Министерство образования Республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

**Отчет по лабораторной работе**

по курсу «Архитектурные решения для обработки больших объемов информации»

Выполнил студент группы 956241: Зязюлькин С.П.

Проверил: Стержанов М.В.

Минск, 2021

Модульность решения

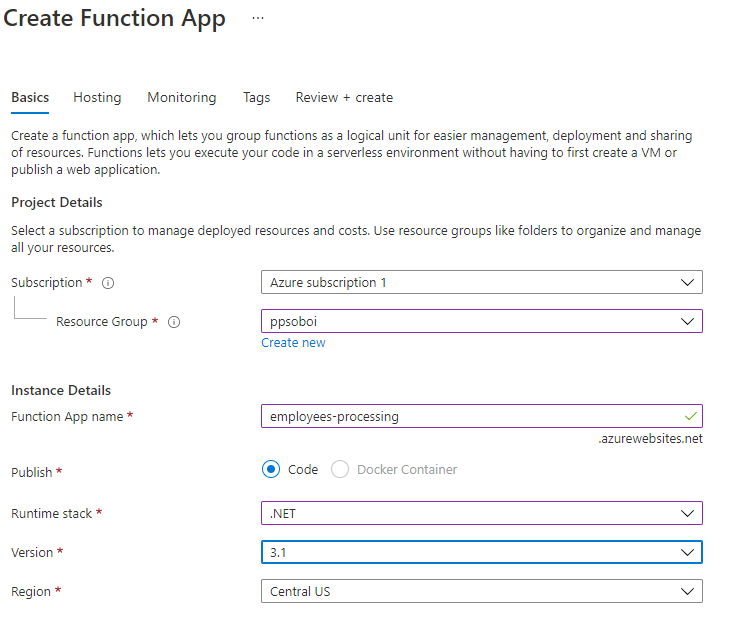
Решение представляет собой набор независимых компонентов.

* Источники данных. Выполняют загрузку данных в сервисную шину. Например, это могут быть веб-скраперы или скрипты по выгрузке офлайн-данных и т.п. Они могут быть написаны на любом ЯП, поддерживающем azure-библиотеки для работы с сервисной шиной: Java, C/C++, Python, PHP, Ruby, Go, C#/F#/VB, JavaScript/Node.
* Сервисная шина Microsoft Azure Service Bus. Накапливает новые данные о вакансиях для их дальнейшей обработки. Является частью облачного решения Microsoft Azure.
* Функции для обработки данных Microsoft Azure Functions. Выполняют предобработку поступающих в сервисную шину данных, сохраняют их в БД, обрабатывают данные из БД и собирают по ним статистику. Могут быть реализованы на любом ЯП, поддерживаемом Microsoft Azure Functions: C#, Go, Java, JavaScript, PowerShell, Python, Rust, TypeScript.
* База данных Cosmos DB. NoSQL БД, предоставляющая низкое время отклика, а также автоматическое и мгновенное масштабирование. Предоставляет большое число разных API: SQL API, MongoDB API, Cassandra API, Gremlin API, Table API.
* Анализ данных. Анализ статистики, собираемой при помощи Microsoft Azure Functions, можно выполнять на любом ЯП. Например, взаимодействуя с Azure Functions при помощи HTTP API, можно выполнять анализ в Jupyter Notebook на ЯП Python.

### Гибкость

За счёт независимости компонентов решение позволяет гибко добавлять новые или модифицировать существующие компоненты.

Например, можно добавить отдельную очередь в сервисной шине для хранения другого типа данных (например, данные о трудоустроенных сотрудниках компаний), а также отдельные Azure Functions для их обработки, причём они могут быть реализованы на ЯП, отличающемся от ЯП, на котором написаны другие функции.



**Рисунок 1 – Создание Function App для функций на ЯП C#**

### Разделённость

Все компоненты имеют стандартизированное API для взаимодействия с ними. Для сервисной шины существует библиотека для взаимодействия с ней, доступная для разных ЯП. Разработанные Azure Functions предоставляют HTTP API, которое не зависит от конкретной реализации функции. Cosmos DB предоставляет целый ряд API для взаимодействия.

Облачное решение Microsoft Azure обеспечивает автономность развёрнутых в облаке компонентов, а также их масштабируемость в случае изменения нагрузки.

Всё это позволяет легко заменять существующие и добавлять новые компоненты. Пример взаимодействия с сервисной шиной на ЯП C# представлен ниже.

static async Task SendMessageAsync()

{

// create a Service Bus client

await using (ServiceBusClient client = new ServiceBusClient(connectionString))

{

// create a sender for the queue

ServiceBusSender sender = client.CreateSender(queueName);

// create a message that we can send

ServiceBusMessage message = new ServiceBusMessage("Test message");

// send the message

await sender.SendMessageAsync(message);

Console.WriteLine($"Sent a single message to the queue: {queueName}");

}

}

Наблюдаемость решения

Облачное решение Microsoft Azure предоставляет встроенный инструмент Azure Monitor для мониторинга решения. Azure Monitor позволяет:

