МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

СОЗДАНИЕ И РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЯ С ПОМОЩЬЮ ОБЛАЧНЫХ ТЕХНОЛОГИЙ AMAZON WEB SERVICES

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Горячкина Александра Владимировича

Научный руководитель:
Старший преподаватель _____ М. С. Портенко
подпись, дата

Зав. кафедрой:
к.ф.-м.н., доцент _____ М. В. Огнева
подпись, дата

Саратов 2022

ВВЕДЕНИЕ

Актуальность темы.

С каждым годом корпоративная ИТ-инфраструктура чаще использует облачные сервисы для своих потребностей.

Процесс возник пару десятилетий назад, но только в последние годы он стал еще более явным и проявляется не только в миграции в облако, но и в популярности разработки нативных облачных приложений и сервисов.

В результате то, что называют «глобальные облака» – это публичные облачные платформы, например, Google, Microsoft и Amazon, способные обеспечить бизнесу специализированную масштабируемую среду для разработки ИТ-приложений.

При этом у них получается закрыть потребности всех этапов разработки, от создания прототипа и минимально рабочего прототипа до тестирования и запуска рабочей версии.

Этому оказывает поддержку сильное развитие таких инструментов, как автоматизация разработки, бессерверные модели предоставления ресурсов облака, контейнеры, а также гибкое автоматическое распределение облачных мощностей под задачи работы приложения.

Цель выпускной квалификационной работы:

Цель бакалаврской работы – создать сервис с использованием модели обслуживания Infrastructure as a Service на технологии Amazon Web Services.

Поставленная цель определила **следующие задачи**:

1. Рассмотреть актуальность создания сервисов с использованием облачных технологий.
2. Изучить модели развёртывания облачных сервисов, их преимущества и недостатки.
3. Изучить модели обслуживания облачных сервисов.
4. Исследовать основные возможности инструментариев Microsoft Azure, Google Cloud Platform, Amazon Web Services.

5. Изучить механизм работы событийно-ориентированной архитектуры приложения.

6. Разработать сервис с использованием модели обслуживания Infrastructure as a Service на технологии Amazon Web Services.

7. Реализовать сервис по агрегации курсов обмена валют с использованием инструментария Amazon Web Services.

Методологические основы создания приложения с использованием облачных технологий представлены в работах И. Леденева, А. С. Мусафирова, З. Ш. Сувонова, Ш. А. Уралова, Ю. В. Лященко, Т. М. Мансурова, М. Таллоч, А. М. Мудранова, Е. В. Крахоткиной, В. Шринивасан, Д. Рави, Д. Раджа, В. В. Петрова, Е. Ю. Авксентьева, К. В. Брюханов, А. В. Геннадиника, Н. О. Хитрова, М. А. Горбачева, С. Паттерсона.

Теоретическая и практическая значимость бакалаврской работы.

Теоретическая значимость данной исследовательской работы состоит в том, что были успешно рассмотрены различные облачные сервисы, сервисные модели и модели развертывания облачных сервисов, а также были рассмотрены преимущества и недостатки событийно-ориентированной архитектуры.

Практическая значимость данной исследовательской работы состоит в том, что было реализовано приложение с использованием инструментария облачного сервиса Amazon Web Services. Со стороны пользователя данная разработка позволяет агрегировать курсы валют с различных банков РФ, что значительно упрощает получение информации о курсах валют. В то же время, данная выпускная работа представляет собой хороший пример практического приложения и его развёртывания с использованием модели обслуживания Infrastructure as a Service, а также с использованием современной событийно-ориентированной архитектуры, которая позволяет добиться прекраснейшей независимости модулей и большого масштабирования.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и пяти приложений. Общий объем работы – 125 страниц, из них 104 страницы –

основное содержание, включая 102 рисунка, цифровой носитель в качестве приложения, список использованных источников информации – 47 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические основы облачных сервисов» посвящен рассмотрению основных определений облачного сервиса, моделей развёртывания и моделей обслуживания.

Термин «облако» начал активно применяться к инфокоммуникационным технологиям на рынке только в 2006 году благодаря Amazon.

Облачные вычисления – это модель для обеспечения повсеместного, удобного сетевого доступа по требованию к общему пулу настраиваемых вычислительных ресурсов (серверов, систем хранения данных, сетей, приложений, услуг), которые могут быть быстро предоставлены и запущены с минимальными усилиями по управлению или взаимодействие с поставщиком сервиса.

Существуют модели развертывания облачных сервисов такие, как:

- публичные (Public);
- частные (Private);
- гибридные (Hybrid).

