УДК 543.42:544.531:004.413

О КОНЦЕПЦИИ ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ ИСТОРИЧЕСКИХ ПАМЯТНИКОВ РЕСПУБЛИКИ БЕЛАРУСЬ

[1]Рудикова-Фронхёфер Л. В., [2]Жвалевский А. И., [3]Трус Ю. П.
[1]УО «Гродненский государственный университет имени Янки Купалы», Гродно, Беларусь, lada.rudikowa@gmail.com
[2]УО «Гродненский государственный университет имени Янки Купалы», Гродно, Беларусь, zhvaleuski.andrei@gmail.com
[3]УО «Гродненский государственный университет имени Янки Купалы», Гродно, Беларусь, yurik851@gmail.com

В статье описывается процесс проектирования хранилища данных для задач обработки информации об исторических памятниках Республики Беларусь. Отмечена важность правильной разработки и построения хранилища данных, а также предоставление доступа к его содержимому в задачах анализа данных.

Об особенностях предметной области. Разработка программного обеспечения в рамках предметной области, связанной с достопримечательностями, историческими памятниками, историческими событиями и историческими личностями Республики Беларусь, актуальна. Актуальность обусловлена поддержкой государством, а именно проведением 2018–2020 годов под знаком Года малой родины [1], способствованию развития туризма в стране. Соответственно информационно-аналитический портал (система) исторических памятников Республики Беларусь может решать задачу привлечения людей к тем или иным достопримечательностям, предоставляя единое информационное пространство, как для жителей Беларуси, так и для туристов.

Помимо предоставления исторической информации для пользователей, система должна иметь API для взаимодействия с другими системами / подсистемами, расширяю возможности интеграции и анализа, использую внешние сервисы.

Для решения данной задачи необходимо спроектировать общий план работы системы, взаимодействия её составных частей между собой, составить концептуальную модель базы данных и составить требования, предъявляемые к готовой системе.

В качестве источника данных будут выступать пользователи системы, которые будут заносить информацию, использую веб-интерфейс системы и средства для управления данными. Также рассматривается возможность получения данных из открытых источников данных наподобие Wikipedia.

Основные требования к системе. Для дальнейшего проектирования стоит определить выдвигаемые требования к системе.

Система будет построена используя микро сервисную архитектуру и технологии контейнеризации, позволяя масштабировать при необходимости, и изменять их независимо друг от друга.

Проектируемая система должна предоставлять пользовательский интерфейс в виде веб-приложения для обеспечения возможности ввода, модификации данных для администраторов сервиса. А также Web API для взаимодействия с подсистемами. С точки зрения пользователя, система обеспечивает его с исторической информацией (памятники, события, личности), в необходимом ему виде и порядке, например в хронологическом. В дополнение к этому система должна строить туристический маршрут на основе выбранных пользователем мест для посещения. И выступать своеобразным виртуальным гидом.

Используемая микро сервисная архитектура [3] предполагает наличие ряда сервисов, обеспечивающих целостную работу системы. Рассмотрим эти сервисы далее (рисунок 1).