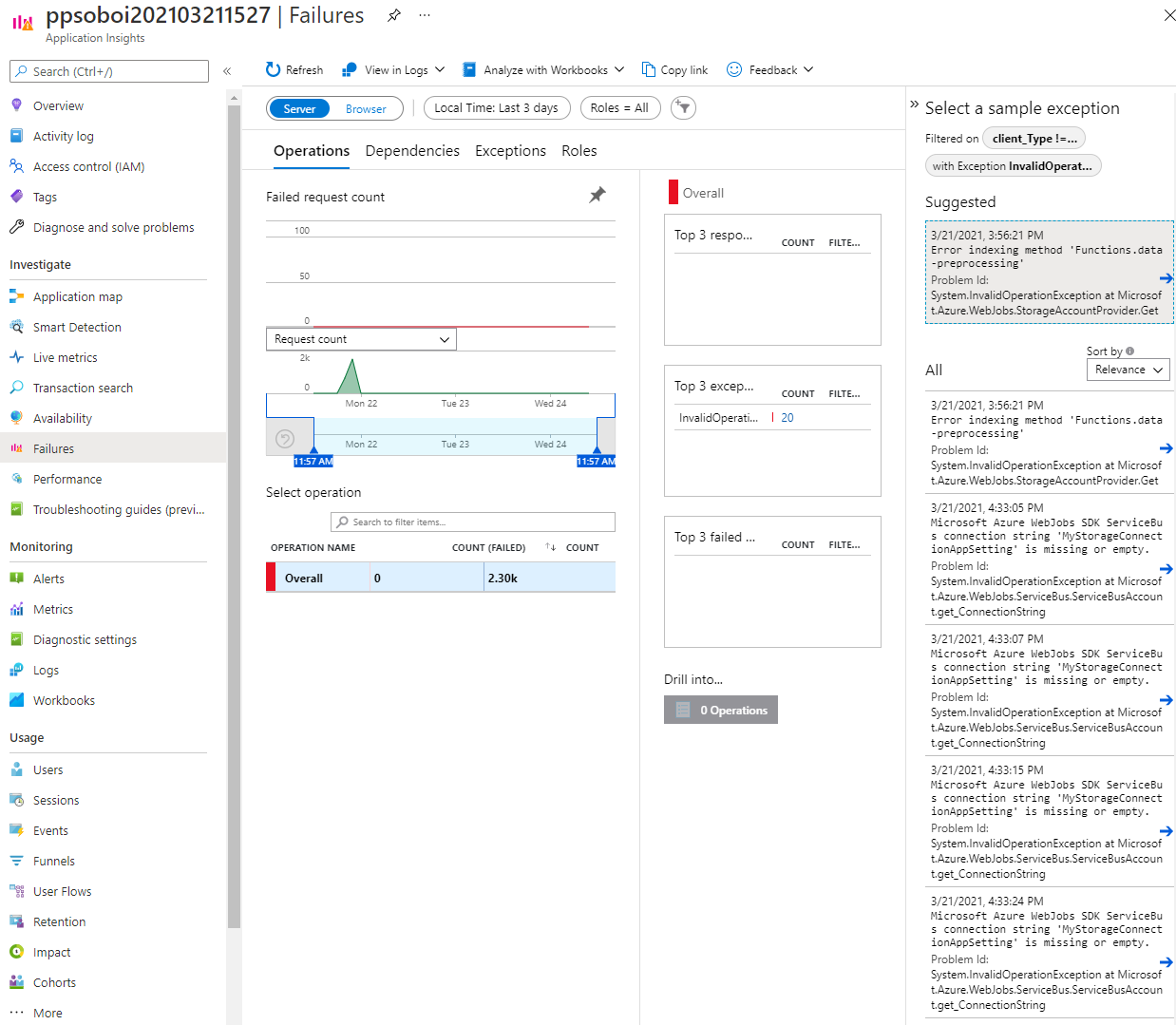
* Выявлять и диагностировать проблемы и зависимости между приложениями с помощью Application Insights.
* Сопоставлять проблемы инфраструктуры с аналитикой виртуальных машин и аналитикой контейнеров.
* Анализировать данные мониторинга с помощью Log Analytics для устранения неполадок и выполнения глубокой диагностики.
* Поддерживать работу при масштабировании с помощью интеллектуальных оповещений и автоматизированных действий.
* Создавать визуализации с помощью информационных панелей и книг Azure.
* Собирать данные от отслеживаемых ресурсов с помощью метрик Azure Monitor.

Компоненты, расположенные локально, используют встроенные в ЯП или фреймворк средства для их мониторинга. Например, модуль logging ЯП Python для анализа данных в Jupyter Notebook.

### Мониторинг

Azure Application Insights – компонент Azure Monitor – расширяемая служба управления производительностью приложений для разработчиков и специалистов по DevOps. Эта служба автоматически отслеживает аномалии производительности и предоставляет мощные аналитические инструменты, которые позволяют диагностировать проблемы и анализировать, что пользователи фактически делают в приложении. Служба работает с приложениями на разных платформах, включая .NET, Node.js, Java и Python, с размещением в локальной, гибридной или общедоступной облачной средах.

