А. В. Бортник

АЛГОРИТМИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ С РАЗЛИЧНОЙ АРХИТЕКТУРОЙ

Рассматриваются алгоритмы функционирования информационной системы с монолитной и микросервисной архитектурой применительно к задачам организации автоматизированных информационных систем на примере системы продажи проездных документов на железнодорожный транспорт.

Введение

При создании сложных автоматизированных информационных систем (приобретение проездных документов на различные виды транспорта, а также документов на посадочные места в концертные залы и стадионы и т.п.) важными этапами являются разработка структуры и алгоритмов функционирования таких систем, что влияет на эффективность, производительность, надежность и удобство эксплуатации применяемых аппаратный и программных средств. В настоящей работе исследуется информационная система приобретения проездных документов на железнодорожный транспорт.

В состав системы входит несколько модулей: регистрации пользователей; регистрации пассажиров пользователя; оформления проездных документов на поезда; оформления проездных документов на электрички; оплаты проездных документов; справочной информации.

При построении системы продажи проездных документов необходимо учитывать особенности монолитной и микросервисной архитектуры.

I. Монолитная система

Для системы с монолитной архитектурой характерно нахождение всех модулей системы в одном проекте с общей кодовой базой. Основным достоинством такой системы является удобство в работе: настройка одного модуля, пусть и большого, что проще и быстрее по сравнению с настройкой приложения с микросервисной архитектурой, включающей набор отдельно настраиваемых различных модулей. Основным недостатком в сравнении с микросервисной архитектурой является сложность распределения ресурсов между модулями, так как система с монолитной архитектурой содержит их внутри себя, что не позволяет распределять ресурсы между модулями. В ряде случаев это играет ключевой фактор при построении крупных систем и сильно влияет на отказоустойчивость.

Схема алгоритма функционирования системы продажи документов на железнодорожный транспорт с монолитной архитектурой изображена на рисунке 1.

Когда пользователь отправляет запрос, система через API-шлюз принимает его и обрабатывает соответствующими модулями. Если, например, пользователь оформляет проездной документ на поезд, то вся система обрабатывает его запрос, при этом выполняя функции, соответствующие модулю оформления проездных документов на поезда, а остальные модули в это время не работают ввиду отсутствия необходимости, и недоступны извне, так как система обрабатывает конкретный запрос. Конечно же модули взаимодействуют между собой, поэтому после оформления проездного документа, пользователь будет перенаправлен на оплату оформленного заказа, или, к примеру, для оформления заказа система предложит варианты транспорта по некоторым критериям поиска из модуля справочной информации. Здесь стоит отметить преимущество монолитной архитектуры в прямом доступе к остальным модулям. И в дальнейшем при обработке данных по оплате заказа система будет задействована, используя модуль оплаты проездных документов, и в это время остальные модули не используются и не доступны извне.

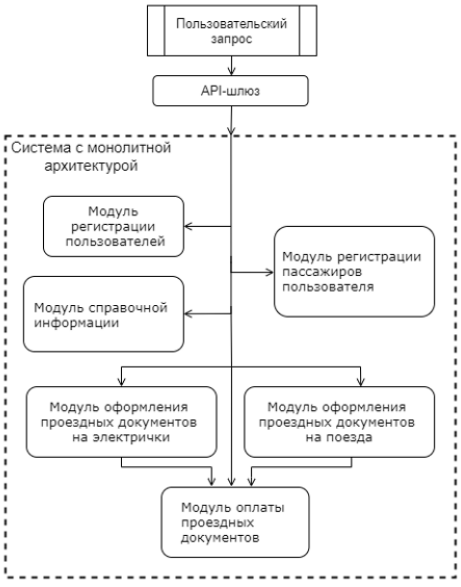

Рис. 1 – Схема алгоритма функционирования системы с микросервисной архитектурой

Управление ресурсами такой системы происходит следующим образом: что бы система могла обрабатывать несколько запросов одновременно, аппаратное обеспечение выполняет обработку данных в нескольких параллельных

процессах. Тогда для создания отдельного процесса для обработки запроса другого пользователя в отдельном потоке запускается копия системы. Если модуль оплаты заказов используется в два раза чаще каждого из модулей оформления проездных документов, потому что оплачивать необходимо оба вида проездных документов, то для поддержания стабильности работы ресурсы необходимо рассчитывать так, чтобы самый нагруженный модуль системы выдерживал нагрузку и обрабатывал стабильно все запросы пользователей.

