Министерство образования Республики Беларусь

Учреждение образования

"Белорусский Государственный университет информатики

и радиоэлектроники"

Лабораторная работа

по учебной дисциплине “Проектирование программных систем обработки больших объемов информации”

|  |  |
| --- | --- |
| Выполнил:  Проверил: | Студент гр. 956241 Дубовик Н.О.  Доцент, к.т.н Стержанов М.В. |

Минск 2021

СОДЕРЖАНИЕ

[1. Параллельная обработка данных 3](#_Toc67312567)

[2. Интеграционная шина для Big-data на базе Service Bus 6](#_Toc67312568)

[3. Хранение данных проекта в NoSQL(CosmosDB) БД 7](#_Toc67312569)

[4. Разворачивание полученного приложения в Azure. Реализация симуляции поступления данных в систему из разных источников/приложений 9](#_Toc67312570)

[5. Анализ соцсетей. Обоснование выбора решения для построения платформы обработки больших данных 12](#_Toc67312571)

[Вывод 14](#_Toc67312572)

# **1. Параллельная обработка данных**

Основными средствами Azure для работы с проектами типа PaaS являются Web App и Azure Functions. Первое средство представляет из себя облегченную версию виртуальной машины, в которой пользователю не надо проводить дополнительный настройки системы, вместо этого пользователю необходимо лишь выбрать тип операционной системы и мощность машины при создании сервиса, всю остальную работу на себе берет Azure. Azure Functions и вовсе представляют из себя бессерверное решение, в котором пользователю надо выбрать лишь среду выполнения.

Одной из главных особенностей данных сервисов является то, что у них есть возможность скаллировать количество рабочих машин (Instants) в зависимости от текущей или предполагаемой нагрузки. Скаллирование можно производить интенсивным (Up) и экстенсивным (Out) путем. За счет экстенсивного скаллирования можно добиться того, что одинаковые запросы будут выполняться на различных машинах, при этом будет соблюдена консистенция данных, то есть конечный пользователь не заметит, если его в один момент перекинет на новую машину.

На рисунке 1 представлен вариант интенсивного скаллирования с базового уровня (S1) до премиумного уровня (P3V3)

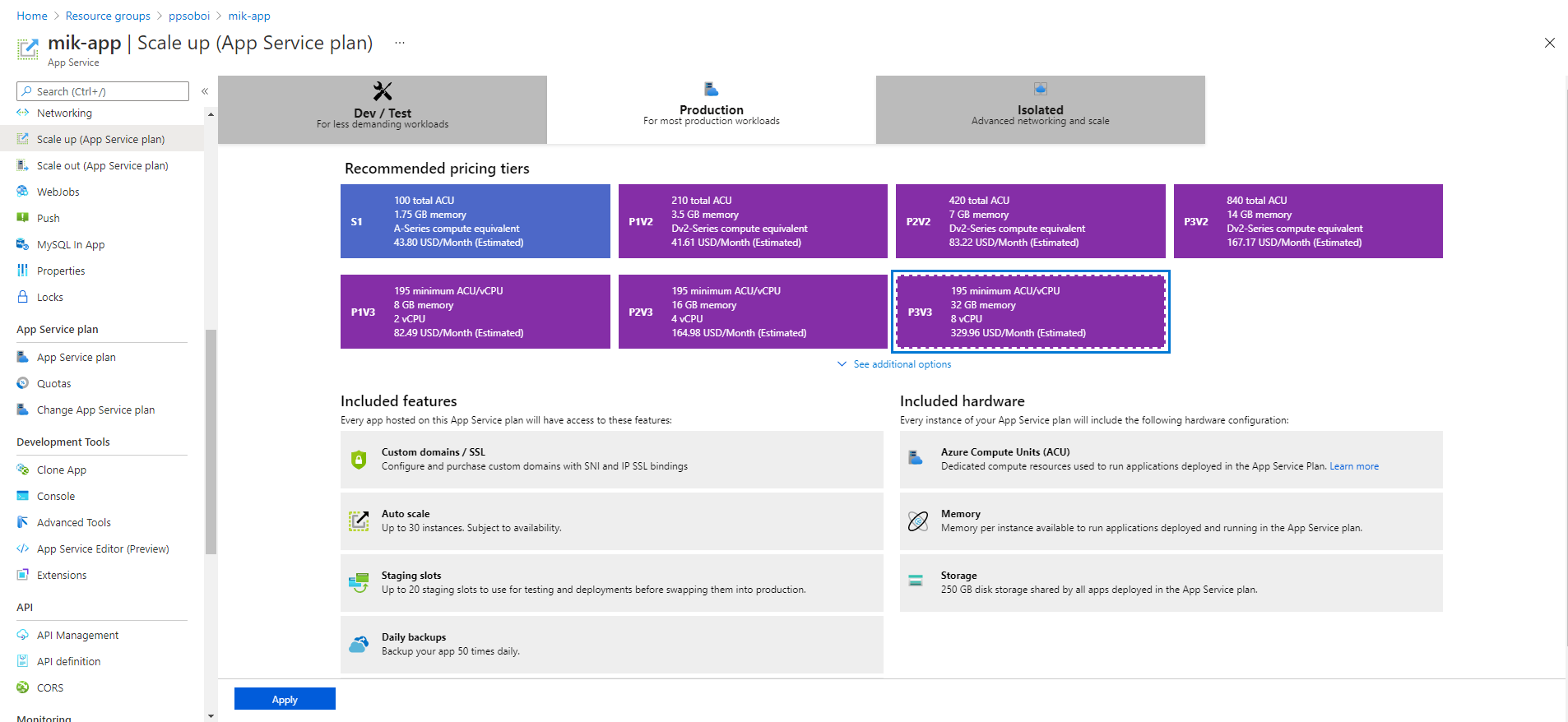


Рисунок 1 – Вариант интенсивного скаллирования для Web App

На рисунке 2 и 3 представлены варианты экстенсивного расширения. В первом случае количество машин задается вручную, во втором же создается специальное правила, по которому определяется необходимое количество машин. Количество машин, на которое можно проскаллировать созданный Web App зависит от выбранного типа подписки, чем он выше, тем большее количество машин можно использовать.