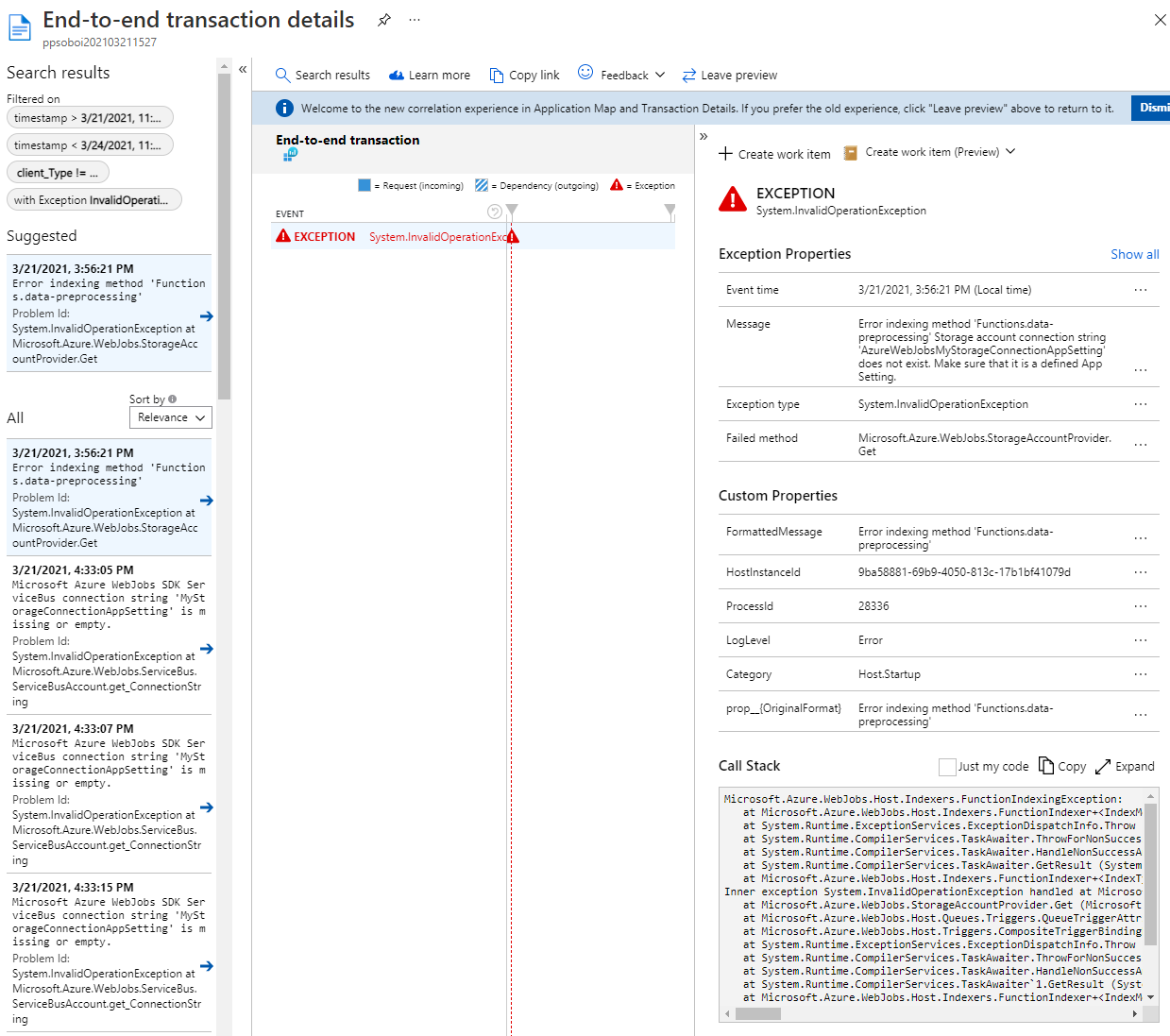
Пример анализа проблем деплоинга Azure Functions при помощи Azure Application Insights представлен ниже.



**Рисунок 2 – Анализ проблем развёртывания Azure Functions**

### Отслеживание транзакций

Служба Azure Application Insights также предоставляет end-to-end отслеживание деталей транзакции.



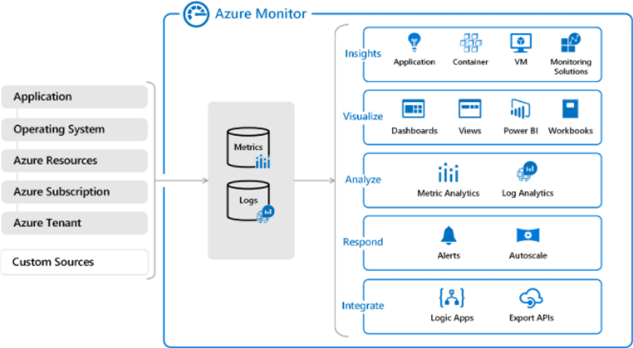
**Рисунок 3 – End-to-end отслеживание деталей транзакции**

### Агрегирование логов

Azure Monitor собирает данные со следующих уровней:

* Данные мониторинга приложений. Данные о производительности и функциональности написанного кода.
* Данные мониторинга ОС на виртуальной машине. Данные об операционной системе, в которой выполняется приложение. Это может быть Azure, другое облако или локальная среда.
* Данные мониторинга ресурсов Azure. Данные о работе ресурса Azure.
* Данные мониторинга подписки Azure. Данные об операциях и управлении подпиской Azure, а также о работоспособности и операциях самой платформы Azure.
* Данные мониторинга клиента Azure. Данные о работе служб Azure на уровне клиента (например, Azure Active Directory).