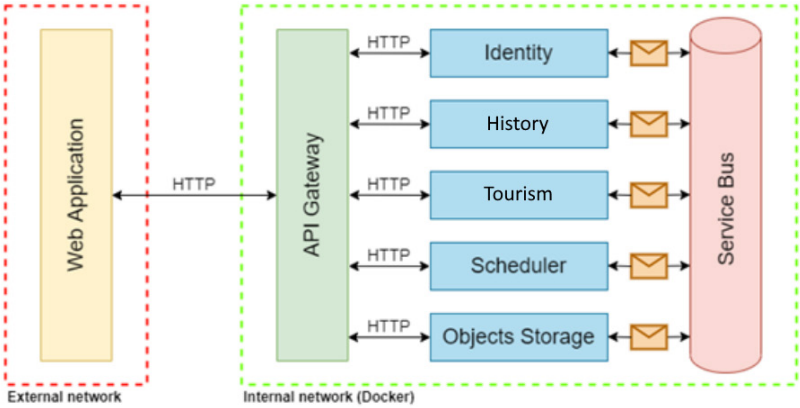

Рисунок 1 – Обзорная диаграмма архитектуры системы

Подсистема идентификации (Identity Service) отвечает за: хранение информации о пользователях системы; решает задачу регистрации и входа пользователей в систему, а также разграничение прав доступа; управление учетной записью; смена персональных данных (имени, фамилии и имени пользователя); смена данных для входа (пароль и почтовый адрес); смена изображения профиля. Помимо всего вышесказанного, основной (приоритетной) задачей, которую выполняет Identity Service, является генерация токена доступа (JWT) и токена обновления (refresh). Так как во всей системе авторизация построена на использования JWT-токена, хранящего информацию о пользователе, данной информации должно быть достаточно для установления того имеет ли конкретный пользователь права на выполнение той или иной задачи или нет.

Подсистема с исторической информацией (History). Сердце системы со всей информацией об исторических памятниках, событиях, личностях. Выступает историческим справочником для пользователей. Информация, предоставляемая этим сервисом, используется при составлении туров.

Подсистема для туризма (Tourism). Позволяет создавать туристические маршруты на основе выбранных для посещения достопримечательностей, с возможностями виртуального гида.

Планировщик задач (Scheduler Service). Используется для внутренних целей системы, чтобы инициировать определенные события по расписанию, например, удаление истекших токенов, отправку уведомлений, отправку интеграционных событий. При старте каждого из микросервисов, они отправляют планировщику задач сообщения (по шине сообщений), в которых указан интервал, тип сообщения и получатель. Задачей планировщика является обработка полученных сообщений и сохранение триггеров со вспомогательной информацией в БД. Когда проходит указанный в период, планировщик считывает данные из БД и отправляет заданное сообщение на заданный сервис. В случае с подсистемой идентификации, при старте она уведомляет планировщик задач о том, что он должен каждую минуту отправлять со-общение/команду на удаление истекших токенов из подсистему идентификации. В свою очередь система идентификации обрабатывает сообщение и начинает обработку команды по шаблону, описанному на рисунке 2 (поток обработки команды).

Хранилище объектов (Objects Storage). В системе активно используются фото и видео файлы. Для этих целей в экосистеме

приложения развернут отдельный сервис, специализирующийся сугубо на хранение объектов, и делает это очень хорошо. В качестве основы сервиса используется высокопроизводительный пакет объектного хранилища MinIO. Остальные сервисы хранят информации о расположении объекта, в нашем случае фотографии или видео, название корзины и файла и могут получать к ним доступ на чтение и запись. Плюсом использования подхода с хранилищем объектов является его совместимость с более мощными и производительными облачными решениями, такими как: Amazon S3 и Azure Storage.

