ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ УПРАВЛЕНИЯ И ОБМЕНА СООБЩЕНИЯМИ НА ОСНОВЕ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Е. С. Мойсейчик

fpm.moyseychES@bsu.by;
Научный руководитель – М. И. Давидовская, старший преподаватель

Характеристики программного обеспечения определяются еще на этапе проектирования. Целью работы является рассмотрение влияния архитектуры и решений в области организации хранения данных на расширяемость, масштабируемость и доступность распределенных систем на примере проектирования приложения сферы заботы о здоровье в микросервисной архитектуре. Практическим результатом является реализация приложения с веб- и мобильным клиентами для предоставления пользователям унифицированного доступа к метрикам здоровья.

Ключевые слова: архитектура ПО; асинхронность; микросервисы; масштабирование; хранилища данных; SQL; NoSQL; забота о здоровье; .NET.

Среди наиболее перспективных трендов в течение 5-10 лет известные аналитические и консалтинговые компании, такие как Gartner и IDC [1], выделяют сверх приложения, метавселенные и другие концепции, основанные на децентрализации, облачных технологиях и сочетании различной функциональности в рамках одной экосистемы. Рынок программного обеспечения (ПО) стремительно развивается в условиях высокой конкуренции и увеличения числа пользователей. Предъявляются новые требования к проектируемым решениям, касающиеся не функциональности самой системы, но ее качественных характеристик. Такие требования называют нефункциональными.

Под архитектурой ПО понимают определение основных структурных элементов программы, их интерфейсов и способов взаимодействия. Именно архитектура решения, закладываемая на раннем этапе создания продукта, определяет, насколько полно система сможет реализовать предъявленные к ней нефункциональные требования [2]. В связи с этим, актуальность исследования вопросов проектирования распределенных решений и выработки соответствующей теоретической базы очевидна.

Данная работа рассматривает две основные категории архитектур типа клиент-сервер: классические архитектуры, основанные на централизованных хранилищах, и событийно-ориентированные архитектуры, основанные на раздельном хранении и управляемые порождаемыми в системе событиями. В качестве представителей данных архитектур проводится сравнительный анализ монолитной и

микросервисной архитектуры, как наиболее распространенных решений. Для анализа предложена методология, использующая следующие критерии: расширение, масштабирование, доступность.

В ходе анализа выявлено, что:

- Монолитная архитектура имеет преимущества в небольших решениях, где необходимо обеспечить минимальные расходы на хранение, размещение и коммуникацию между компонентами приложения.
- Микросервисная архитектура имеет преимущества в сложных решениях, где имеет место использование различных технологий, разнородность данных, необходимость поддерживать высокие нагрузки и обспечивать повышенную доступность.

Для реализуемого приложения выбрано следование событийно-ориентированной парадигме, а именно микросервисной архитектуры в сочетании с протоколами REST и брокером сообщений RabbitMQ для организации синхронного и асинхронного взаимодействия.

Исходя из выбранного архитектурного подхода, в работе произведена декомпозиция предметной области на следующие поддомены, в соответствии методологией Э. Эванса [3]:

- Поддомен профиля пользователя – хранение и управление личными данными пользователя, а также базовыми метриками здоровья, такими как рост, вес, объемы и т.д.
- Поддомен планирования питания – хранение и управление данными о рекомендуемом ежедневном составе и калорийности питания, хранение данных о употребленной пище и подсчет текущего прогресса в достижении дневной цели.
- Поддомен рецептов – хранение и управление библиотекой рецептов, доступных пользователю.

В качестве асинхронных операций выделены подсчет рекомендации по питанию на основании цели и метрик пользователя, отправка уведомлений и журналирование действий в системе. Для решения проблем раздельного хранения данных в поддоменах выбрано использование шаблонов «Агрегат» и «Доменное событие» [4].

Также в ходе работы над проектированием хранения проведен сравнительных анализ хранилищ реляционного и нереляционного типа. Для сравнения использовалась методология, предлагаемая теоремой CAP. В ходе анализа выявлено [5], что:

- Реляционные базы данных предпочтительно использовать в случаях, где необходимы высокая согласованность, доступность и минимальный размер хранимых данных.

- Нереляционные решения предпочтительно использовать в случаях, когда необходимы высокая доступность и толерантность к разбиению, то есть при высоких нагрузках.

На основании полученных результатов выбраны соответствующие решения для каждого поддомена, спроектированны схемы баз данных. В качестве представителей реляционных решений выбрано использование Microsoft SQL Server, а в качестве документ-ориентированного хранилища – MongoDB. Для решения проблем высокой нагрузки использованы шаблоны «Разделение ответственности команд и запросов» и «Источник событий». Результат проектирования представлен на рис. 1.

