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

Кафедра информатики и программирования

РАЗРАБОТКА И РЕАЛИЗАЦИЯ WEB-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ МОНОЛИТНОЙ И МИКРОСЕРВЕСНОЙ АРХИТЕКТУР: СРАВНИТЕЛЬНЫЙ АНАЛИЗ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Маслова Андрея Александровича

Научный руководитель:
доцент кафедры ИиП Е.В. Кудрина

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

Саратов 2021

ВВЕДЕНИЕ

Актуальность темы. При построении программного обеспечения в первую очередь нужно обращать внимание на выбор архитектуры, на основе которой будет вестись разработка. Выбор архитектуры задаёт темп разработки и возможности для дальнейшей оптимизации, рефакторинга, и расширения функциональности системы. Исходя из архитектуры приложения, можно оценить сложность разработки, сделать подсчёты трудоёмкости и рассчитать количество разработчиков нужное для модификации или разработки новых модулей.

Практическая значимость выбора, это поиск оптимальной архитектуры для разработки. Каждая архитектура имеет свои достоинства и недостатки, и каждая архитектура имеет определенную направленность на тип и размер приложения.

Цель бакалаврской работы – разработать и реализовать приложение «Интернет-магазин», используя разные архитектурные стили – монолитный и микросервисный, а так же сделав сравнительный анализ архитектур, оценить сложность перехода от монолитной к микросервисной архитектуре.

Поставленная цель определила **следующие задачи**:

1) представить обзор архитектур веб-приложений;

2) рассмотреть монолитную архитектуру и особенностей её реализации;

3) рассмотреть микросервисную архитектуру и особенностей её реализации;

4) разработать и реализовать веб-приложения с использованием монолитной архитектуры;

5) реализовать переход от монолитной архитектуры к микросервисной;

6) провести сравнительный анализ монолитной и микросервесной архитектур.

Методологические основы монолитной и микросервисной архитектур представлены в работах Аррорры Г. [1], Мартина Р. [2], Рейгана Р. [3], Ньюмана С. [4] , Диаса Н. [5] и Митры [6].

Теоретическая значимость бакалаврской работы. Рассмотрение достоинств и недостатков монолитной, и микросервисной архитектур, понятий связанных с архитектурой программного обеспечения, и факторов из которых формируется архитектура.

Практическая значимость бакалаврской работы. Разработка и реализация приложения с использованием различных архитектурных стилей для последующего сравнительного анализа. Проведение сравнительного анализа монолитной и микросервисной архитектур, создав идентичные приложения и условия развёртывания.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и одного приложения. Общий объем работы – 44 страницы, из них 43 страницы – основное содержание, включая 31 рисунок и 2 таблицы, 1 цифровой носитель в качестве приложения, список использованных источников информации – 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «ТЕОРЕТИЧЕСКАЯ ЧАСТЬ» посвящен, обзору архитектур веб-приложений.

Архитектура программного обеспечения – это правила и ограничения, по которым строится система. Общего определения «архитектуры программного обеспечения» не существует, так например, SEI приводит более 150 определений архитектуры программного обеспечения.

Архитектурный стиль – это множество принципов, которые рассматриваются как обобщение для описания семейства систем, архитектура программного обеспечения и архитектурный стиль взаимозаменяемы понятия.

Рассмотрены самые распространённые архитектурные шаблоны:

1) Model-View-Controller
2) Чистая архитектура

3) Многоуровневая архитектура
4) Клиент-сервер

Отдельно рассматриваются микросервисная и монолитная архитектуры. Проведён исторический обзор микросервисной архитектуры и причины её появления.

Рассмотрена монолитная архитектура, её преимущества и недостатки. Описан формат взаимодействия с приложением построенным на основе монолитной архитектуры. Монолит или монолитная архитектура – это архитектурный подход когда, конечное приложение представляет собой одно целое неделимое решение.

Рассмотрены следующие аспекты монолитной архитектуры:

1) Время работы с запросами
2) Масштабируемость
3) Гибкость кода
4) Возможные технологии для реализации
5) Возможности для развёртывания

Далее описывается микросервисная архитектура и особенности её реализации. Микросервисы или микросервисная архитектура – это архитектурный стиль, который строится на основе коллекции сервисов умеющих общаться друг с другом. Сервис – это отдельное приложение которое следует принципу единственной ответственности и следует Unix философии «Делать одну вещь и делать её хорошо».