На приведенной ниже схеме показано высокоуровневое представление Azure Monitor. В центре схемы находятся хранилища данных для метрик и журналов – двух основных типов данных, используемых службой Azure Monitor. В левой части находятся [источники данных мониторинга](https://docs.microsoft.com/ru-ru/azure/azure-monitor/agents/data-sources), наполняющие эти [хранилища данных](https://docs.microsoft.com/ru-ru/azure/azure-monitor/data-platform). Справа представлены разные функции, которые Azure Monitor выполняет с собранными данными. К ним относятся такие действия, как анализ, генерация оповещений и потоковая передача данных во внешние системы.



**Рисунок 4 – Агрегирование данных мониторинга**

Разделение контента и функций

### Независимое создание контента

Решение допускает независимое добавление контента. Например, можно добавить совершенно новый источник данных – загрузку данных о вакансиях из подготовленных файлов на любом ЯП, поддерживающем azure-библиотеки. Это не потребует никаких доработок в сервисной шине, Azure Functions или БД.



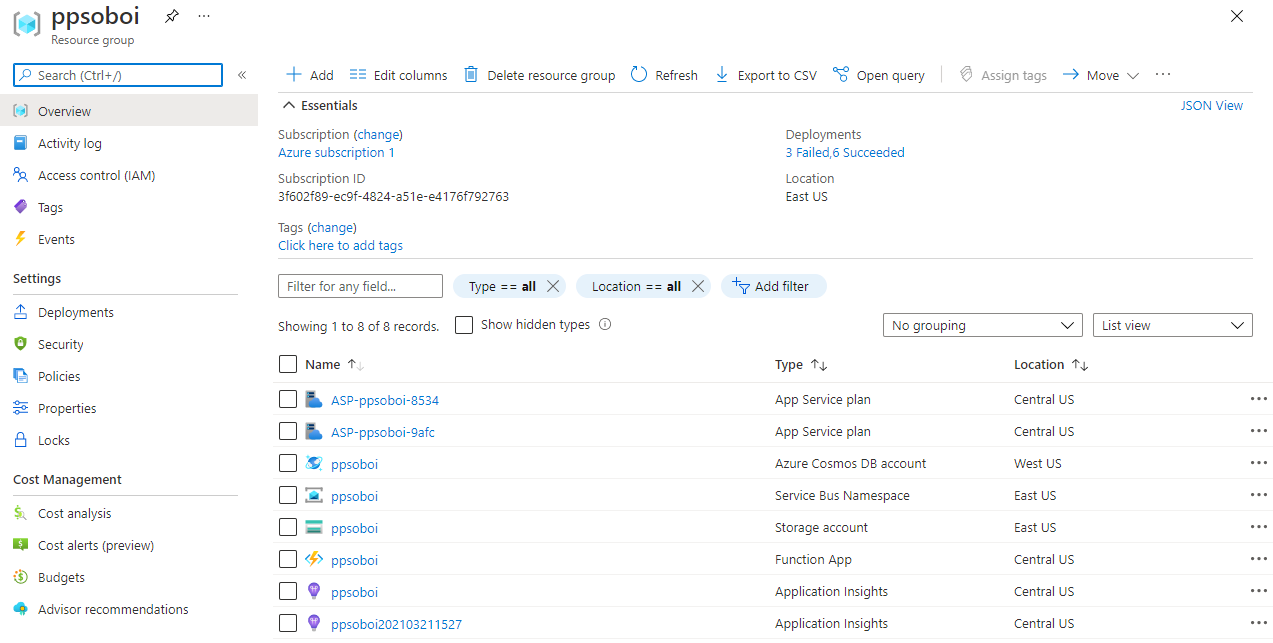
**Рисунок 5 – Пример файла с данными для загрузки**

Облачное решение Microsoft Azure позволяет гибко настраивать доступ к данным и Azure Functions при помощи Azure Active Directory. Эта служба позволяет настроить доступ пользователя только к конкретным частям решения. Например, к конкретным Azure Functions для анализа данных. При этом отсутствует необходимость вносить какие-либо изменения в код.

### Разделение ответственности

Решение имеет модульную архитектуру. За загрузку новых данных о вакансиях, предобработку данных, хранение данных, анализ данных и построение аналитики отвечают разные компоненты – скрипты для загрузки данных, работающие с azure-библиотеками, Azure Service Bus, Azure Functions, Cosmos DB, скрипты для построения аналитики. Это позволяет снизить системную сложность, а также повысить надёжность и гибкость решения.

Облачные компоненты решения представлены на следующем рисунке.



**Рисунок 6 – Облачные компоненты решения**

Готовность к выходу на рынок

### Возможности конфигурирования

Осуществлять брендирование решения для последующего выхода на рынок, а также создавать шлюзы API для существующих внутренних служб позволяет служба Azure API Management (APIM). Система состоит из следующих компонентов.