Публичное облако обеспечивает доступ пользователей к вычислительной инфраструктуре через интернет. Владельцем ресурсов вычислительной инфраструктуры является провайдер облачных услуг. Каждый пользователь создает столько виртуальных серверов, сколько ему нужно, и управляет ими, но физического доступа к оборудованию нет. Пользователи платят за использованную вычислительную мощность и объем хранения данных.

Частное облако работает по схожему принципу с той разницей, что «физическая» инфраструктура принадлежит пользователю и размещается в его собственном дата-центре.

Гибридное облако представляет собой сочетание моделей как частного, так и публичного облака. То есть в случае нехватки ресурсов частного облака могут дополнительно арендоваться ресурсы публичного облака, либо это может делаться в порядке обеспечения отказоустойчивости.

У облачных сервисов также существуют сервисные модели обслуживания, к ним относятся:

- программное обеспечение как услуга (SaaS);
- платформа как услуга (PaaS);
- инфраструктура как услуга (IaaS).

Сервисные модели можно представить в виде «пирамиды» доступных для контроля ресурсов. На верхнем уровне располагается SaaS – это облачные приложения, доступ к которым предоставляется через веб-интерфейс. За ним следует PaaS – платформа для самостоятельной разработки и развертывания приложений. На третьем уровне расположился IaaS – серверы, хранилища, сети, вычислительная инфраструктура, которую клиент получает в пользование для запуска своих решений.

Infrastructure as a Service (IaaS) – инфраструктура как сервис. К инфраструктуре относят такие вычислительные ресурсы, как виртуальные серверы, хранилища и сети. Это похоже на виртуальные «компьютеры», на которые можно установить всё, что нужно для работы приложения.

Platform as Service (PaaS) – платформа как сервис. Провайдеры облачных услуг могут предоставлять уже настроенные инструменты, или по-другому платформы, под разные задачи.

Software as a Service (SaaS) – платформа как услуга – это полностью настроенное и готовое к работе приложение для пользователя, выполняющее определенные функции. Отличие технологии SaaS от настольных или мобильных приложений в том, что само программное обеспечение находится в облаке.

В работе были рассмотрены три популярных провайдера облачных сервисов:

1. Облачный сервис Microsoft Azure – это общедоступная облачная платформа с различными продуктами и службами, которая предоставляет пользователям инструменты для вычислений, хранения информации, размещения приложений.

2. Облачный сервис Google Cloud Platform (GCP) – предоставляемый компанией Google набор облачных служб. Он предлагает набор вычислительных услуг для всего: от управления затратами GCP до управления данными, потоковой передачи видео через Интернет, а также инструментов искусственного интеллекта и машинного обучения.

3. Облачный сервис Amazon Web Services (AWS) – это самая распространенная в мире облачная платформа, предоставляющая более 200 полнофункциональных сервисов для центров обработки данных по всей планете.

Для разработки приложения с использованием облачных технологий применялась событийно-ориентированная архитектура.

Шаблон управляемой событиями архитектуры (событийно-ориентированная архитектура, event-driven architecture, EDA) – это парадигма архитектуры программного обеспечения, способствующая производству и потреблению событий.

Управляемые событиями архитектуры включают следующие ключевые компоненты: производитель событий, события обработки, маршрутизаторы событий и потребители событий 8. Производитель события публикует событие в брокер сообщений, который фильтрует его и передает событие определённым потребителям. Сервисы производителя и потребителя разделены, что позволяет масштабировать, обновлять и развертывать их независимо.

Подводя итог первого раздела, можно сделать вывод, что при его написании были получены знания об облачных сервисах и их моделях. Также были рассмотрены инструменты, которые предоставляют различные провайдеры облачных технологий. Для дальнейшие работы в практической части была рассмотрена событийно-ориентированная архитектура.

Примечание [AG1]: Нужно ли это писать?

Второй раздел «Создание сервиса с использованием модели обслуживания Infrastructure as a Service» посвящен реализации архитектуры приложения с использования Amazon Web Services.

Приложение состоит из трёх отдельных компонентов, представляющие собой сервисы, и одного дополнительного компонента, который содержит общие для всех сервисов конструктивные особенности. Структура проекта представлена на рисунке 1.