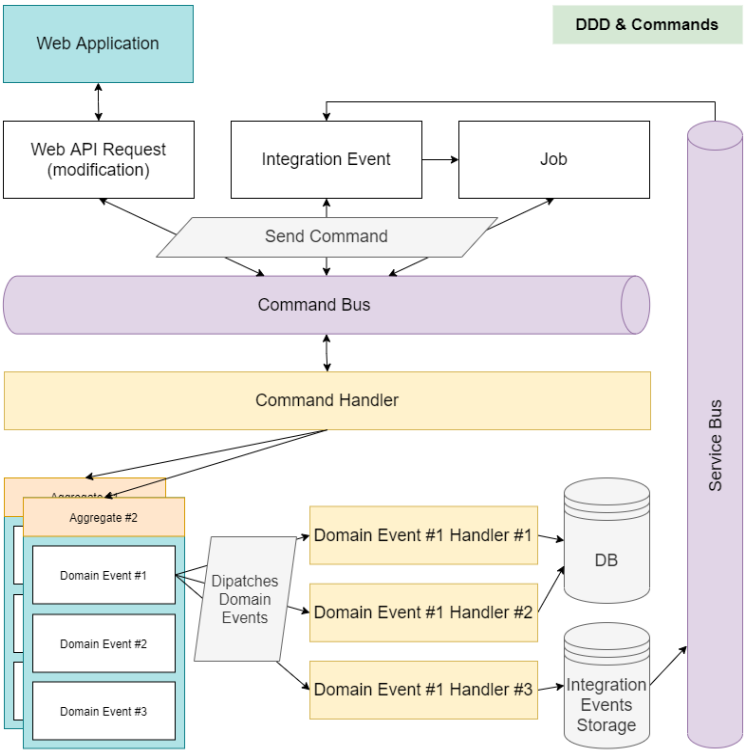

Рисунок 2 – Поток выполнения команд при сочетании предметно-ориентированного дизайна (DDD) и шаблон разделения ответственности команд и запросов (CQRS)

При построении большинства подсистем, основанных на платформе .NET, использовался предметно-ориентированного дизайна (Domain Driven Design, DDD) и шаблон разделения ответственности команд и запросов (CQRS) [2]. Такой подход позволяет наиболее правильно понять бизнес-процессы, выработать Ubiquitous Language (вездесущий язык), соответствующий предметной области и позволит системе гибко реагировать на изменения в требованиях, но в то же время он требует всесторонней первоначальной проработки бизнес-процессов и предполагает аналитику, потом проектирование и лишь затем – разработку системы. Все запросы к серверу разделены на команды (рисунок 2) и запросы (рисунок 3).

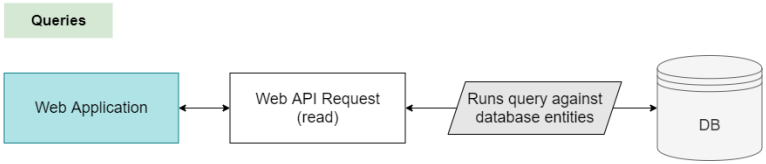

Рисунок 3 – Поток выполнения запросов при сочетании предметно-ориентированного дизайна (DDD) и шаблон разделения ответственности команд и запросов (CQRS)

Команда (например, «создать пользователя») всегда изменяет какую-то доменную сущность приложения. Послать команду может запрос к API ресурсу, интеграционное событие и запланированная задача. У каждой команды может быть один и только один обработчик команд, он может возвращать результат своего выполнение. При работе обработчика команды создаются доменные события (например, «пользователь создан») для определенных агрегатов, их количество не ограничено. В свою очередь каждое событие имеет своих обработчиков, которые могут записать данные в базу данных либо создать интеграционное событие, которое в последующем будет отправлено на шину событий. Количество обработчиков событий не ограничено.

Понятие запроса намного проще по сравнению с командой. При обработке «запросов» сервер не должен изменять хранящиеся данные, а только возвращать их в ответ.

Заключение. В статье изложены общие подходы, связанные с проектированием хранилища данных для задач обработки

информации об исторических памятниках Республики Беларусь. Представлено описание общего архитектурного стиля информационно-аналитической системы, а также приведена структура и характеристика потока выполнения команда в сочетании предметно-ориентированного дизайна и шаблона разделения ответственности команд и запросов.

СПИСОК ЛИТЕРАТУРЫ

1. Трилогия малой родины [Электронный ресурс] / Пресс-служба Президента Республики Беларусь – Режим доступа: [https://president.gov.by/ru/belarus/god-maloj-rodiny]. – Дата доступа: [15.11.2021].

2. Domain-Driven Design [Electronic resource] / Martin Fowler, 2021. – Mode of access: [https://martinfowler.com/tags/domain%20driven%20design.html]. – Date of access: [02.11.2021].

3. .NET Microservices: Architecture for Containerized .NET Applications [Electronic resource] / Microsoft Corporation, 2021. – Mode of access: [https://docs.microsoft.com/en-us/dotnet/architecture/microservices/]. – Date of access: [10.11.2021].