УДК 004.415.25

РАЗРАБОТКА ВСТРАИВАЕМОГО МОДУЛЯ ГАРАНТИРОВАННОЙ ДОСТАВКИ СООБЩЕНИЙ В БРОКЕРЕ RABBITMQ

Натробина О.В.

кандидат экономических наук, доцент, доцент кафедры экономики

Калужский государственный университет им. К.Э.Циолковского

Калуга, Россия

Пашечко В.Ю.

магистрант

Калужский государственный университет им. К.Э.Циолковского

Калуга, Россия

Аннотация.

В статье рассматривается процесс разработки модуля гарантированной доставки сообщений в брокере RabbitMq для встраивания его в веб-приложения, написанные на .NET Core с использованием микросервисной архитектуры. В процессе работы была выделена модель поведения пользователя, функциональная модель и разработана структура приложения. Также были созданы основные макеты пользовательского интерфейса, с разбиением на компоненты-контейнеры и компоненты-представления. Указаны такие важные блоки, как настройка модуля и формат передаваемых данных. Также в этой части описывается процесс разработки документации к модулю и публикации в пакетном менеджере NuGet.

Ключевые слова: веб-приложения, брокер сообщений RabbitMq, пользовательский интерфейс, хранилище данных Redis, формат передачи данных JSON.

GUARANTEED DELIVERY OF MESSAGES IN RABBITMQ BROKER

Natrobina O.V.

Candidate of Economic Sciences, Associate Professor, Associate Professor of the Department of Economics
Kaluga State University named after K.E. Tsiolkovsky
Kaluga, Russia
Pashechko V.U.
Undergraduate Student,
Kaluga State University named after K.E. Tsiolkovsky,
Kaluga, Russia

Annotation.
The article discusses the process of developing a guaranteed message delivery module in the RabbitMq broker for embedding it in web applications written in .NET Core using a microservice architecture. In the process of work, a user behavior model, a functional model were identified, and the application structure was developed. Also, the main layouts of the user interface were created, divided into container components and view components. Such important blocks as module settings and the format of transmitted data are indicated. This part also describes the process of developing documentation for a module and publishing it to the NuGet package manager.
Keywords: web applications, RabbitMq message broker, user interface, Redis data storage, JSON data transfer format.

На сегодняшний день проблема утери данных во время передачи их между сервисами в приложении является важным аспектом в высоконагруженных приложениях. В современной разработке мало времени уделяют безопасности передачи данных между сервисами, а ведь в некоторых приложениях потеря одного сообщения может играть немаловажную роль в дальнейшем функционировании приложения.

Современные технологии позволяют значительно снизить вероятность потери информации, стоит только грамотно подойти к этому вопросу.

Разработка подобных модулей рассматривается представителями бизнеса как лишние трудозатраты, не приносящие практической пользы в конечный продукт.

Разрабатываемый модуль предназначен для внедрения в веб-приложения, написанные на .NET Core, которые построены на микросервисной архитектуре и требует повышения уровня гарантии при доставке сообщений между сервисами.

Создание подобного модуля обусловлено отсутствием подобных аналогов в одном из крупнейших во всем мире пакетных менеджеров nuget. При разработке приложение с использованием брокера сообщений RabbitMq было обнаружено, что в пакетном менеджере nuget отсутствуют аналоги, решающие конкретно эту задачу, при этом модифицировать эти модули выходило бы значительно дороже.

В современной backend-разработке потеря времени на создание таких крупных модулей недопустима. Модульный подход к разработке значительно ускоряет процесс создания приложения. Программисту не нужно знать о внутренней реализации модуля, ему лишь нужно импортировать его в свой проект и настроить необходимую конфигурацию.

Таким образом встраиваемый модуль гарантированной доставки со стандартным функционалом ощутимо сократит временные затраты на разработку, а также будет учитывать уязвимые места в логике, которые может допустить программист при поверхностном изучении предметной области.

Для разработки был выбран фреймворк языка C# .Net Core в архитектуре Web Api, представляющий из себя кросс-платформенное приложение. [1]

После разработки, модуль планируется выложить в общедоступный центр для размещения пакетов NuGet. Выбор NuGet обусловлен тем, что для .NET Core на данный момент нет аналогов для подключения пакетов.

Для формализации и описания бизнес-процессов мы воспользовались IDEF0 методологией моделирования. Будем считать, что есть один основной процесс, выполняемый в системе – отправка и контроль доставки сообщений. [2] На рисунке 1 представлена IDEF0-диаграмма этих процессов.