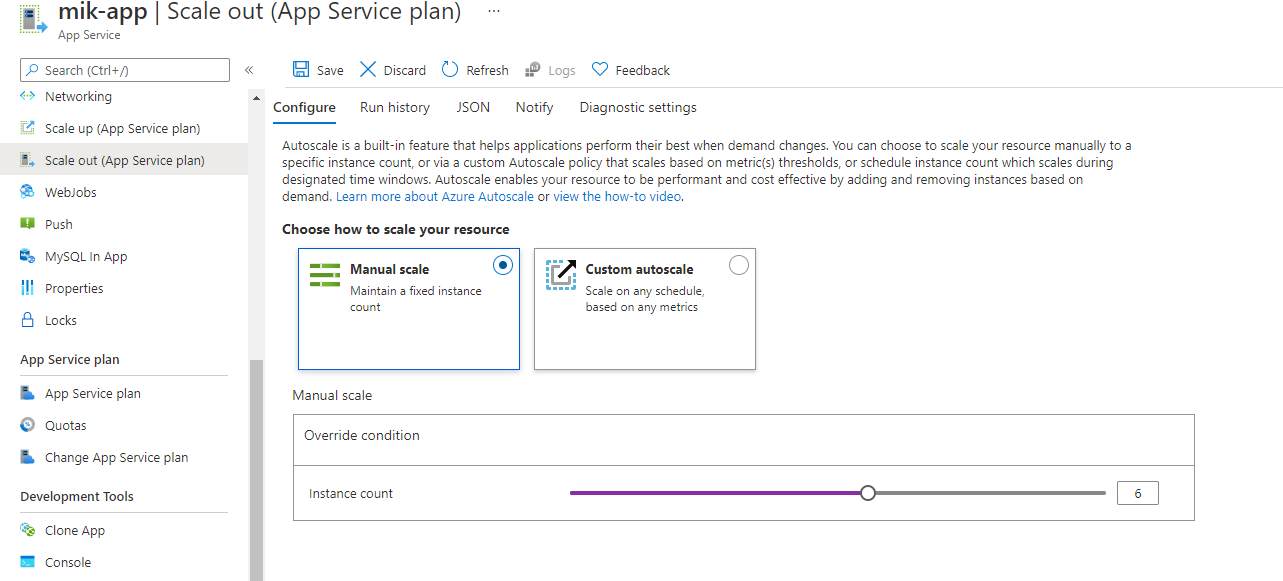


Рисунок 2 – Вариант мануального экстенсивного скаллирования для Web App

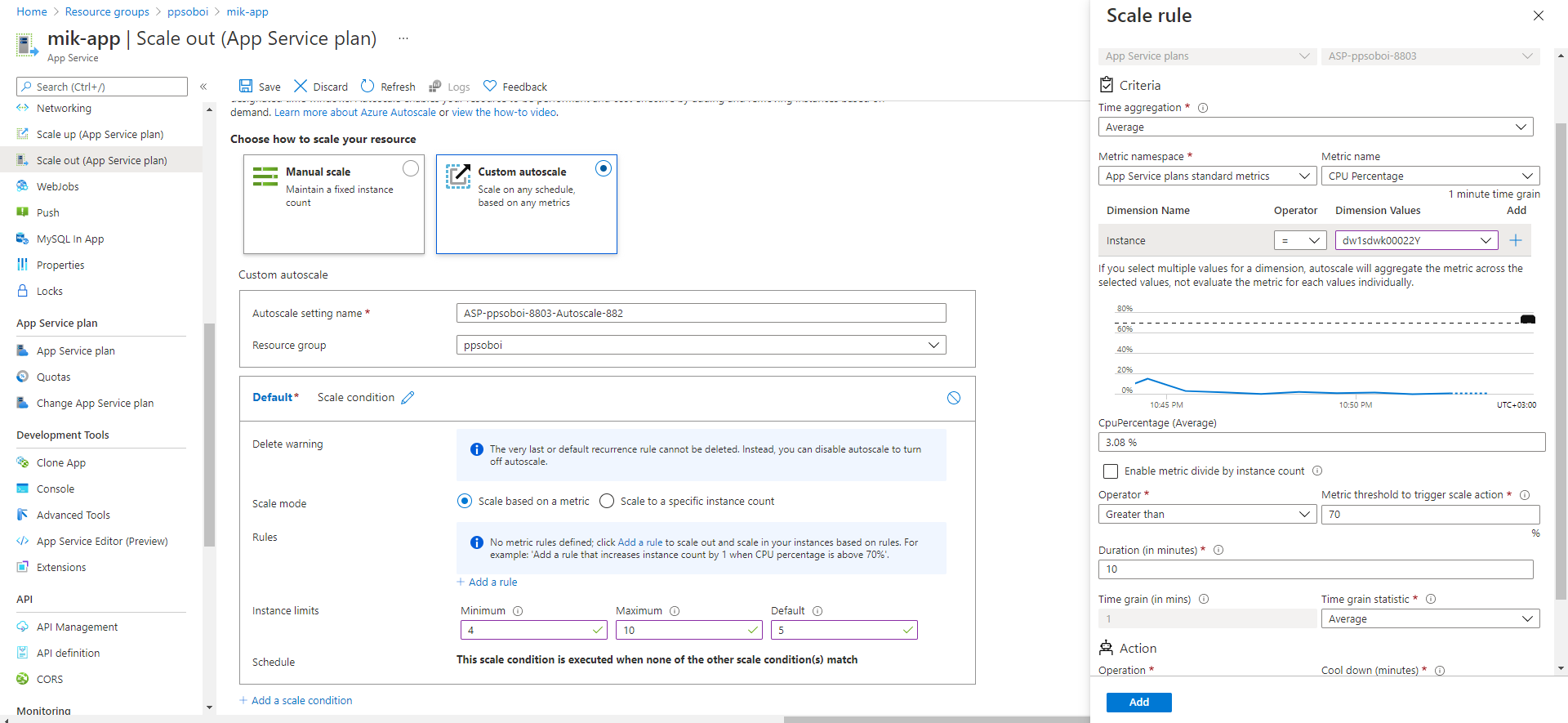


Рисунок 3 – Вариант автоматического экстенсивного скаллирования для Web App

Так как в Azure Function нет различных видов подписок, а вместо этого существуют два вида использования данного сервиса, по затраченному времени на вычисление и по времени использования, то для них возможен вариант только экстенсивного расширения, рисунок 4.

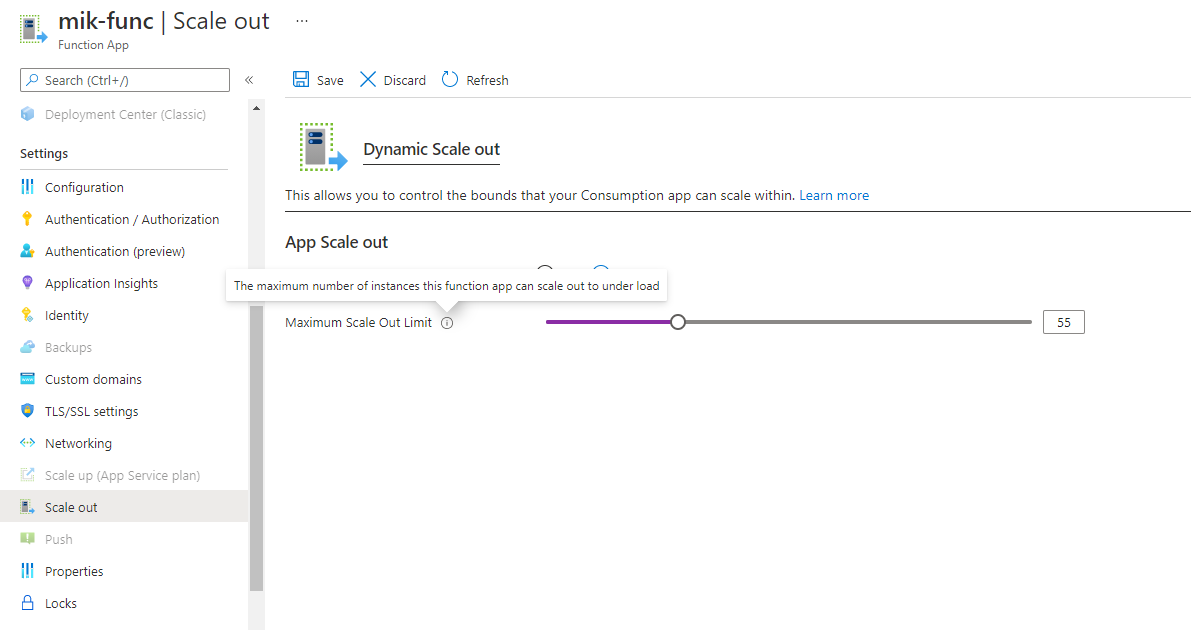


Рисунок 4 – Вариант мануального экстенсивного скаллирования для Azure Function

В ходе лабораторной работы были использованы как Azure Functions, так и Web App для параллельной обработки больших объемов данных. Внутри данных сервисов были реализованы .NET проекты.

Был создан Web App, представляющий из себя социальную сеть доступную для конечного пользователя. В данной социальной сети пользователи делятся друг с другом фотографиями через посты. Созданный Web App получает отправленный от пользователя пост и передает его через Service Bus далее для обработки, рисунок 5.