Шлюз API – конечная точка, которая:

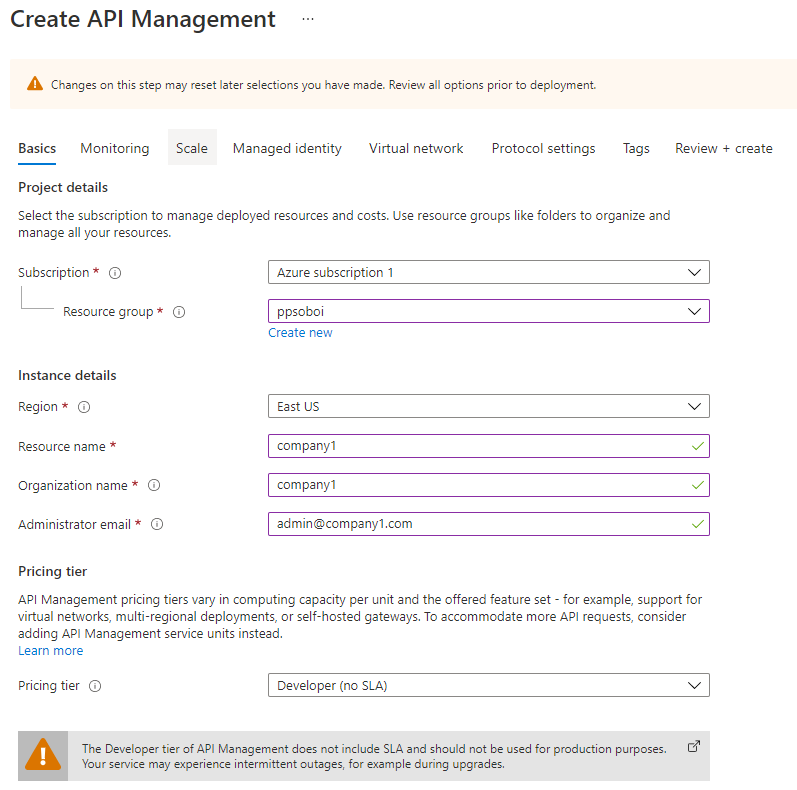
* принимает вызовы API и направляет их на серверы;
* проверяет ключи API, маркеры привязки JWT, сертификаты и другие учетные данные;
* принудительно применяет квоты использования и ограничения скорости;
* преобразует API в режиме реального времени без изменения кода;
* кэширует ответы сервера, если они настроены;
* регистрирует метаданные вызова для аналитики.

Портал Azure – это административный интерфейс, предназначенный для настройки программы API. Он используется, чтобы:

* определять или импортировать схемы API;
* упаковывать интерфейсы API в продукты;
* настраивать политики, например квоты или преобразования, в интерфейсах API;
* получать дополнительную информацию на основе аналитики;
* управлять пользователями.

Портал разработчика – это основная веб-платформа для разработчиков, которая позволяет:

* читать документацию по API;
* использовать API через интерактивную консоль;
* создать учетную запись и оформить подписку, чтобы получить ключи API;
* получить доступ к аналитике по использованию.



**Рисунок 7 – Подключение службы API Management**

### Ответственность компонентов

Решение состоит из независимых компонентов, каждый из которых имеет свою область ответственности и стандартизированный интерфейс. Это позволяет выполнять доработку и тонкую настройку компонента решения, не затрагивая остальные части решения.

В программной реализации компонентов используются инструменты документирования ЯП, при помощи которого компонент реализуется. Например, для Azure Functions, реализованных на ЯП Java, используется Javadoc. Javadoc – генератор документации в HTML-формате из комментариев исходного кода на Java. Javadoc – стандарт для документирования классов Java.

Пример Javadoc-комментариев в коде представлен ниже.

/\*\*

\* Azure Function with Azure Service Bus Queue trigger.

\*/

public class QueueTriggerFunction {

…

/\*\*

\* This function will be invoked when the queue 'data-preprocessing' receives a new message. The message contents are provided as input to this function.

\*/

@FunctionName("data-preprocessing")

public void run(

@ServiceBusQueueTrigger(

name = "message", queueName = "data-queue", connection = "MyStorageConnectionAppSetting")