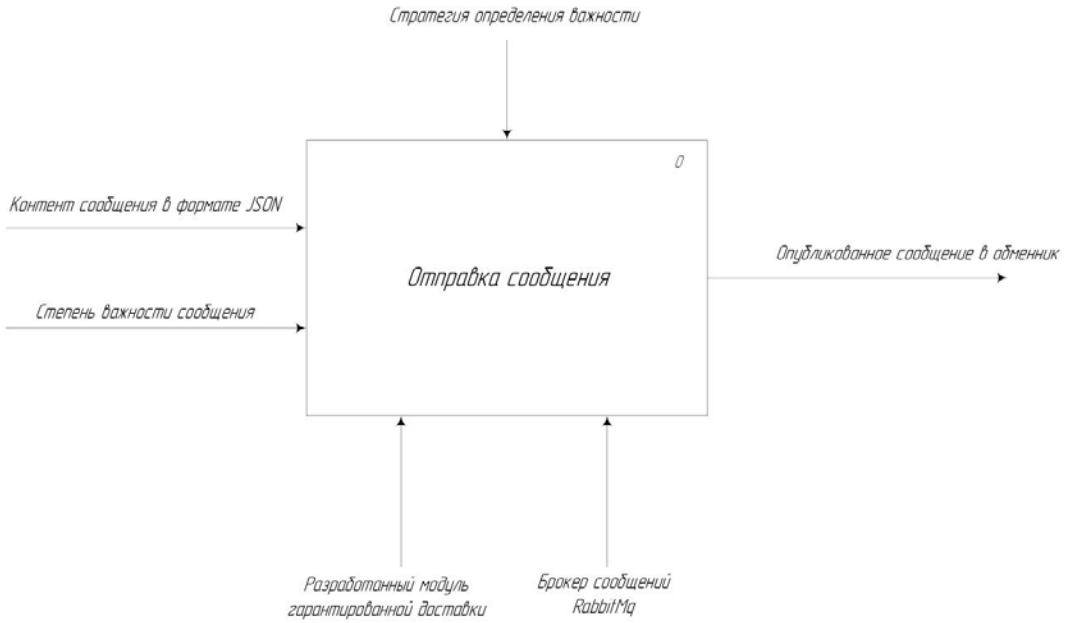

Рисунок 1 – IDEF0-диаграмма бизнес-процессов (разработано автором в программе Компас-3Д)

При исследовании предметной области было принято решение разработать гибкий для настройки модуль гарантированной доставки сообщений. Такой вариант конфигурации позволяет с большей вероятность привлечь пользователей к данному модулю.

RabbitMQ – это мультипротокольный брокер сообщений, позволяющий организовать отказоустойчивый кластер с полной репликацией данных на несколько узлов, где каждый узел может обслуживать запросы на чтение и запись. [3]

Очередь хранит и отдаёт получателям все поступающие сообщения. Обменник занимается маршрутизацией сообщений (не хранит их) на основе созданных связей между ним и очередями (или другими обменниками).

Посмотреть состояние брокера можно в веб интерфейсе, где будут наглядно показаны параметры конфигурации очередей и обменников сообщений, такие как количество потребителей сообщений из очереди, количество необработанных сообщений, привязка к конкретному обменнику со своим ключом или количество привязанных очередей к обменнику.

Для нашей работы был выбран вариант минимум однократной доставки сообщений. Такой вариант гарантирует, что сообщение к потребителю было доставлено, а контроль за дублированием сообщений осуществляется на потребителе сообщений.

Так как модулю есть необходимость хранить копию сообщения до момента получения подтверждения от получателя, в нашей работе мы будем использовать NoSql хранилище данных Redis.

Redis - хранилище данных типа «ключ» - «значение». [3] Благодаря тому что, Redis хранит данные в оперативной памяти, работа с ними осуществляется предельно быстро, что в сочетании с поддержкой различных типов данных, возможностью сохранения данных на диск и рядом других особенностей, сделало Redis если не самым популярным решением для реализации кеша в сложных высоко нагруженных приложениях, то во всяком случае одним из таковых.

В приложении «ключом» будет выступать уникальный идентификатор сообщения, по которому будет определяться к какому сообщению пришло уведомление от получателя. «Значение» - json-сообщение, которое будет публиковаться в обменник.

Исследуя предметную область, было принято решение использовать стандартный для брокера RabbitMq формат передачи данных – JSON.

JSON или JavaScript Object Notation — это формат, реализующий неструктурированное текстовое представление структурированных данных, основанное на принципе пар ключ-значение и упорядоченных списках. Хотя JSON начал свое распространение с JavaScript, он поддерживается в

большинстве языков, либо изначально, либо с помощью специальных библиотек. Обычно Json используется для обмена информацией между веб-клиентами и веб-сервером.