II. Микросервисная система

Для системы с микросервисной архитектурой характерна самостоятельность каждого из модулей, так как фактически они являются отдельными приложениями. Это позволяет настраивать и распределять ресурсы для каждого модуля в отдельности. Основным и важнейшим преимуществом такой системы является увеличение отказоустойчивости, когда при недоступности или выходе из строя одного модуля все остальные продолжат работать в своем предопределённом настройкой режиме. Для больших систем с множеством различного функционала зачастую этот фактор может являться наиболее значимым. Схема алгоритма функционирования системы с микросервисной архитектурой изображена на рисунке 2.

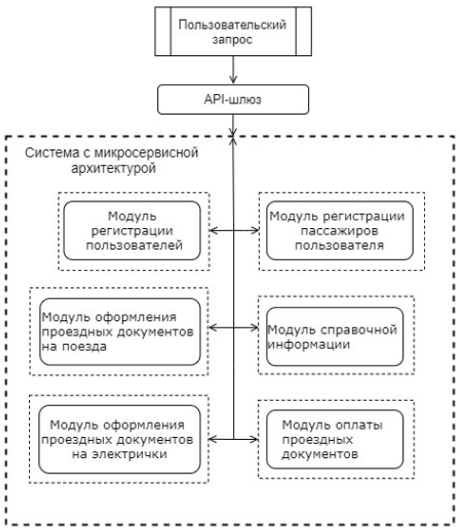

Рис. 2 – Схема алгоритма функционирования системы с монолитной архитектурой

Система принимает пользовательский запрос и через API-шлюз направляет его на обработку соответствующему модулю. Таким образом, если пользователь направляет запрос на оформление проездного документа на поезда, то его запрос поступит на соответствующий модуль, а все остальные модули системы будут доступны и могут обрабатывать запросы других пользователей. При этом модули могут взаимодействовать между собой, и не важно на одном ли физическом устройстве они находятся, поскольку взаимодействуют они как независимые приложения посредством сети Интернет.

Достоинством такой системы является возможность настройки каждого модуля. При ситуации, когда, к примеру, высоконагруженный модуль не справляется с потоком запросов, появляется возможность увеличения производительности такого модуля методом создания копии.

Копии могут находиться на различных аппаратных средствах. К примеру, в регионе, где интенсивность потока запросов выше, установить две копии модуля на более мощном сервере, а в регионе с меньшей интенсивностью – две копии на более слабом серверном оборудовании. В общем случае это уже особенности настройки, но показать ее гибкость важно, потому что это одно из ключевых преимуществ системы с микросервисной архитектурой.

III. Выводы

Иногда именно благодаря настройке принципа функционирования можно значительно увеличить эффективность системы, а микросервисную систему можно более гибко настраивать, так как каждый ее модуль – самостоятельный. Появляется возможность создать копию отдельного модуля, а не полностью всей системы. При этом увеличивается отказоустойчивость системы. Это становится важным фактором для крупных систем с большим набором различных функций, которые удобно разнести в отдельные модули.

Окончательные рекомендации по эффективности рассмотренных структур могут быть даны в результате моделирования системы с учетом конкретных исходных данных относительно размерности системы, интенсивности входящих информационных потоков и возможностей используемых ресурсов.

1. Крис Ричардсон. Микросервисы. Паттерны разработки и рефакторинга/ К.Ричардсон – Изд-во: Издательский дом “Питер”, Санкт-Петербург, 2016. – 544 с.
2. Сэм Ньюмэн. Создание микросервисов / С.Ньюмэн. – Изд-во: Издательский дом “Питер”, Санкт-Петербург 2016. – 304 с.
3. Интернет-ресурс https://habr.com/ru/

Бортник Артем Владимирович, магистрант кафедры информационных технологий автоматизированных систем, ttemmik97@gmail.com

Научный руководитель: Лукьянец Степан Валерьянович, кандидат технических наук, профессор, Lukyanetsstv@gmail.com