Для построения сервисов используется REST стиль, в разделе показаны примеры ответов и запросов для сервисов в данном стиле. Описаны общие концепции Web API приложений и как происходит взаимодействие, в качестве примера приводится схема взаимодействия микросервисов и схема иллюстрирующая пример масштабирования сервисов.

Изложены основы безопасности при работе с микросервисами, перечислены самые распространённые варианты для обеспечения надежного общения между сервисами.

Рассмотрены следующий аспекты микросервисной архитектуры:

1) Тестирование
2) Развёртывание
3) Масштабирование
4) Время выполнения запросов

Самые главные аспекты при реализации микросервисов, это время выполнения запросов и развёртывание приложения, рассмотрены более подробно.

Приведены примеры и схемы запросов для сервисов, описано из чего складывается общее время выполнения. Каждая особенность реализации микросервисов рассмотрена в сравнении с монолитной реализацией, в частности отличие технологий.

Также в разделе приведена информация о контейнеризации и оркестрации приложений. Описаны наиболее популярные инструменты для создания контейнеров и управлением кластерами контейнеров, такие как

1) Docker Compose
2) Docker Swarm
3) CRI-O
4) Kubernetes
5) Amazon Web Services
6) Azure Service Fabric

Где 5 и 6 инструменты являются облачными решениями, для них приведено описание со стороны облачных сервисов. Рассмотрено отличие облачных решений и чем они удобнее для разработки и развёртывания микросервисов.

Теоретическая часть описывает все общие концепции при выборе архитектуры, даёт определение архитектуры и понимание насколько оно гибко. Описаны возможные трудности при переходе и реализации микросервисной и монолитной архитектур. Рассмотрена теоретическая часть контейнеризации приложений и развёртывания приложений при помощи

кластеров, и их дальнейшая оркестрация. Приводятся списки возможных технологий для создания приложений на основе выбранной архитектуры. Показано как один архитектурный шаблон может использоваться внутри другого.

Второй раздел «ПРАКТИЧЕСКАЯ ЧАСТЬ» посвящен реализации веб-приложения «Интернет-магазин» и сравнительному анализу архитектур. Описана постановка задачи для разработки и реализации приложения с использованием монолитной и микросервисной архитектуры.

Для разработки применялись технологии ASP.NET Core 5, ASP.NET Core 5 MVC, ASP.NET Core 5 Web API. Приложение использовало базу данных Microsoft SQL Server, для работы с базой данных использовался Entity Framework Core.

Приложение разрабатывалось в двух вариациях, с использованием монолитной архитектуры и микросервисной. Для разработки монолитного приложения использовалось одно решения в рамках которого создавались библиотеки классов содержащие логику приложения и клиентский слой использующий Razor страницы.

В приложении реализован функционал работы с товарами и работы с пользователями.

Title	Description	Cost	
Salt	Is a mineral composed primarily	13.00	Edit | Details | Delete | Add characterisitc | Add image
Meat	Is animal flesh that is eaten as food	23.00	Edit | Details | Delete | Add characterisitc | Add image
Milk	Is a nutrient-rich liquid food produced by the mammary glands of mammals.	12.00	Edit | Details | Delete | Add characterisitc | Add image

Рисунок 1 – Внешний вид списка товаров

Созданы страницы для отображения товаров, включая операции создания удаления и добавления характеристик к товару.