RawJobView message,

final ExecutionContext context

) {

…

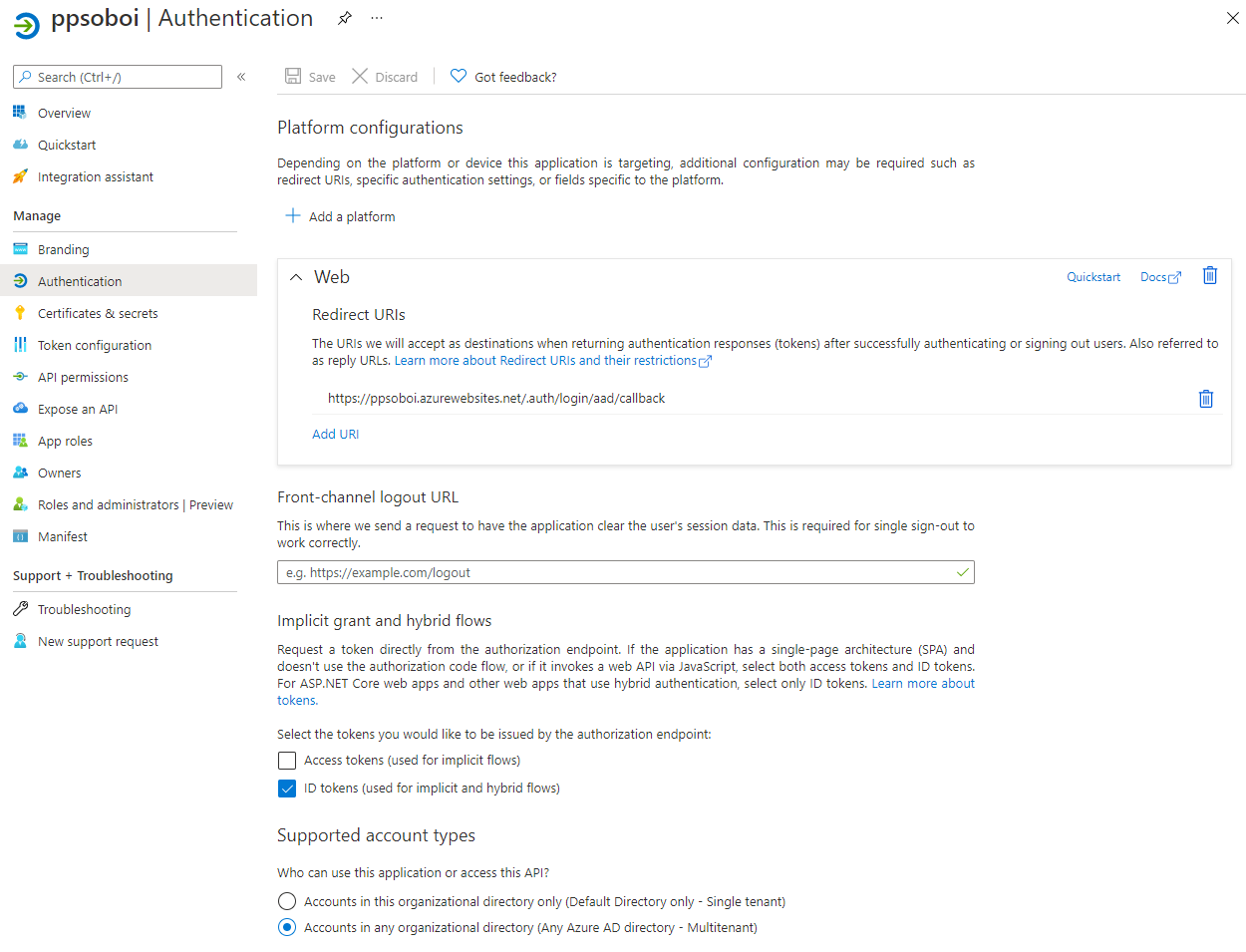
Лёгкость добавления новых пользователей

### Управление пользователями

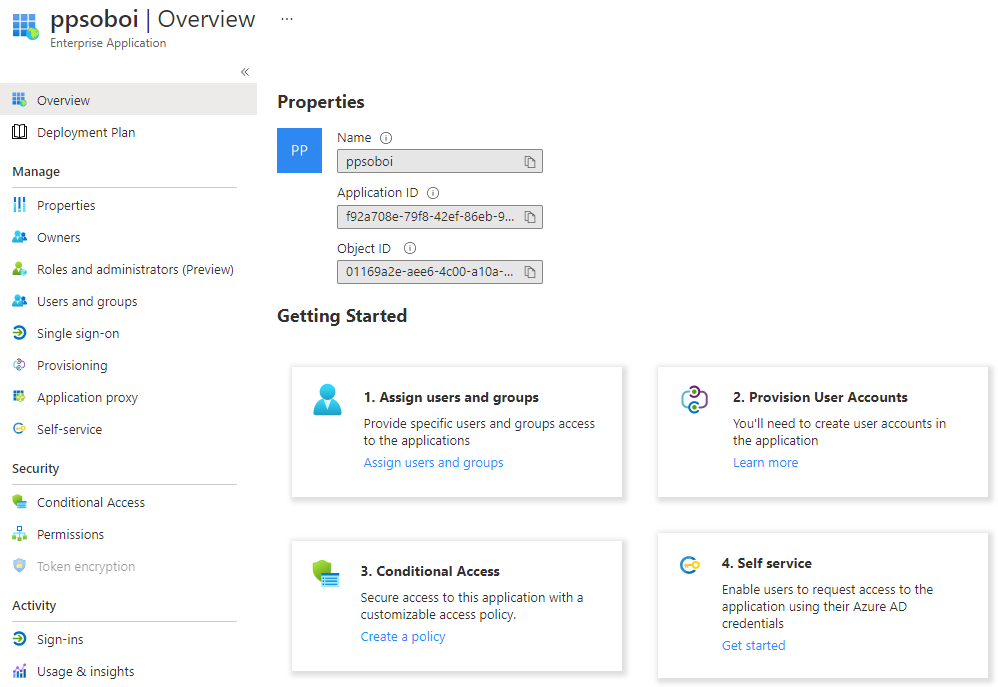
Для управления пользователями, работающими с решением, используется служба Azure Active Directory.

