УДК 004.4’2

БИЗНЕС-ТЕЗИС "СТМАР": РЕАЛИЗАЦИЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ С ИСПОЛЬЗОВАНИЕМ ОБЛАЧНЫХ ТЕХНОЛОГИЙ ДЛЯ СЕРВИСА РЕШЕНИЯ ЗАДАЧ МАРКЕТИНГА

Бученков Е.Е. (Университет ИТМО)
Научный руководитель – доцент, кандидат военных наук Билятдинов К.З.
(Университет ИТМО)

В статье рассматривается сравнение и выбор архитектурной реализации современного клиент-серверного кросс-платформенного решения, а также выбор облачного сервиса под созданное решение в текущих реалиях. Как организовать струткуру Request-Response от клиента через сервер и до БД; как правильно, оптимально, эффективно и тестируемо организовать серверный код; какие подходы стоит применить на клиентской стороне; и какой облачный сервис выбрать в 2022 году, все эти вопросы освещаются в статье.

Введение.

Моё исследование, по сути, является полностью прикладным и происходит в момент создания нового программного продукта. В его рамках рассматриваются вопросы выбора той или иной архитектуры организации кода как на клиенте, так и на серверной части приложения, а также вопросы передачи данных между слоями приложения и выбор облачного сервиса для публикации программного продукта. Исследование основывается на множестве статей и книг по теме Software design/architecture. Как пример такого ресурса, труд спикера по части облачных систем и микросервисов Сэма Ньюмана (Sam Newman) – «Monolith to Microservices: Sustaining Productivity While Detangling the System» (2021, ISBN: 9781492047841). В данном ресурсе как раз и рассматривается затронутая в моём исследовании проблема выбора архитектуры продукта, что это может за собой повлечь и какие риски для бизнеса это несёт.

Основная часть.

В рамках исследования разрабатываемый программный продукт изначально имеет монолитную архитектуру, т. е. на серверной части продукта мы имеем единое ядро, в котором происходят все бизнес-процессы, валидации, мутации данных и прочее. Было замечено явное преимущество данного решения: его очень легко и быстро реализовать, ведь не нужно думать об окружении, коммуникации и связи разных частей продукта, когда у тебя единое ядро, совмещающее в себе абсолютно всё. Однако тут же есть и огромные минусы: отсутствие адекватного масштабирования, «слипание» функционала отдельных модулей, трудность тестирования, трудность сопровождения кодовой базы и прочее. В таком случае, если обратиться к книге «The Art of Scalability» за авторством Эддисон Уэсли (Addison, Wesley), необходимо рассмотреть понятие «куб масштабирования». По нему видно, что существуют три ортогональных способа увеличения производительности приложения: sharding, mirroring и microservices:

Sharding («разбиение») – расположение однотипных, но разных данных на разных узлах. Здесь используется метод интеллектуального выравнивания нагрузки приложения с помощью ключа шардирования, по которому данные распределяются по разным кускам приложения. Преимущества перед стандартным монолитным решением есть, но не такие значительные, чтобы перекрыть сложность реализации

Mirroring («зеркалирование») – горизонтальное дублирование или клонирование данных с размещением одинаковых хостов друг с другом. Данные хостом полностью копируются. Это в основном не производительное, а безопасное решение, гарантирующее ожидаемое время отклика системы.

Microservices («микросервисы») – декомпозиция или разбиение функциональности системы по бизнес-задачам. Каждый отдельный сервис выполняет определённые задачи и, по сути, имеет под собой собственное ядро. Идеальный вариант, однако он требует экспертности и

строгости в планах и структуре разработки при реализации. Оценить выбор микросервисного подхода можно посредством CAP-теоремы (Бюрер, 2000 г.), которая говорит о том, что система не может быть одновременно согласованной, доступной и разделяемой, что позволяет составить треугольную схему оценки подхода к разработке. Если говорить более конкретно про микросервисы, то также следует рассмотреть и SOA – сервисно-ориентированная архитектура, в которую входят микросервисы. По сути, микросервисы – наборы строгих правил и соглашений при разработке программного продукта, однако в данном случае сервисы являются: маленькими, сфокусированные, слабосвязанные и высокосогласованные. Для более подробной фокусировки на данном вопросе необходимо пояснить большое количество теории, поэтому можно пропустить этот вопрос. Также сервисы имеют целый ряд характеристик, которые помогают их оценить.

Вопрос об облаках стоит остро уже давно, однако сводится это к противостоянию продукта Microsoft – Azure, и продукта Amazon – AWS Cloud. Самая яркая разница – это стоимость, Azure примерно в 5 раз дешевле AWS, однако Azure более совместим с решениями на платформах и инструментах Microsoft (.NET и прочее), в то время как AWS одинаково совместим с любыми платформами. Оба решения поддерживают микросервисные реализации программных продуктов, поэтому здесь средством исследования выступает мониторинг производительности и доступность на той или иной платформе. Поскольку проект, в рамках которого проводится исследование, разрабатывается на платформе .NET, то и выбор в сторону Azure вполне очевиден.

Выводы. 1. Микросервисное решение является одним из лучших масштабируемых решений на данный момент, однако существует проблема реализации данного решения, так как оно требует экспертности разработчиков и следования строгим правилам разработки. 2. Облачные платформы сводятся к противостоянию Azure и AWS, в котором в среднем Azure выигрывает по стоимости, но проигрывает по поддержке платформ. Так как проект, в рамках которого проводится исследование, разрабатывается на платформе .NET, то и выбор сделан в сторону Azure.