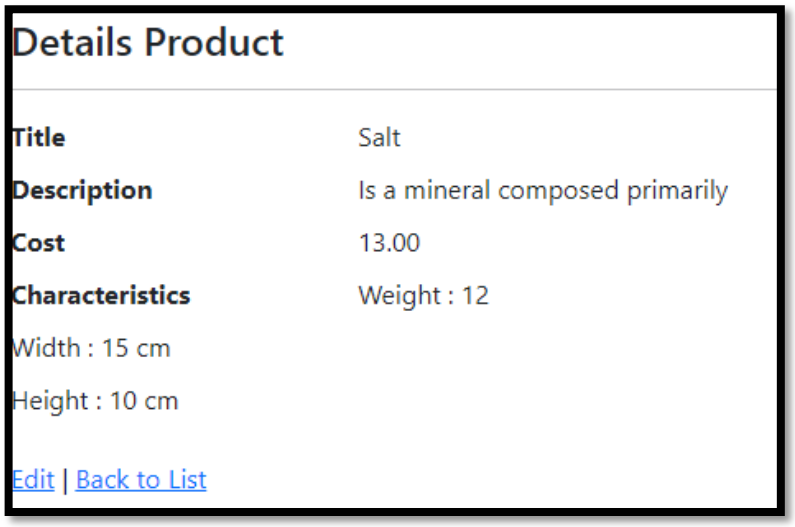

Рисунок 2 – Детали товара

Созданы формы для детализированного представления товара.

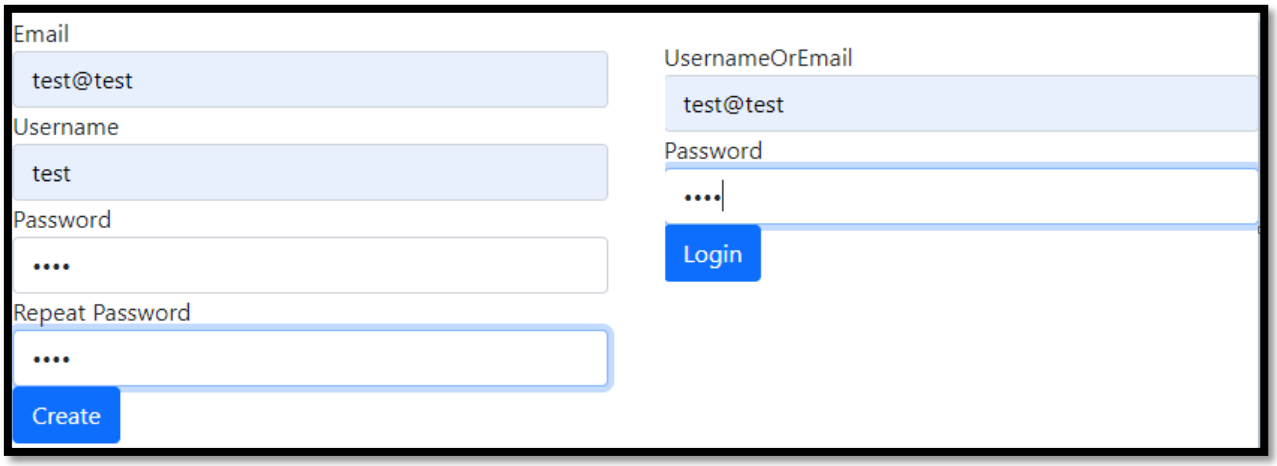

Рисунок 3 – Формы для работы регистрации и авторизации

Для работы с пользователем, созданы формы регистрации и авторизации.

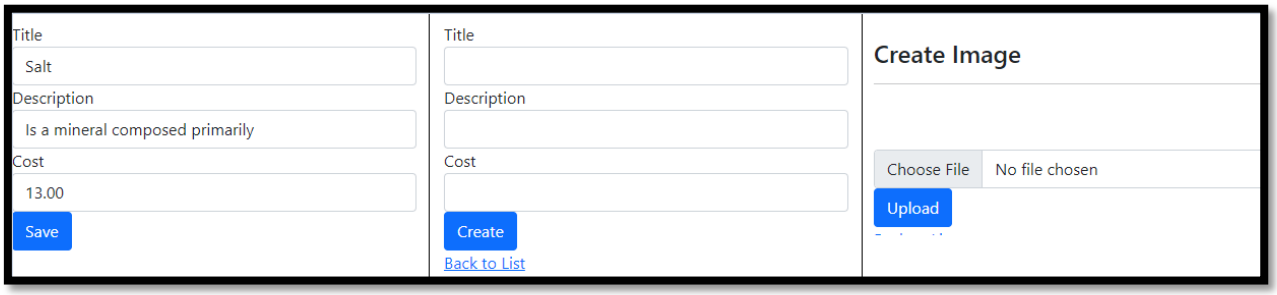

Рисунок 4 – Формы создания, редактирования и загрузки изображений

Так же для товаров предусмотрена функция добавления изображений, добавлять изображения можно с помощью отдельной формы и удалять их при редактировании товара.