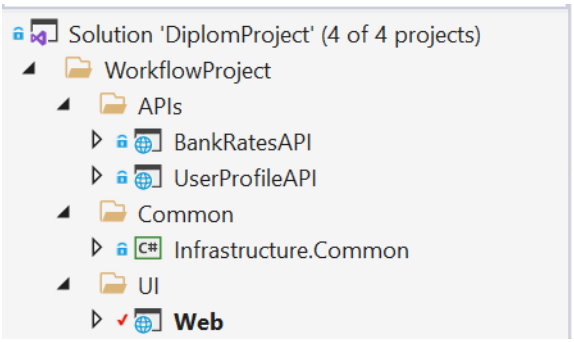

Рисунок 1 – Структура и общий вид проекта

Компоненты проектов API и Infrastructure.Common написаны на языке C# с использованием ASP.NET Core 3.1. А компонент UI Web содержит в себе ещё такие технологии, как HTML, JS, CSS и AJAX. Проекты API и UI помещены в контейнеры с помощью технологии Docker и разворачиваются удалённо.

Архитектура приложения предполагает две ступени работы приложения – запрос обработки или получения данных из сервисов и последующее получение обработанных данных из сервисов. Схема работы приложения при запросе пользователя через сервис Web изображена на рисунке 2.

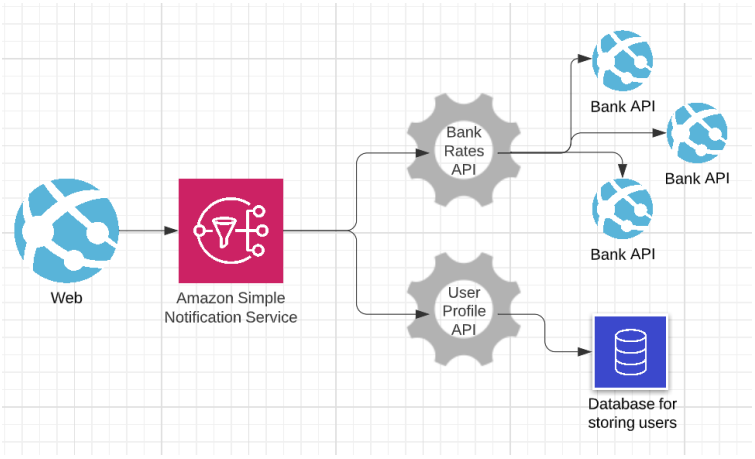

Рисунок 2 – Процесс доставки информации в обрабатывающие сервисы

Работает это следующим образом – пользователь открывает страницу на сайте, сайт в свою очередь посылает запрос в Amazon Simple Notification Service в специально созданную тему, также называемую топиком. Amazon SNS посылает сообщение во все сервисы, которые подписаны на созданную тему и не отфильтрованы политикой подписки. После получения сообщения из Amazon SNS, сервис пытается найти доступные методы для обработки запроса, используя механизмы маршрутизации ASP.NET и написанный механизм соотнесения информации из сообщения с работой маршрутизации ASP.NET.

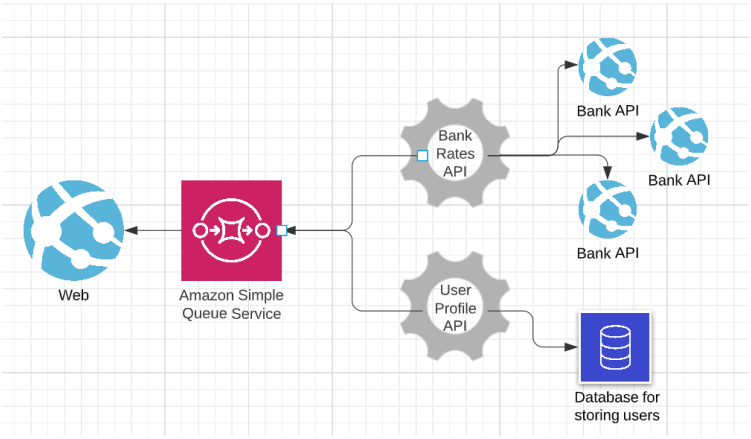

Рисунок 3 – Процесс получения обработанной информации из обрабатывающих сервисов

На рисунке 3 изображён схематичный процесс пересылки сообщений обратно web клиенту. Как только определённый сервис обработал информацию, он тут же помещает своё сообщение в Amazon Simple Queue Service. Amazon Simple Queue Service – это очередь сообщений, которая позволяет точно и однозначно обработаться всем сообщениям.

На главной странице сайта располагается информация об изменении валют на сегодня, а также статистика определённых банков по валютам с начала января. Статистика определённых банков по месяцам по выбранной валюте представляет собой графики изменения валюты.

Также была реализована возможность изменения данных профиля пользователя таких как имя, фамилия и отчество, а также указать email, номер телефона и страну проживания. Поля с номером телефона и страной являются

опциональными. Ниже этих полей можно выбрать язык сайта и текущую валюту для сравнения на главной странице сайта. В профиле пользователя присутствует возможность удаления аккаунта.