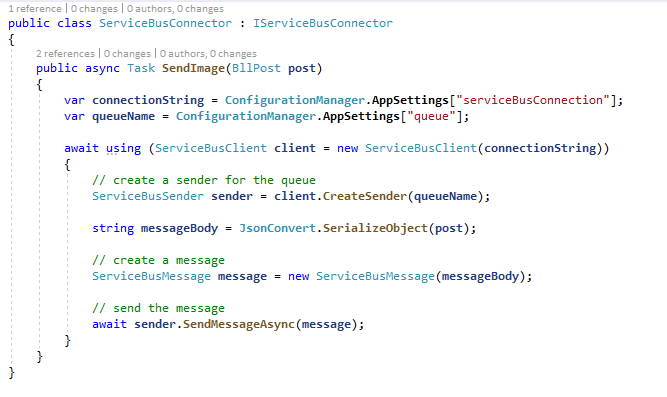


Рисунок 5 – Отправка данных в шину из Web App

# **2. Интеграционная шина для Big-data на базе Service Bus**

Для интеграции данных между различными компонентами приложения был использован Azure Service Bus, рисунок 6. Данный сервис представляет из себя шину, которая может принимать данные из множества источников и передавать их дальше.

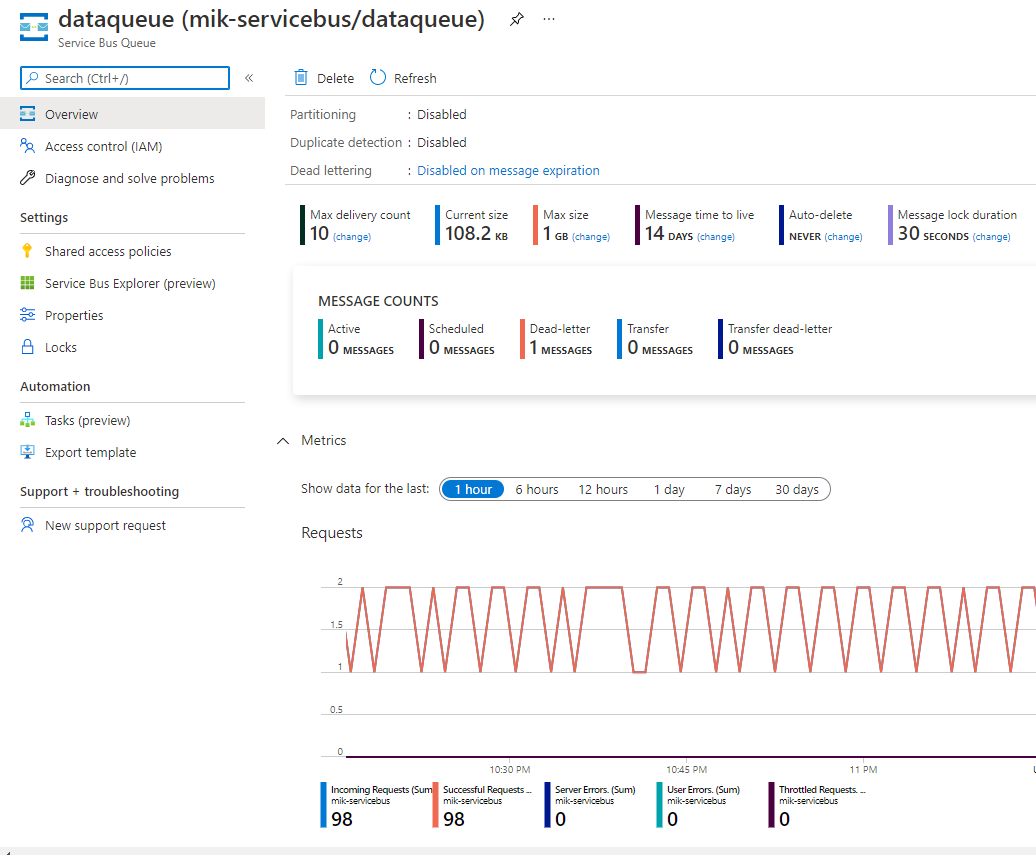


Рисунок 6 – Созданный Azure Service Bus

Данные передаются между приложениями и (или) службами с помощью сообщений. Сообщением называется контейнер с данными, дополненный метаданными. В нем могут содержаться любые данные, включая структурированные данные, закодированные в любом из распространенных форматов (например, JSON, XML, обычный текст).

В созданную шину поступают посты, который представляют из себя сериализованные JSON объекты. После попадания в шину и при наличии доступных получателей, данные отправляются дальше. Если же возможности передать данные дальше нет, то они остаются в шине до тех пор, пока кто-то не сможет их принять или не пройдет определенное количество времени, после которого данные будут автоматически удалены.

# **3. Хранение данных проекта в NoSQL(CosmosDB) БД**

Для хранения постов используется база данных CosmosDB, представляющая из себя NoSQL базу нового поколения. Данная БД обладает следующими преимуществами по сравнению со своими конкурентами:

1. обеспечение непрерывности бизнес-процессов, 99,999 % доступности и безопасность на уровне предприятия для каждого приложения;
2. легкое распределение данных в любой регион Azure с автоматической репликацией данных;
3. полностью управляемая служба баз данных. Полностью автоматизированное обслуживание, исправление и обновление для экономии разработчикам времени и денег;
4. одинаковая задержка не более чем 3 мс на чтение и 10 мс на запись, как при малой нагрузке, так и при нагрузке огромными данными.

После получения данных из шины в Azure Functions происходит предобработка поста пользователя, добавляются необходимые поля для последующей монетизации контента, а также рекалькуляция полей для последующей настройки контекстной рекламы, рисунок 6. Таким образом, сущность поста содержит в себе следующие элементы:

* name, содержит название поста;
* description, содержит описание поста;
* numberOfLikes, показывает количество отметок “Нравится” на посте;
* tags, в котором содержатся все добавленные к посту теги.

Помимо этих элементов, которые есть во всех постах, в рекламных постах содержатся следующие свойства:

* ageId, хранит информацию о том, каким возрастным группам показывать пост;
* countryId, говорит о том, в каком стране необходимо показывать этот пост;
* sexId, определяет кто является целевой группой – мужчины или женщины;
* languageId, показывает какая языковая группа предпочтительна для этого поста.

Помимо этих полей за счет выбора NoSQL базы данных в дальнейшем есть возможность расширять признаки, по которым будет показываться реклама. За счет гибкости CosmosDB в дальнейшем можно, например, добавить группы картинок в постах без потери существующей информации.  
 Для того, чтобы созданные и обработанные данные хранились в Cosmos, внутри нее была создана база данных Post, внутри которой находится контейнер с данными о постах, PostContainer.

