УДК 004.72:339.15

Хандрик Юлия Андреевна,
магистрант,
кафедра анализа систем и принятия решений,
Институт экономики и управления,
ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина»
г. Екатеринбург, Российская Федерация

Иванцов Павел Сергеевич,
магистрант,
кафедра анализа систем и принятия решений,
Институт экономики и управления,
ФГАОУ ВО «Уральский федеральный университет имени первого Президента России Б.Н.Ельцина»
г. Екатеринбург, Российская Федерация

ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ МИКРОСЕРВИСОВ ПРИЛОЖЕНИЯ ПО КОНТРОЛЮ ЗА ЗДОРОВЫМ ОБРАЗОМ ЖИЗНИ

Аннотация:
В рамках работы будут рассмотрены ключевые подходы к построению архитектуры мобильных приложений, в разрезе применимости к высоконагруженному приложению, агрегирующему функции контроля за показателями здорового образа жизни. После чего будет проведен анализ выявленных решений и выбор наиболее подходящего.

Ключевые слова:
Мобильное приложение, архитектуры, информационные технологии, здоровый образ жизни, высоконагруженность, архитектура мобильного приложения.

При планировании разработки приложения – перед техническими специалистами всегда стоит задача на основании бизнес-потребности продумать и описать архитектурный подход к реализации системы. Подходы к построению архитектуры сильно отличаются при сравнении мобильных и WEB приложений. В рамках статьи будут рассматриваться именно подходы к созданию архитектуры мобильных приложений. Перед рассмотрением возможных подходов – необходимо понять ключевые особенности системы и бизнес-возможности и показатели, которые требуется достичь. Ключевые показатели, необходимые для выбора архитектуры:

- среда, в которой будет запускаться приложение (как было указано ранее, принципы создания архитектуры критически отличаются в зависимости от того, какое приложение предполагается реализовать, WEB, мобильное, десктопное или кроссплатформенное);
- существующие в компании сетевые ресурсы (необходимо определить, какие архитектурные подходы компания может обеспечить. Так, при отсутствии сервера достаточных мощностей, придется рассматривать аренду оборудования, что может кратно удорожить процесс дальнейшей эксплуатации);
- целевые показатели нагруженности системы, включая количество пользователей, ожидаемое время отклика на действие, пиковые показатели нагруженности (одной из ключевых характеристик системы являются эксплуатационные показатели. В зависимости от сферы бизнеса – показатели могут сильно разниться, так на B2C рынке за скоростью и безотказностью приложения следят обычно более пристально и выдвигают более строгие требования, чем в B2G сфере);
- функциональные требования к системе (при определении архитектуры – обязательно понимать, какая система будет реализовываться и какие функциональные возможности для пользователя будут в ней реализованы. Не зная требований к функциям, невозможно определить, какая архитектура будет наиболее валидна).

Рассмотрим сценарий, при котором создается мобильное приложение, в рамках которого будет осуществляться контроль за показателями здорового образа жизни. Тогда ключевые показатели (в существенно упрощённом виде) могут выглядеть следующим образом:

- мобильное приложение;
- сервера и прочее сетевое оборудование отсутствуют;
- среднее количество одновременно работающих пользователей – 10 000, пиковое количество одновременно работающих пользователей – 50 000, Время отклика – 1 сек. Время отклика при пиковой нагрузке – 2 сек.;

- пользователь должен иметь возможность записать в базу данных через интерфейс прием пищи, посредством выбора блюда из статичного справочника и указания объема блюда. Пользователь должен иметь возможность на главном экране ознакомиться с общим объемом потребленных за день калорий, а также потребленных активных веществ (БЖУ). Пользователь должен иметь возможность внести в базу данных через интерфейс список упражнений и ключевые значения (вес, количество подходов, количество повторов) посредством выбора значений из статического справочника. Пользователь должен иметь возможность ознакомиться с историческими данными о потребленных и потраченных калориях за период не более года в виде линейной диаграммы. Пользователь должен иметь возможность получать информацию о пройденных шагах из базы данных телефона или умных часов.

После определения ключевых показателей – можно начинать рассматривать возможные подходы к построению архитектуры.

Мы рассмотрим подход к реализации взаимодействия между элементами системы и архитектуру приложения. Способы организации кодовой базы внутри приложения описаны ниже.

Монолитная архитектура – означает неделимую систему, когда все компоненты приложения взаимосвязаны и объединены в одной программе на одной платформе [1].