Azure Active Directory (Azure AD) дает возможность организовать объекты, такие как пользователи и приложения, в группы, называемые tenants. Azure Active Directory позволяет настраивать политики и доступы как для отдельных пользователей, так и для групп пользователей в целом.

Решение настроено на поддержку нескольких групп (multi-tenancy).



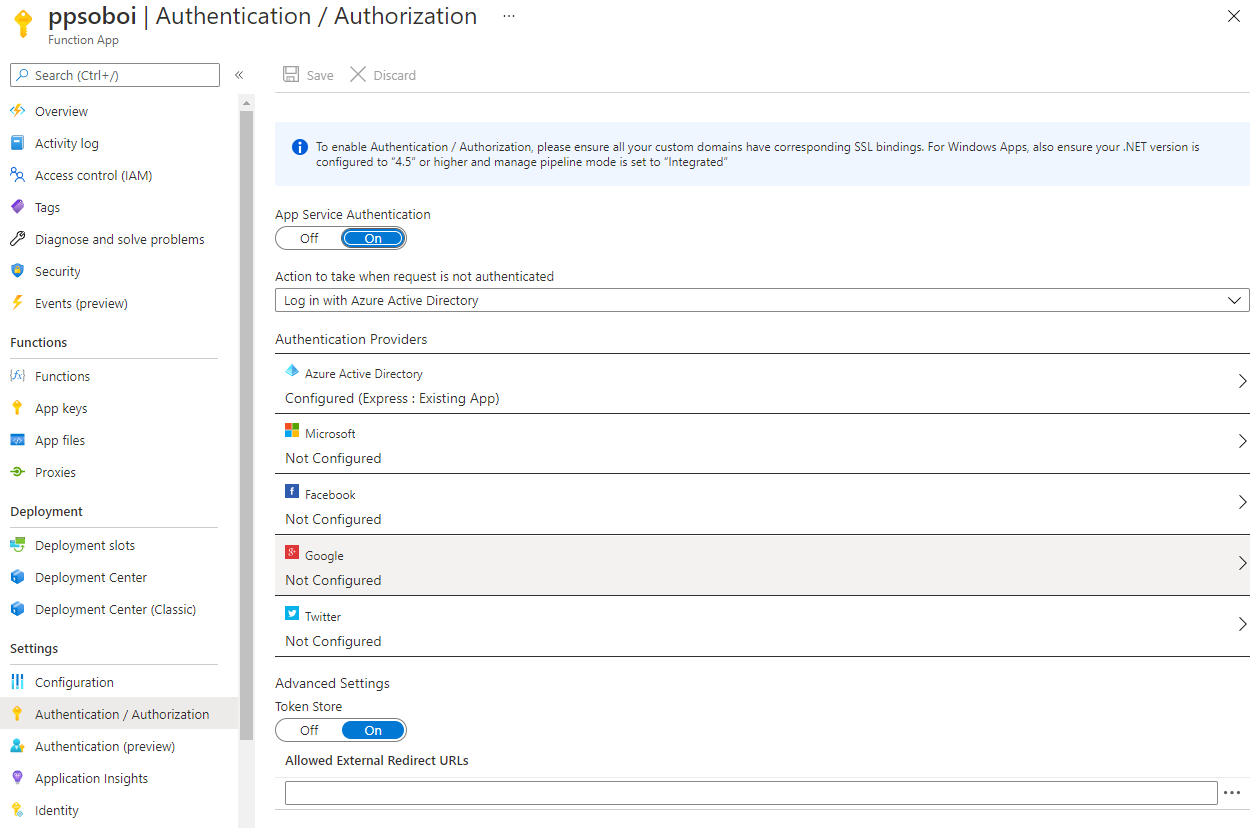
**Рисунок 8 – Включение multi-tenancy для решения**



**Рисунок 9 – Настройка пользователей и доступов**

### Разделение данных и пользователей

В рамках решения все пользователи получают доступ к общей базе данных, содержащей все собранные на текущий момент данные о вакансиях. Для разных пользователей и групп пользователей могут настраиваться различные доступы к HTTP API для аналитики этих данных. Например, некоторые пользователи могут не иметь доступа к некоторой части API, предоставляющей продвинутую аналитику. Также в будущем можно ограничить доступ к определённым группам вакансий (например, по географическому принципу – доступны данные только конкретных стран). В таком случае каждому пользователю будет доступно только некоторое подмножество всех данных о вакансиях.



**Рисунок 10 – Настройка доступа к API**

Стратегии развёртывания

Существует несколько различных типов стратегий развертывания, которыми можно воспользоваться в зависимости от цели. Например, может потребоваться внести изменения в некое окружение для дальнейшего тестирования или в подмножество пользователей/клиентов, или возникнет необходимость провести ограниченное тестирование на пользователях, прежде чем сделать некую функцию общедоступной.