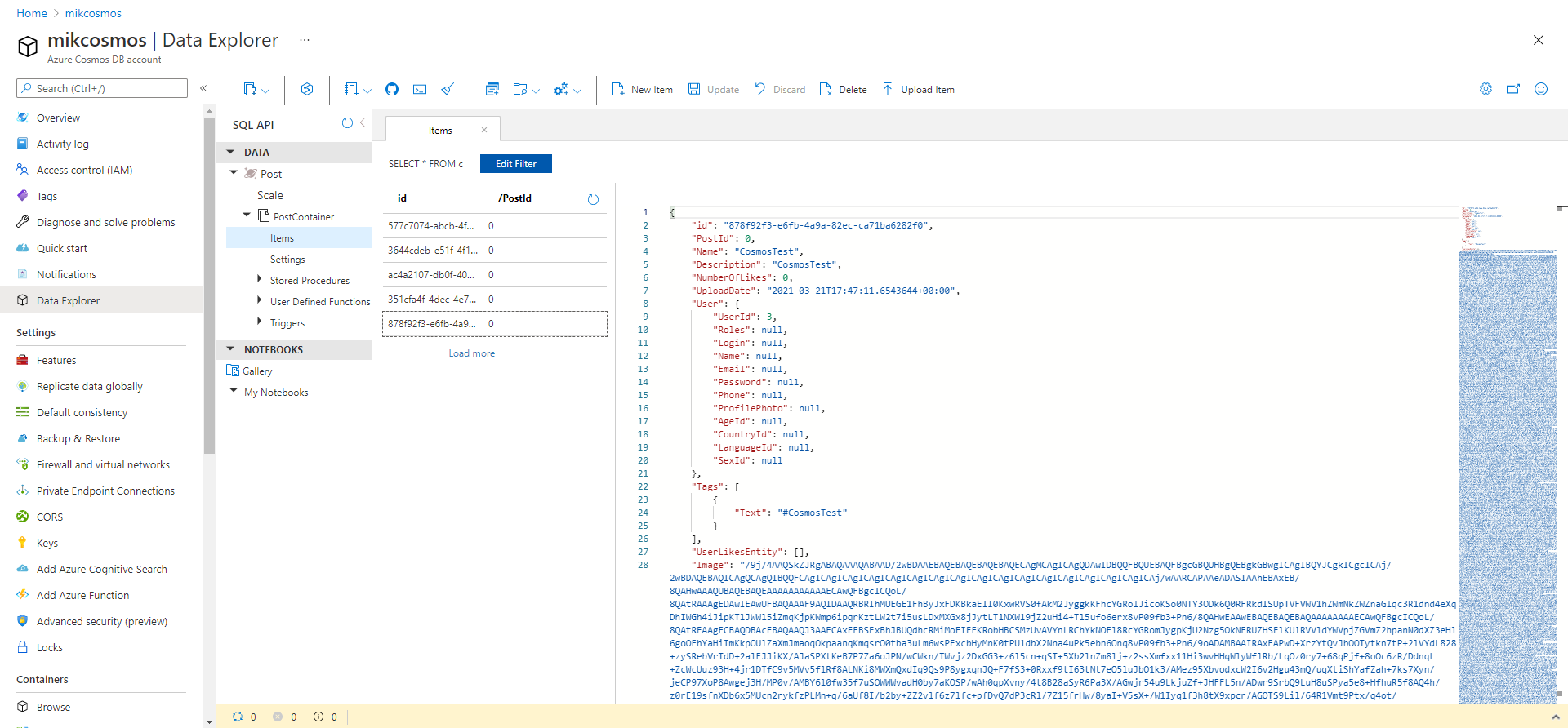


Рисунок 6 – Использование CosmosDB в приложении

В CosmosDB каждый контейнер имеет политику индексирования, определяющую, как должны индексироваться элементы контейнера. Политика индексирования по умолчанию, задаваемая для только что созданных контейнеров, индексирует каждое свойство каждого элемента и применяет диапазонные индексы для любых строк или чисел. Это позволяет получить хорошую производительность запросов без необходимости думать об индексировании и управлении индексами. Как видно из рисунка 7 в созданном контейнере используются как раз стандартная политика индексирования, обозначаемая словом “consistent”, а также спецсимволами в поле “includedPaths”.

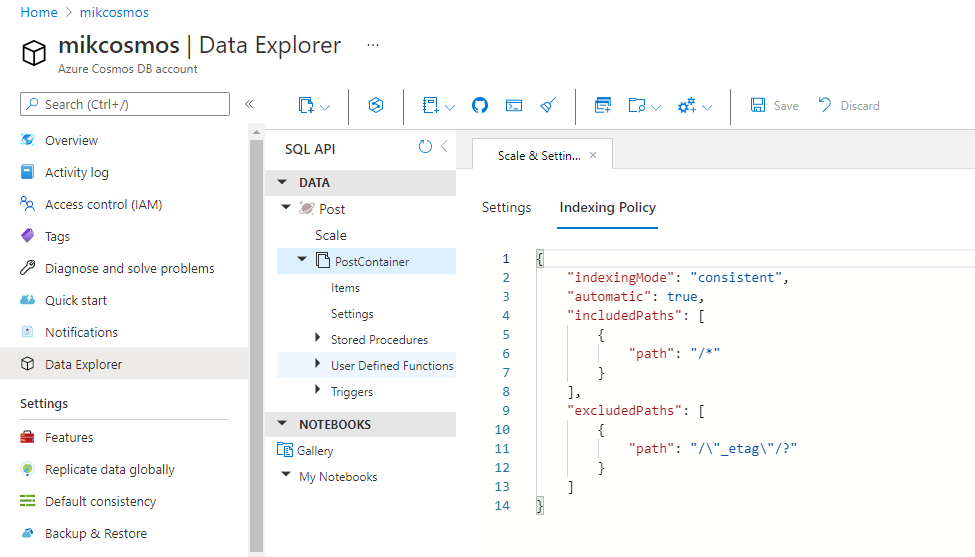


Рисунок 7 – Индексация в приложении

За счет такого подхода появляется возможность “на лету” подстраивать основные рекламные категории, если какие-то из них теряют или наоборот приобретают популярность.

Обработка сообщений из шины с помощью Azure Function представлена на рисунке 8.



Рисунок 8 – Получение данных из шины в Azure Functions

После того, как данные были получены, десериализованы и обработаны, они сохранятся в CosmosDB. На рисунке 9 представлен процесс подключения к базе и последующая запись нового поста в базу.

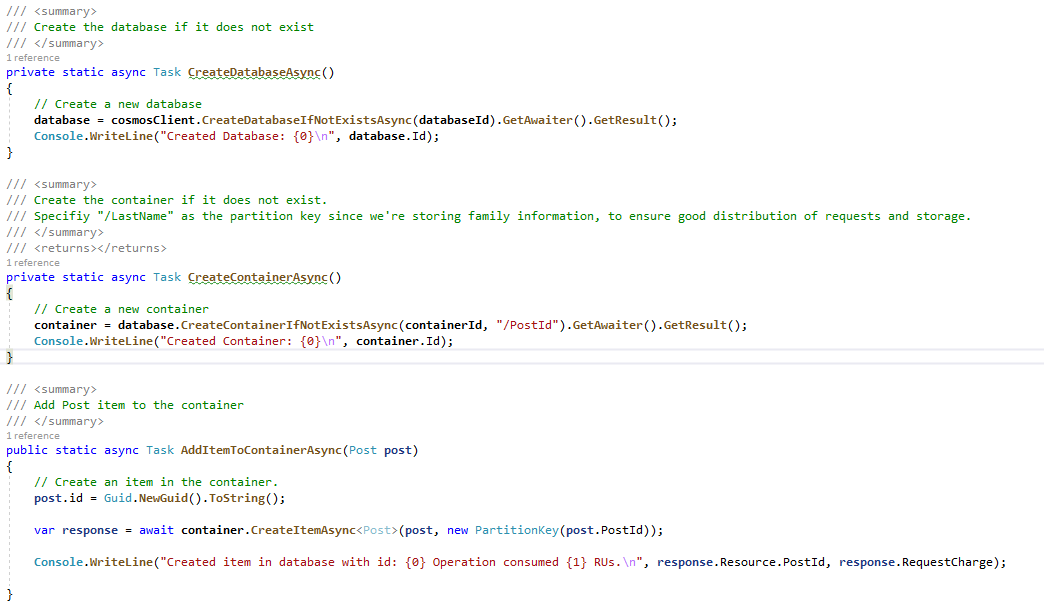


Рисунок 9 – Отправка данных в CosmosDB из Azure Functions

Количество хранимых данных в базе не ограничено, конечная стоимость использования базы зависит от количества запросов к ней, автоматического дублирования данных между регионами, использования различного уровня согласования данных между регионами, а также размера хранимых данных и количества индексов. Поэтому при продолжительном использовании CosmosDB обычно перестают использовать стандартный подход с включением всех полей в отдельные индексы и генерируют их на основании текущих требований. Помимо этого, начинают использовать усложненные составные индексы, позволяющие оптимизировать получение данных по определенным запросам.