Микросервисная архитектура – это несколько независимых однофункциональных протоколов или сервисов, которые собираются в приложение [2].

SOA – это набор архитектурных принципов, не зависящих от технологий и продуктов, совсем как полиморфизм или инкапсуляция [3].

В случае нашего приложения – наиболее подходящей будет микросервисная архитектура в связи с необходимостью выделения уникального клиентского слоя, слоя справочной информации, разделения функций по работе с тренировочным циклом, дневником питания и модулем статистики, базирующемся на OLAP кубе.

Для организации работы между микроервисами – необходимо определить способы обмена информацией между ними [4], существуют синхронные вызовы, асинхронные вызовы и API-шлюзы. Рассмотрим каждый из них подробнее.

Синхронные вызовы – один микросервис напрямую вызывает другой микросервис, ожидая ответа. Обычно это делается через сетевой протокол, такой как HTTP или gRPC. В таком случае вызывающий микросервис блокируется до получения ответа от вызываемого микросервиса.

Асинхронные вызовы – микросервисы могут взаимодействовать посредством использования сообщений или событий. Вместо непосредственного вызова, микросервисы публикуют сообщения или события в шину сообщений или брокер событий, и другие микросервисы, заинтересованные в этих сообщениях или событиях, могут подписываться на них и реагировать соответственно.

API-шлюзы – при наличии большого количества микросервисов в архитектуре целесообразно использовать API-шлюзы. API-шлюзы предоставляют единый точку входа для всех клиентов и агрегируют запросы от клиента на несколько микросервисов, управляя авторизацией, аутентификацией, маршрутизацией запросов и другими аспектами.

В нашем случае – необходимо установить гибридный способ, так как операции подразумевают как запрос пользователя на получение простых данных в интерфейсе (например, список блюд), для такой операции – асинхронный способ получения информации будет неудобен в связи с установлением очереди на выполнение запроса. Вместе с тем получение массивного блока информации о статистических показателях синхронно реализовать будет неправильно, так как время на обработку статистики может потребоваться немаленькое и блокировать функции системы в ходе ожидания ответа на конкретной странице так же затруднит работу пользователя и ухудшит пользовательский опыт.

Подводя итог, для реализации приложения, со схожими функциональными возможностями – стоит рассмотреть приложение с микросервисной архитектурой. В рамках которой отдельное компоненты приложения обмениваются сообщениями гибридным способом в зависимости от назначения конкретного сервиса. Такой подход позволит в дальнейшем обеспечить бесшовное масштабирование системы и реализацию качественного пользовательского опыта, при этом полноценную реализацию функциональных требований, предъявляемых к системе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Микросервисная и монолитная архитектуры: сравнение, плюсы и минусы // iq dev URL: https://iqdev.digital/blog/perehod-na-mikroservisnuyu-arhitekturu-perejti-nelzya-ostatsya (дата обращения: 13.11.2023).

2. Микросервисы или монолит - Какую архитектуру выбрать? // getit.academy URL: https://getit.academy/micromono (дата обращения: 13.11.2023).

3. Сервис-ориентированная архитектура (SOA) // habr URL: https://habr.com/ru/companies/vk/articles/342526/] (дата обращения: 13.11.2023).

4. Взаимоотношения между микросервисами // bigdevops URL: https://bigdevops.ru/article/vzaimootnosheniya-mezhdu-mikroservisami (дата обращения: 13.11.2023).

Khandrik Julia A.,
Master student,
Department of Systems Analysis and Decision Making,
Graduate School of Economics and Management,
Ural Federal University named after the first President of Russia B.N. Yeltsin
Yekaterinburg, Russian Federation

Ivantsov Pavel S.,
Master student,
Department of Systems Analysis and Decision Making,
Graduate School of Economics and Management,
Ural Federal University named after the first President of Russia B.N. Yeltsin
Yekaterinburg, Russian Federation

DESIGNING A MICROSERVICES ARCHITECTURE FOR A HEALTHY LIFESTYLE MONITORING APPLICATION

Abstract:
As part of the work, key approaches to building the architecture of mobile applications will be considered, in terms of applicability to a high-load application that aggregates functions for monitoring healthy lifestyle indicators. After which the identified solutions will be analyzed and the most suitable one will be selected.

Keywords:
Mobile application, architecture, information technology, healthy lifestyle, high load, mobile application architecture.