На рисунке 2 показана схема взаимодействия с отправителя с брокером сообщений.

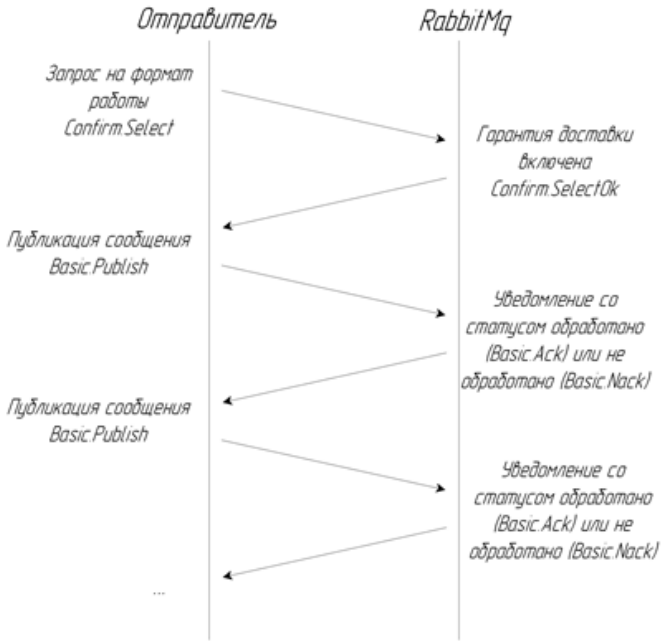

Рисунок 2 – Схема взаимодействия отправителя с брокером (выполнено автором в программе Компас-3Д)

Результаты работы модуля представлены в таблице 1 и таблице 2.

Таблица 1 – Работа брокера без модуля повышения гарантии

	Отправлено, шт.	Получено, шт.	Потеряно, шт.	Скорость, шт/мин
Стабильная работа RabbitMq	10 000	10 000	0	2497
Временная потеря соединения с RabbitMq	10 000	9 860	140	2121

Таблица 2 – Работа брокера с модулем повышения гарантии

	Отправлено, шт.	Получено, шт.	Потеряно, шт.	Скорость, шт/мин
Стабильная работа RabbitMq	10 000	10 000	0	1 468
Временная потеря соединения с RabbitMq	10 000	10 160	140	1 124

После того, как модуль будет готов, его необходимо опубликовать в общедоступный центр для размещения пакетов NuGet. Он используется для получения модулей из облачного сервера NuGet, либо для выгрузки их на эти сервера.

Пакет NuGet содержит манифест (файл .nuspec) с соответствующими метаданными, такими как уникальный ключ пакета, код версии, описание и другое. Некоторые свойства могут быть получены напрямую из параметров проекта, в результате чего пропадает необходимость редактировать их как в проекте, так и в манифесте модуля.

Получив библиотеку DLL и заполнив свойства проекта, необходимо использовать команду nuget spec, чтобы сгенерировался начальный файл .nuspec из проекта. Этот шаг включает в себя необходимые ключи замены для получения сведений из файла проекта.

Получив файл .nupkg, следует опубликовать его на сайте nuget.org, используя nuget.exe, также ключ API, полученный на этом сайте, для прохождения идентификации модуля. Для nuget.org нужно использовать самые последние версии nuget.exe для избежания проблем с размещением и корректной работой модуля.

Чтобы другие разработчики смогли разобраться, как устанавливать и настраивать модуль, необходимо создать и заполнить документ пользовательской документации REDME.md.

Заранее удостоверившись в работоспособности модуля опубликовать командой nuget push. При успешном размещении, его можно будет увидеть в общем каталоге модулей, либо по форме поиска на сайте https://www.nuget.org/.

Библиографический список:

1. Кан, М. Основы программирования на C#. — 2-е изд. — Москва: ИНТУИТ, 2019. — 167 с. — Текст: электронный// Лань: электронно-библиотечная система. — URL: https://e.lanbook.com/book/100320

2. Миндалёв, И. В. Моделирование бизнес-процессов с помощью IDEF0, DFD, BPMN за 7 дней : учебное пособие / И. В. Миндалёв. — Красноярск : КрасГАУ, 2018. — 123 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/103833 (дата обращения: 08.04.2020). — Режим доступа: для авториз. пользователей.

3. Рудинский И.Д. Технология проектирования автоматизированных систем обработки информации и управления. — М.: Горячая линия-Телеком, 2018. — 304 с. — URL: http://e.lanbook.com/book/5191

Оригинальность 76%