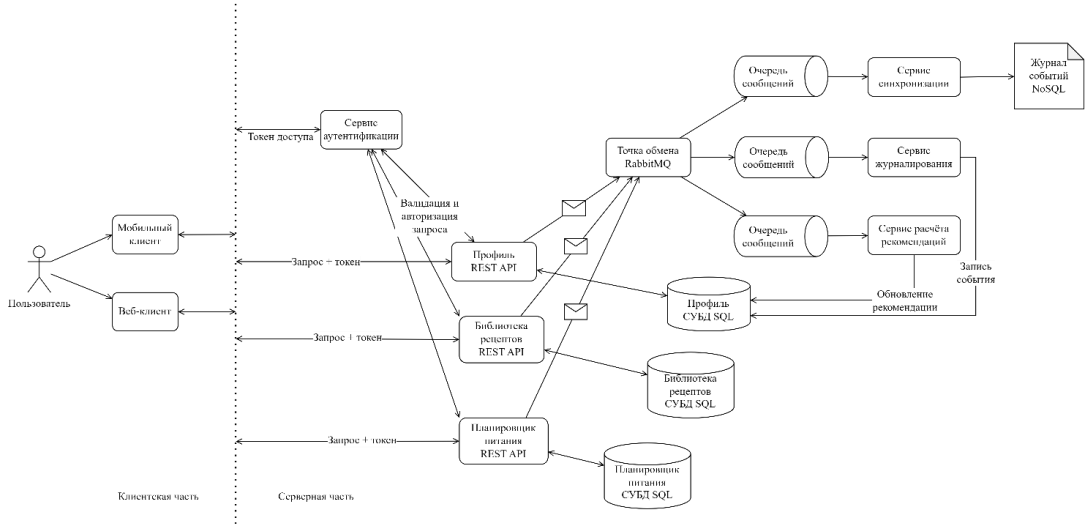

Рис.1. Схема модулей распределенного приложения

Для реализации API и сервисов серверной части приложения в работе использовался стек технологий .NET, например, такие решения как библиотека ASP .NET Core. Доступ к данным организован средствами ORM-решения Entity Framework Core. В качестве сервиса аутентификации и авторизации использовался Identity Server 4, поддерживающий протоколы аутентификации и авторизации OpenID и OAuth 2.0.

Для реализации веб- и мобильного клиента были выбраны библиотеки Angular и Flutter. Эти решения предоставляют гибкие возможности для проектирования одностраничных приложений, в том числе в соответствии с Material Design, и доступны для пользователей ПК через веб-браузер и пользователей смартфонов с ОС Android в качестве мобильного приложения. Пример экрана приложения изображен на рис. 2.

Приложение Planner спроектировано и реализовано в микросервисной архитектуре для хранения сведений о питании и

состоянии здоровья пользователей, планирования дневного рациона согласно рекомендациям и подбора рецептов.

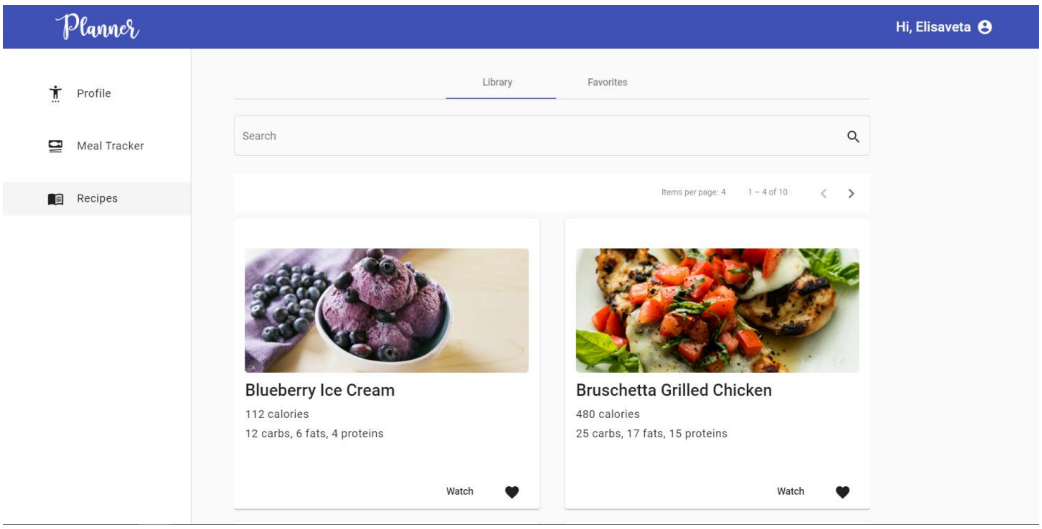

Рис. 2. Пример экрана итогового веб-приложения

В ходе работы над приложением изучены и решены такие задачи проектирования распределенных систем как декомпозиция предметной области, оптимизация хранения данных, задачи поддержания согласованности, масштабирования при возрастающей нагрузке, увеличение доступности системы. Итоговое решение может применяться в области бизнес-решений для заботы о здоровье.

Библиографические ссылки

1. What’s New in the 2022 Gartner Hype Cycle for Emerging Technologies [Electonic resource] / Gartner, 2022 - Mode of access: https://www.gartner.com/en/articles/what-s-new-in-the-2022-gartner-hype-cycle-for-emerging-technologies. Date of access: 28.03.2023.
2. *Мойсейчик, Е. С.* Проектирование систем, управляемых событиями / *Е. С. Мойсейчик, М. И. Давидовская* // Информационные системы и технологии [Электронный ресурс]: материалы междунар. науч. конгресса по информатике. В 3 ч. Ч. 3, Респ. Беларусь, Минск, 27–28 окт. 2022 г. / Белорус. гос. ун-т ; редкол.: С. В. Абламейко (гл. ред.) [и др.]. – Минск : БГУ, 2022. – 1 электрон. опт. диск (CD-ROM).
3. *Эванс Э.* Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем / *Э. Эванс* – Москва: 000 «И.Д. Вильямс», 2011. 448 с.
4. Ричардсон К. Микросервисы. Паттерны разработки и рефакторинга/ *К. Ричардсон* – СПб.: Питер, 2019. 544 с.
5. Мойсейчик, Е. С. Анализ решений для хранения и доступа к данным в распределенных системах / Е. С. Мойсейчик, науч. рук. М. И. Давидовская // Научные исследования XXI века. 28.02.2023. №1 (21) с. 55 – 59.