# **4. Разворачивание полученного приложения в Azure. Реализация симуляции поступления данных в систему из разных источников/приложений**

После написания кода и проведения тестовых работ, вся инфраструктура была развернута в Azure. На рисунке 10 представлена ресурсная группа ppsoboi, которая содержит в себе все необходимые элементы: Web App для конечных пользователей, Azure Function для обработки постов, Service Bus для интегрирования отдельных компонентов, CosmosDB для хранения данных, Application Insights для мониторинга состояния системы, а также дополнительные элементы для корректной работы всей инфраструктуры.

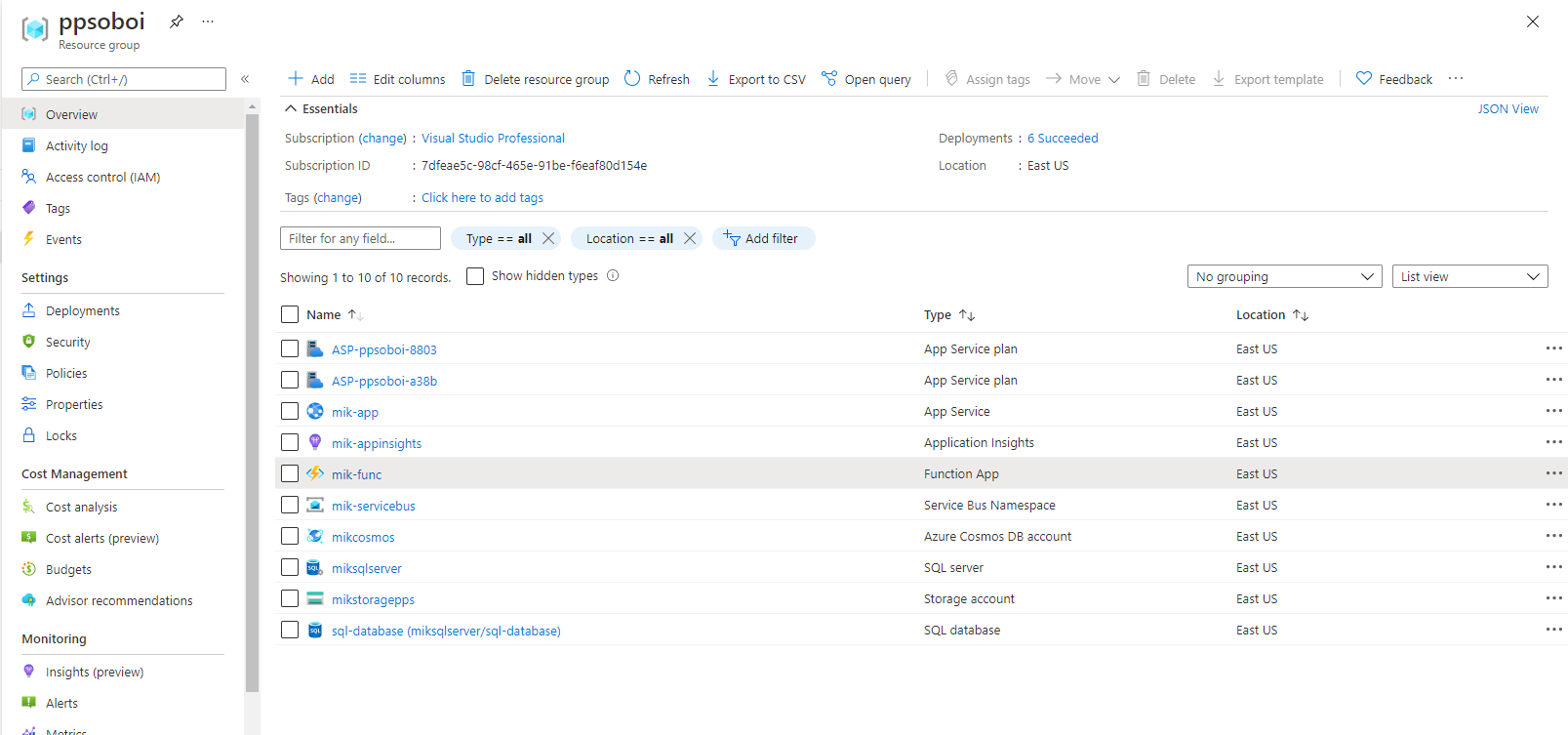


Рисунок 10 – Развернутая инфраструктура в Azure

Для проверки корректности работы системы были проведены нагрузочные тесты, предоставляемый самой Azure, а также с помощью создания дополнительных приложений, которые провели интеграционные тесты. Результаты этих тестов представлены ниже на рисунках 11, 12, 13, 14.

