандартный процесс развертывания для проверки изменений перед выкладыванием их в продакшн. Практически любое изменение требует пересборки и публикации в продакшн новой версии всего приложения. Чем старше приложение, тем сложнее становится сохранить модульность и тем больше изменение одного модуля влияет на другие модули, заставляя менять их код. Именно эти неудобства привели к архитектурному решению – микросервисы. Построение приложения в виде набора сервисов. Кроме возможности независимого развертывания, каждый сервис также получает четкую физическую границу, которая дает возможность разным частям, т.е. сервисам, быть написанными на разных языках программирования. Микросервисы дают большие возможности к масштабируемости проектов, дают возможность разрабатывать отдельные проекты, но и тут есть свои недостатки:

Разработка распределенных систем вызывает сложности. Положительной стороной является распределенность модулей, но она же так же усложняет разработку , разрабатывая отдельный микросервис, нужно не забывать о том, что к нему обращаются другие сервисы и его разработка не должна задеть API, к которому обращаются сервисы.

У каждого сервиса есть своя база данных, а значит управление транзакциями, в определенный момент, станет вызывать все большие сложности.

Тестирование приложений будет занимать больше времени и станет достаточно громоздким, т.к. сервисов становится больше и каждый из них нужно запустить, что бы проверить контакт с базой данных, прежде чем запустить само тестирование.

Так же, публикация в продакш сервисов усложниться, т.к. в момент публикации одного сервиса может потребоваться остановка других, что бы они не выдавали ненужных ошибок пользователям.

Подводя итог исследования приходим к выводу, что Микросервисы не только облегчают разработку сервисов, своей независимостью друг от друга, но и в некоторых моментах обеспечивают усложнения этой самой

разработки. А так же, при проектировании необходимо помнить о том, что сервис должен отвечать только за выполнение одной задачи.

Библиографический список

1. Джамшиди П. и др. Микросервисы: пройденный путь и дальнейшие цели //Открытые системы. СУБД. 2018. №. 3. С. 17-17.
2. Опарин Г. А., Богданова В. Г., Пашинин А. А. Микросервисы как фундаментальная основа распределенного сборочного программирования //ИТНОУ: информационные технологии в науке, образовании и управлении. 2018. №. 2 (6).
3. Патент РФ № 2018616006, 21.05.2018 Программа синтеза моделей аппаратных ресурсов микросервисов// Патент России № 2018612982. 2018. / Жевнерчук Д. В., Рассадин О. С.
4. Патент РФ № 2018660682, 28.08.2018 программный комплекс микросервисов NCRAWLER// Патент России № 2018614166. 2018. / Поникаров М. В.
5. Стоянченко С. С., Полищук М. А. Повышение быстродействия взаимодействия микросервисов restful информационной системы медицинского ассистанса //Научные достижения и открытия современной молодёжи. 2018. С. 36.