Rolling (постепенное, «накатываемое» развёртывание). Эта стратегия предполагает последовательную замену инстансов приложения со старой версии на новую.

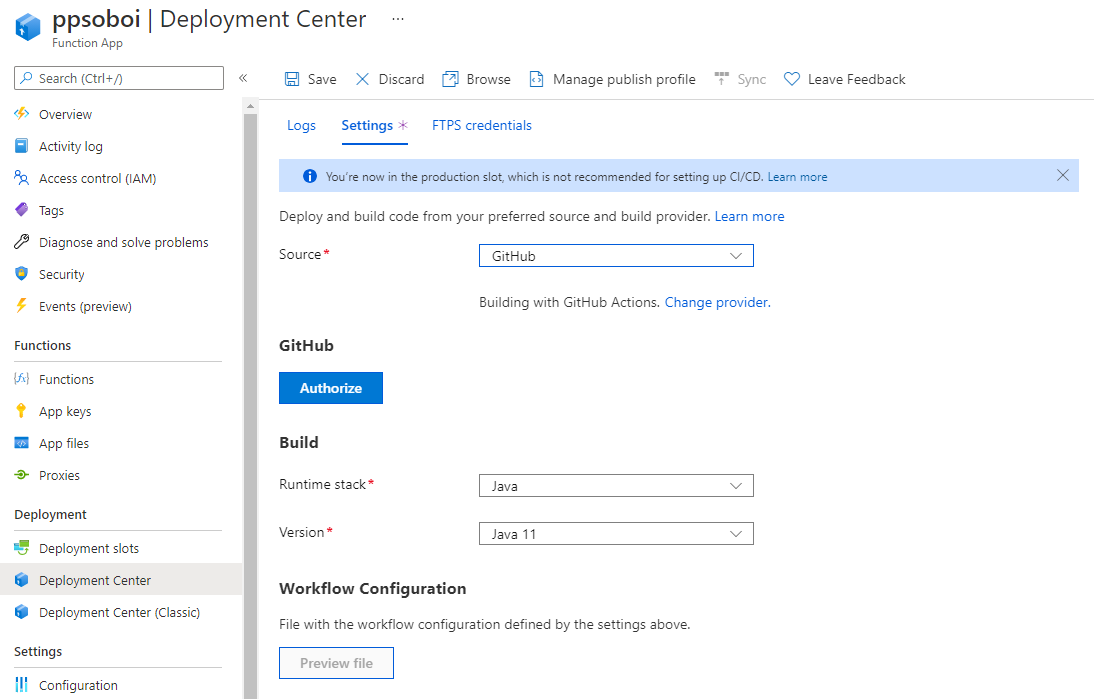
Blue/Green (сине-зелёное развёртывание). Стратегия сине-зелёного развертывания (иногда её ещё называют red/black, т.е. красно-чёрной) предусматривает одновременное развертывание старой (зелёной) и новой (синей) версий приложения. После размещения обеих версий обычные пользователи получают доступ к зелёной, в то время как синяя доступна для QA-команды для автоматизации тестов через отдельный сервис или прямой проброс портов. После того как синяя (новая) версия была протестирована и был одобрен ее релиз, сервис переключается на неё, а зелёная (старая) сворачивается.

Canary (канареечные развёртывания). Канареечные развёртывания похожи на сине-зелёные, но лучше управляются и используют [прогрессивный](https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/) поэтапный подход. К этому типу относятся несколько различных стратегий, включая «скрытые» запуски и А/В-тестирование. Эта стратегия применяется, когда необходимо испытать некую новую функциональность, как правило, в бэкенде приложения. Суть подхода в том, чтобы создать два практически одинаковых сервера: один обслуживает почти всех пользователей, а другой, с новыми функциями, обслуживает лишь небольшую подгруппу пользователей, после чего результаты их работы сравниваются. Если всё проходит без ошибок, новая версия постепенно выкатывается на всю инфраструктуру.

### Стратегия развёртывания решения

Для управления развёртывания решений в Microsoft Azure используется служба Azure DevOps.

Azure DevOps – это встроенная служба Azure, которая автоматизирует выполнение каждой части процесса DevOps для любого ресурса Azure. Независимо от того, использует ли приложение виртуальные машины, веб-приложения, Kubernetes или любой другой ресурс, имеется возможность реализовать подход IaC (инфраструктура как код), включая непрерывную интеграцию, непрерывное тестирование, непрерывную поставку и непрерывный мониторинг с использованием Azure и Azure DevOps.



**Рисунок 11 – Настройка развёртывания решения**

Вывод

В ходе выполнения лабораторной работы было реализовано решение для обработки данных о вакансиях с функциональностью multi-tenancy и проведён анализ получившегося проекта.

Решение имеет модульную архитектуру, позволяющую гибко заменять существующие или добавлять новые компоненты. Решение имеет систему мониторинга, позволяющую отслеживать аномалии в работе, проблемы с производительностью и иные проблемы системы.

Облачные решения типа Microsoft Azure упрощают процесс разработки и поддержки решений, предоставляя механизмы контроля и управления компонентами решения, систему мониторинга и анализа решения, возможность брендирования решения, поддержку multi-tenancy, а также инструменты для CI/CD.