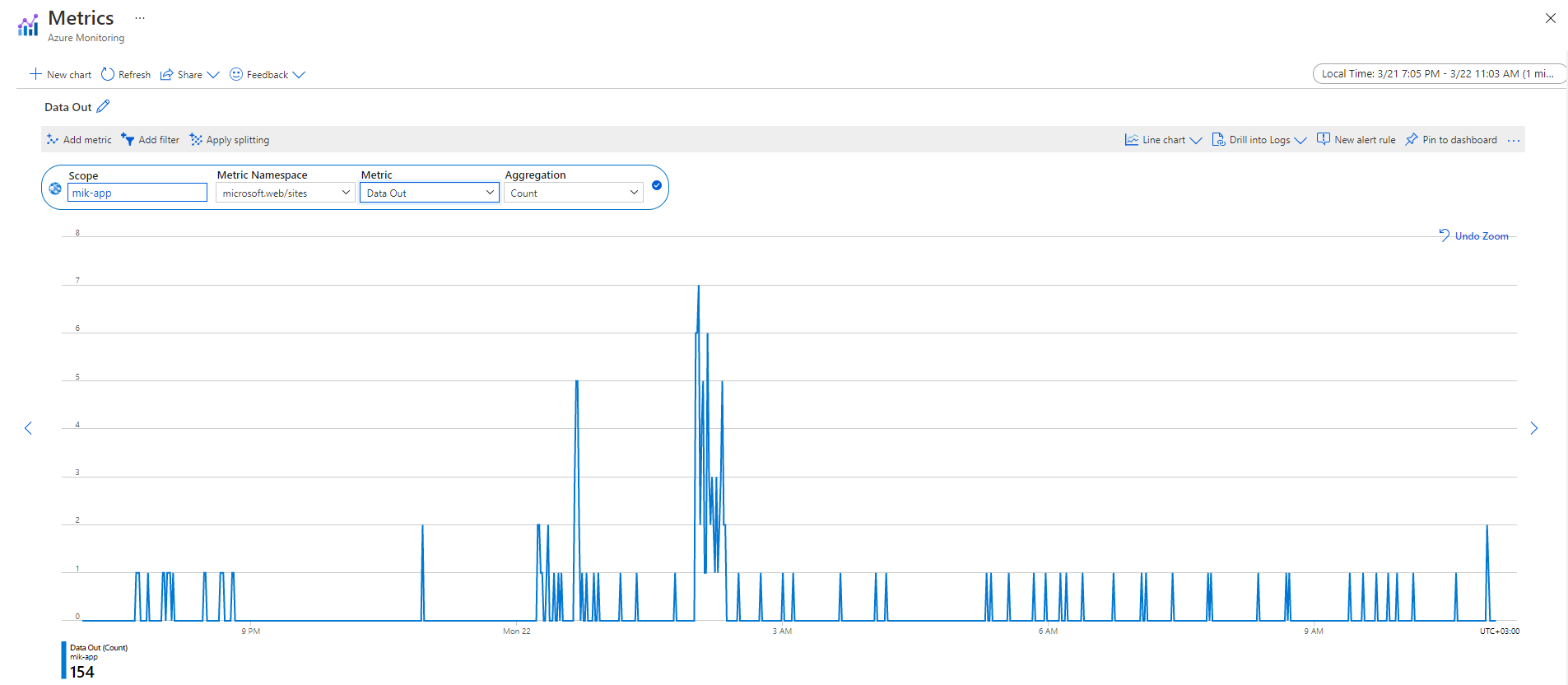


Рисунок 11 – Размер отправленных запросов из тестового приложения

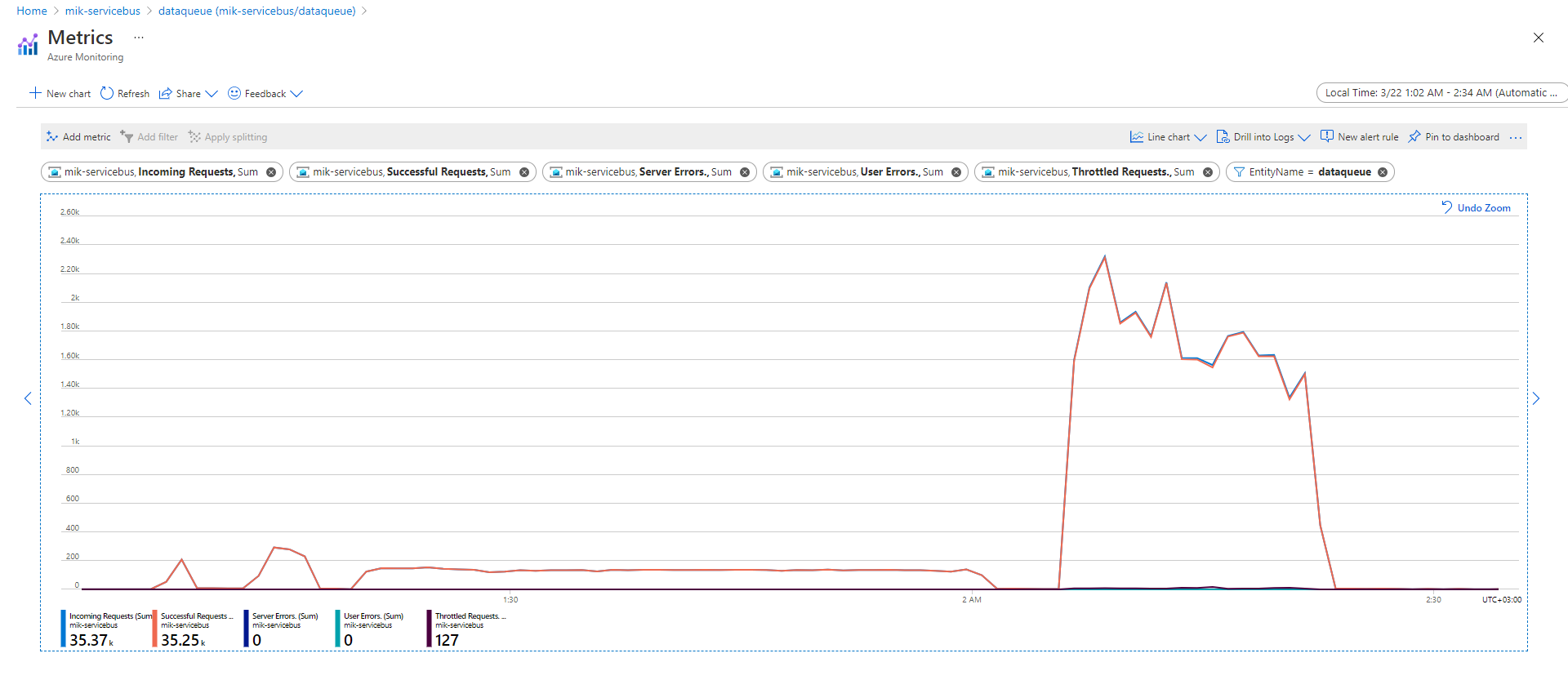


Рисунок 12 – Количество поступивших сообщений в шину

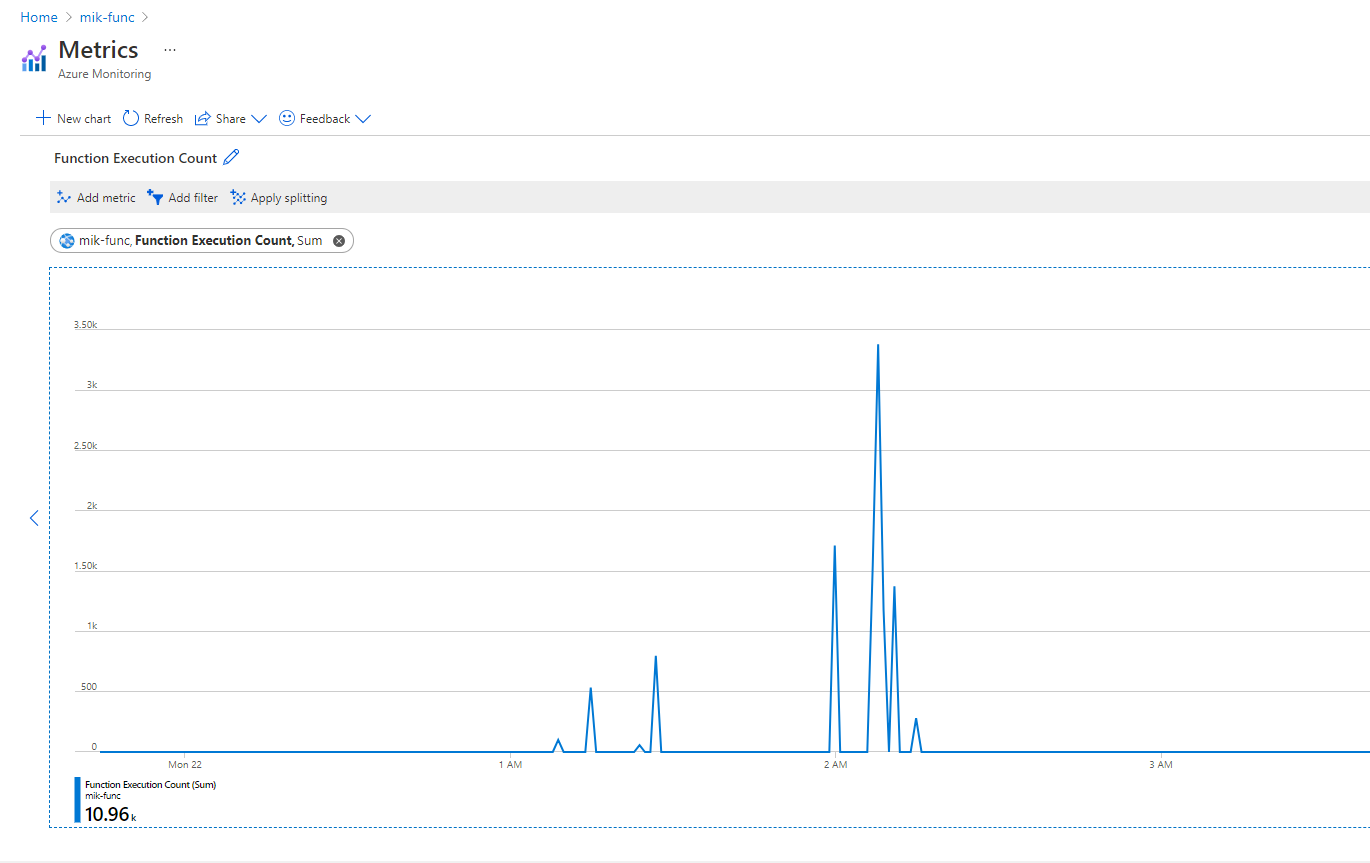


Рисунок 13 – Загрузка Azure Function

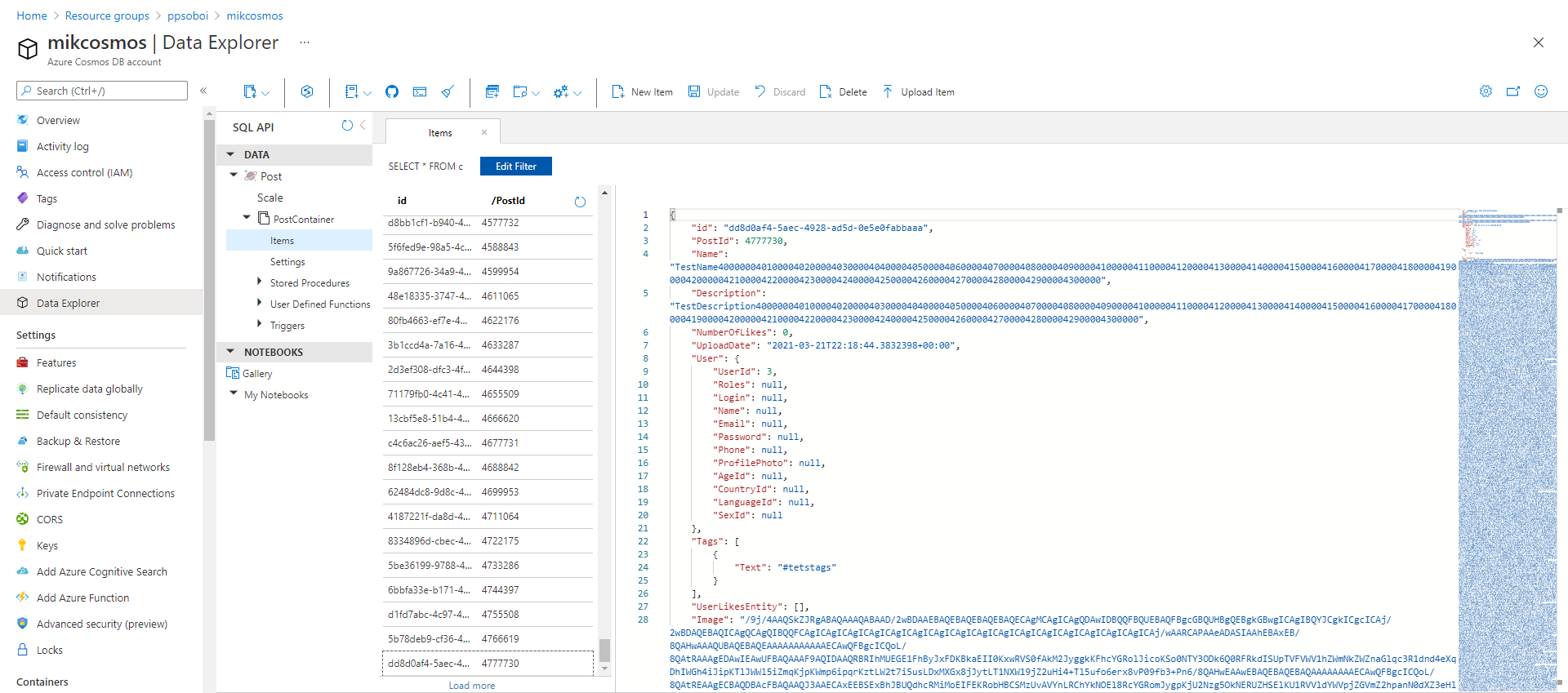


Рисунок 14 – Сохраненные данные после тестовой нагрузки