Со стороны Amazon Web Services была создана тема в Amazon SNS для публикации сообщений в сервисы, а также были созданы очереди для отправки сообщений web-клиенту обратно.

Было реализовано масштабирование при возрастающей нагрузки с использованием конфигураций запуска и Auto Scaling Groups, также был настроен балансировщик нагрузки и целевые группы для него.

Сервисы были размещены в созданную виртуальную сеть Amazon VPC с разделением на публичные и приватные подсети.

Также были настроены Docker контейнеры в экземплярах EC2, автозапуск контейнеризированных приложений и автоматическая подписка на тему SNS и очереди SQS при включении экземпляра EC2.

Подводя итог второго раздела, можно сделать вывод, что при его написании было разработано и реализовано web-приложение с помощью облачных сервисов, в частности с помощью Amazon Web Services. Цель по работы созданию и развертыванию приложения с помощью облачных технологий Amazon Web Services была выполнена.

Примечание [AG2]: Нужно ли это писать?

ЗАКЛЮЧЕНИЕ

В ходе работы были поставлены следующие задачи: данной выпускной квалификационной работы были решены все поставленные задачи и цель достигнута, а именно было реализовано микросервисное приложение в основе которого лежит событийно-ориентированная архитектура с использованием облачных технологий. Были изучены теоретические основы облачных сервисов таких как Google Cloud Platform, Microsoft Azure и Amazon Web Services.

В качестве основного облачного сервиса был выбран Amazon Web Services и были использованы такие возможности как EC2 для развёртывания приложения, SNS в качестве брокера сообщений, SQS в качестве очереди сообщений, Auto Scaling Groups для автоматического масштабирования приложения в связке с Launch Configurations и AMI для быстрого создания новых экземпляров, а также Cloud Watch для мониторинга событий и создания события для автоматического масштабирования приложений. Были также использованы такие сервисы AWS, как Amazon RDS для создания и управления базой данных PostgreSQL, VPC для создания виртуальных сетей и разграничения экземпляров EC2 по подсетям для обеспечения большей безопасности, были использованы группы безопасности для разграничения поступающего и исходящего трафика.

В ходе работы был реализован сайт для мониторинга изменения курса валют относительно выбранной валюты. Также был реализован функционал по регистрации и авторизации пользователей. Созданная архитектура позволяет легко масштабировать приложение, так как каждый потребитель событий может масштабироваться отдельно.

Основные источники информации:

1. Леденев, И. Облачные технологии - технологии будущего / И. Леденев, А. С. Мусафирова // Социально-экономические и общегуманитарные проблемы российского общества в эпоху глобализации : сборник тезисов XIV региональной открытой конференции преподавателей и студентов Университетского колледжа агробизнеса, Омск, 2017. – С. 23.
2. Z. Sh. Suvonov, Sh. A. Uralov. About modern cloud technology services. Бенефициар. – 2020. – 24-27 с.
3. Лященко, Ю. В. Преимущества и недостатки облачных технологий. Актуальные проблемы авиации и космонавтики. – 2014. 380–381 с.
4. Mansurov, T. M. Basic principles of cloud technologies. – Проблемы инфокоммуникаций. 2020г. – 5-10с.
5. Таллоч Митч и команда Windows Azure. Знакомство с Windows Azure. Для ИТ-специалистов – пер. с англ. – М.: ЭКОМ Паблишерз, 2014. — 154 с.
6. Мудранов, А. М. Работа с облачной платформой Azure. – Сборник статей XIX Международной научно-практической конференции: в 3 ч., "Наука и Просвещение", 2018. – 145-148 с.
7. Vitthal Srinivasan, Janani Ravi, Judy Raj. Google Cloud Platform for Architects. – Packt Publishing, 2018. – 474с.
8. V. V. Petrov, E. Y. Avksentieva, K. V. Bryukhanov, A. V. Gennadinik. Current issues and methods of event processing in systems with event-driven architecture – Journal of Theoretical and Applied Information Technology. – 2021, 1943-1954 с.
9. Хитров, Н. О. Особенности применения паттерна CQRS в событийно-ориентированной архитектуре высоконагруженных распределенных систем. – StudNet. – 2021, 124 с.
10. Scott Patterson. Learn AWS Serverless Computing: A beginner's guide to using AWS Lambda, Amazon API Gateway, and services from Amazon Web Services Kindle Edition. – Packt Publishing, 2019г. – 382с