Для разработки микросервисного приложения, логика разделялась на три части. В конечном приложении использовались три сервиса:

1) Сервис для клиентского отображения
2) Сервис для работы с бизнес-логикой приложения
3) Сервис для работы с базой данных

Для взаимодействия сервисов была разработана библиотека на основании HttpClient позволяющая работать с API запросами. Данная библиотека задействовалась в клиентском сервисе и в сервисе с бизнес-логикой. Сервисы для работы с базой данных и логикой, являлись Web API приложениями построенными по RESTfull концепции.

Для развёртывания и сравнительного анализа конечные приложения развёртывались в облачных сервисах Microsoft Azure с использованием различных решений. Так для развёртывания монолитной вариации приложения использовался Azure App Service, а для развёртывания микросервисного приложения использовался кластер Azure Service Fabric. База данных для обоих приложения использовалась одна, для этого был развёрнут экземпляр Azure SQL.

Сравнительный анализ проводился по нескольким критериям:

1) Время запросов
2) Потребление памяти
3) Количество кода

Для сравнения времени запросов приложения сравнивались в локальной среде и облачных сервисах. Локальная среде позволяла вычислить количество времени требуемое для обработки запроса, без учёта времени сети. Для сравнения потребления памяти, приложения разворачивались в режиме отладки и проводились замеры при помощи профилировщика Microsoft Visual Studio. При сравнении количество кода производился

подсчёт количество строчек кода в файлах с расширением cs. Подсчёт производился при помощи системы контроля версий GIT.

Для оценки влияния сети на время запросов бралось среднее время выполнения операций:

1) Создания
2) Удаления
3) Редактирования
4) Извлечения данных

На основе этого составлялись таблицы с сравнением двух архитектур.

Таблица 1 – Время (мс) выполнения приложения в локальной среде

Критерий	Монолитная архитектура	Микросервисная архитектура
Редактирование	11.73	20.33
Создание	10.26	15.33
Удаление	14.93	16.53
Получение информации	8.2	12.6

Таблица сравнения в локальной среде показывает сколько времени требуется на обработку запроса для каждой вариации приложения.

Таблица 2 – Время выполнения (мс) приложения в Azure

Критерий	Монолитная архитектура	Микросервисная архитектура
Редактирование	51.46	74.06
Создание	46.66	66.55
Удаление	51.46	84.66
Получение информации	40.46	62.66

Таблица сравнения развёрнутых приложений в Azure для учёта времени задержки сети.

При сравнительном анализе были сделаны выводы насколько вырастает код, что при переходе к микросервисам требуется написание дополнительных библиотек которые связывают все модули. Проанализирована сложность перехода к микросервисам, данная архитектура требует дополнительных усилий при развёртывании и поддерживании.

Выявлено насколько сильно влияет время задержки сети на ответы от приложения и на сколько быстро отвечает каждый модуль. В конечной вариации было построено приложение с микросервисами, которое разворачивалось внутри Azure Service Fabric и объединялось одной сетью для минимизации задержек и для обеспечения безопасности были созданы ограничения для общения между сервисами.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы все поставленные задачи были решены, благодаря чему было разработано веб-приложение «Интернет-магазин» с использованием монолитной архитектуры, произведен переход от монолитной архитектуры к микросервисной и сделан сравнительный анализ приложений построенных по разным архитектурами, а также удалось выявить достоинства и недостатки каждой архитектуры.

Архитектура программного обеспечения должна выбираться исходя из предположительной нагрузки, размера команды или команд разработки, с учётом возможных стеков технологий и доступной вычислительной мощности.

Основные источники информации:

1. Аппора Г. Building RESTful Web Services with .NET Core. Packt Publishing, 2018 – 334 с.
2. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. Питер, 2018 – 352 с.
3. Рейган Р. Web Applications on Azure: Developing for Global Scale. Apress, 2017 – 438 с.
4. Ньюман С. Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith. O'Reilly Media, 2019 – 272 с.
5. Диас Н. Microservices Security in Action. Manning, 2020 – 616 с.
6. Митра. Р. Microservices: Up and Running. O'Reilly Media, 2020 – 318 с.