Помимо проведения нагрузочного тестирования для созданных методов были разработаны Unit Tests. На рисунке 15 представлены тесты для соединения и отправки данных в шину. На рисунке 16 представлены их результаты.

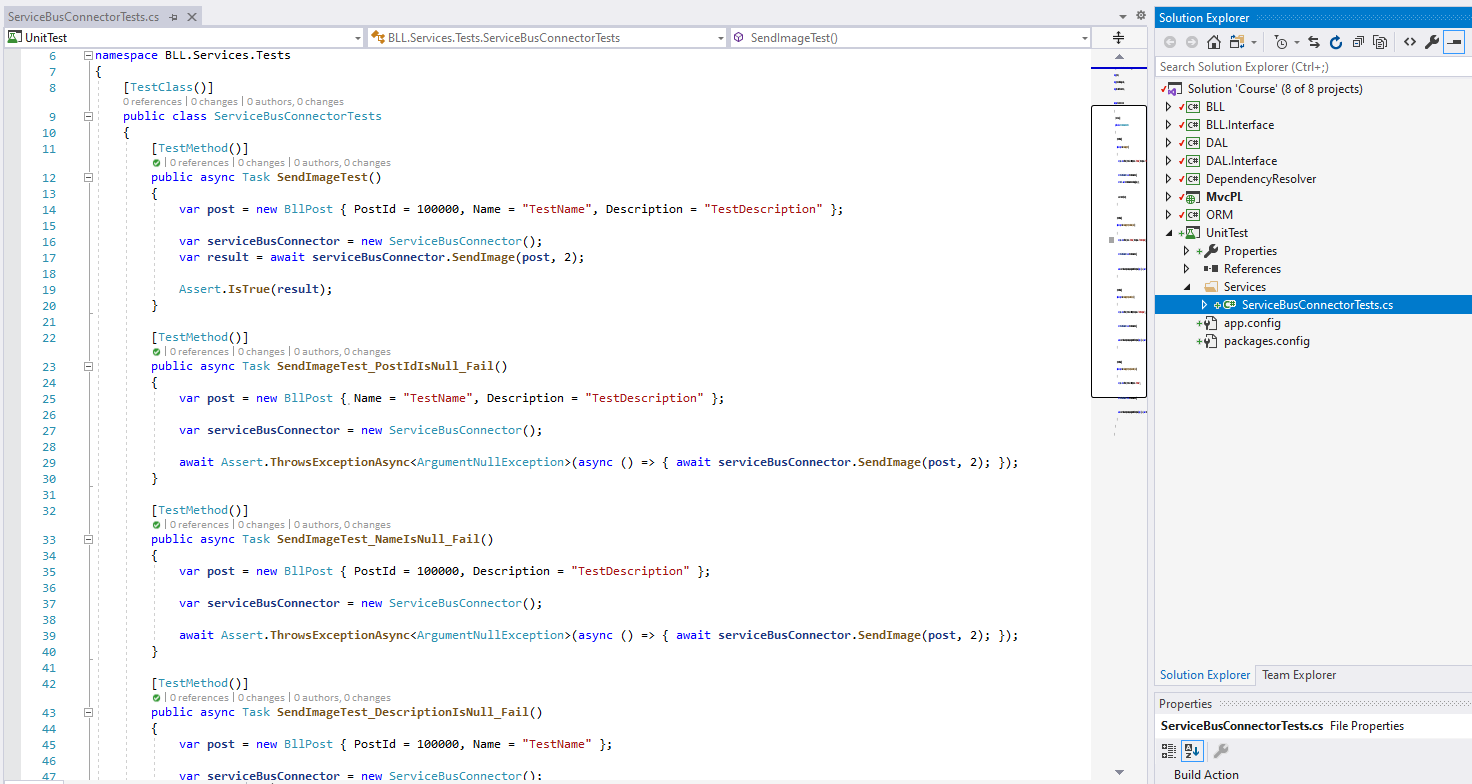


Рисунок 15 – Unit Tests для проверки соединения клиентского приложения с интеграционной шиной

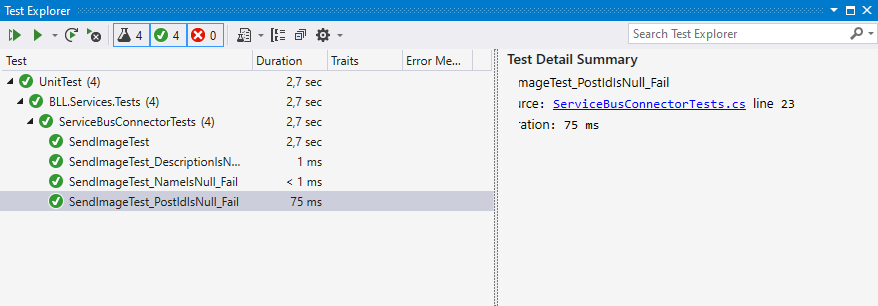


Рисунок 16 – Результаты тестирования соединения с шиной клиентского приложения

1. **Анализ соцсетей. Обоснование выбора решения для построения платформы обработки больших данных**

Социальная сеть – онлайн-платформа, которая используется для общения, знакомств, создания социальных отношений между людьми, которые имеют схожие интересы или офлайн-связи, а также для развлечения и работы.

Социальные сети на сегодняшний день являются одними из самых посещаемых ресурсов в Интернете. По данным исследовательской компании comScore их используют около 85% от всех Интернет-пользователей мира.

Социальные сети – это платформы, на базе которых участники могут устанавливать отношения друг с другом. Социальная сеть – это интерактивный многопользовательский веб-сайт или мобильное приложение, обладающий рядом обязательных качеств:

* содержание (контент) сайта создается исключительно или преимущественно его пользователями;
* сайт представляет собой автоматизированную среду, в рамках которой пользователи имеют возможность создавать связи с другими пользователями (социальные связи) или социальные объекты (тематические группы);
* пользователи имеют возможность получать статическую и динамическую информацию об объектах, существующих в данной социальной среде, о социальных связях между ними;
* пользователям доступны функции коммуникации с другими пользователями и социальными объектами.

Среди функций социальной сети следует выделить следующие:

1. Коммуникационная. В рамках коммуникационной функции люди устанавливают контакты, обмениваются новостями, информацией (фото, видео, аудиоматериалы, ссылки на сайты, комментарии, сообщения), кооперируются для достижения совместных целей (сплочение и удержание социальных связей).
2. Информационная. Поток информации имеет двухстороннюю направленность, т.к. участники общения выступают попеременно и в роли коммуникатора, и в роли реципиента.
3. Социализирующая (саморазвитие, рефлексия в системе «друзей» и «групп»).
4. Самоактуализирующая (самопрезентациия).
5. Идентификационная (при создании индивидуального профиля пользователь наполняет его информацией о себе – имя, дата рождения, семейный статус, школа, ВУЗ, интересы и пр., что позволяет осуществлять поиск анкет по заданным признакам).
6. Функция формирования идентичности. Согласно теории Фестингера, человек склонен сравнивать себя с теми людьми, с которыми у него есть большее количество схожих черт. Кроме того, согласно теории когнитивного диссонанса, похожие люди положительно оценивают друг друга. Это основной механизм, который позволяет человеку четко формулировать свои позиции в отношении других людей и групп.
7. Развлекательная. Социальные сети позволяют обмениваться не только текстовыми сообщениями, но и мультимедиными файлами, кроме того, следует отметить значение виджетов – мини-программ развлекательного характера, создаваемых сторонними производителями для расширения возможностей пользователя (игры, медиа-приложения и т.д.).

Современные социальные сети представляют из себя приложения, обрабатывающие колоссальное количество данных передаваемых, как среди пользователей, так и обрабатывающихся для контекстной рекламы. Для создания и обслуживания собственных дата-центров, позволяющих работать с такими мощностями нужны огромные денежные затраты, которые лишь возрастают, когда социальная сеть выходит на новые рынки и ей становятся необходимы локальные дата-центы для местной публики.

Для решения данной проблемы существуют Cloud сервисы, предоставляющие готовую инфраструктуру без необходимости ее физической настройки. За счет такого подхода увеличение или уменьшение необходимых мощностей для компаний происходит в разы быстрее и с меньшими материальными затратами.

Для данной социальной сети было выбрано облако Azure, являющиеся самым быстро растущим и развивающимся облаком на данный момент. Дата-центы Azure есть во всех регионах мира, что позволяет скаллировать и распространять созданное приложение на множество рынков.

Для того, чтобы конечный пользователь смог воспользоваться социальной сетью был использован Web App, в котором был размещен клиентский код. Так как с предполагаемой нагрузкой Web App мог не справиться, то была добавлена шины Service Bus, которая хранит в себе пользовательские запросы для обработки их постов. Данный механизм позволил сократить время отклика клиентской части в несколько раз, а также распараллелить работу на других сервисах. Непосредственно обработкой постов занимаются Azure Functions, которые имеют встроенный механизм скаллирования в зависимости от количества запросов и их продолжительности. Таким образом, запросы из клиентской части попадают в шину, в которой они находятся до тех пор, пока не освободиться одна из функций и не обработает полученный запрос. Как только запрос обработан, проведена необходимая работа по редактированию поста, пост отправляется в базу данных ComosDB, предоставляющую из себя неограниченное хранилище со всегда стабильным временем ответа. Для того, чтобы иметь возможность отслеживать состояния сервисов был добавлен Application Insights, который собирает статистику и лог-файлы со всех сервисов, куда он был добавлен.

# **Вывод**

В ходе лабораторной работы был реализован механизм по обработке больших данных в социальной сети. Для этого были использованы облачные сервисы Azure такие как Web App, Azure Functions, Azure Service Bus, CosmosDB, Applications Insights. За счет этого была увеличена производительность, уменьшена задержка приложения у конечного пользователя, а также снижено количество ошибок, возникающих из-за перегрузки серверов. На рисунке 15 представлена получившаяся взаимосвязь между элементами инфраструктуры.

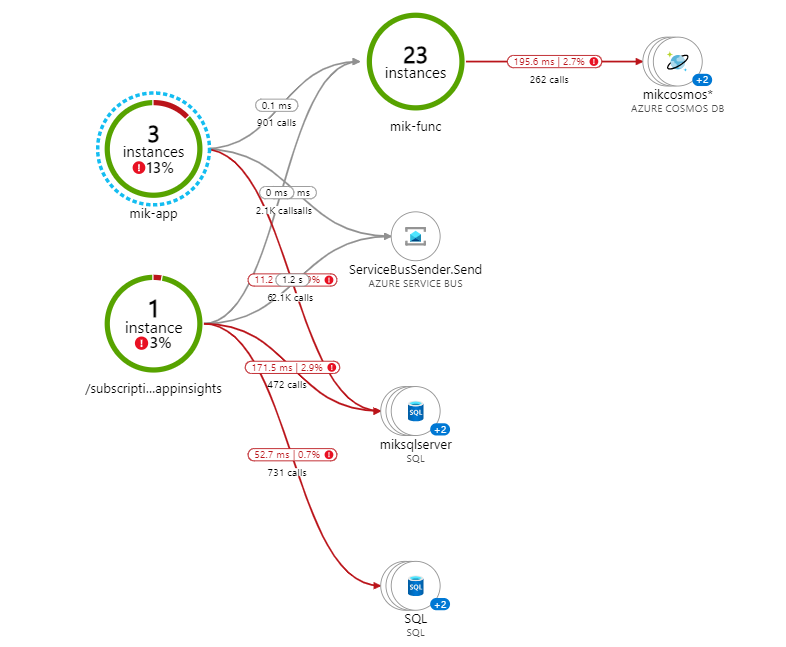


Рисунок 15 – Конечный вид взаимосвязи элементов инфраструктуры