ASP.NET Core – Это фреймворк, который используется для создания веб-сервисов. Каждый микросервис будет представлять собой отдельный веб-сервис, реализованный с использованием ASP.NET Core Web API.

Unit – тесты – Они помогают разработчикам обнаруживать и исправлять ошибки, прежде чем они повлияют на другие части системы. Эти тесты используются для проверки корректности работы каждого микросервиса. Каждый микросервис будет иметь свои модульные тесты, которые проверяют его функциональность.

Docker – технология, которая используется для контейнеризации микросервисов. Каждый микросервис будет упакован в отдельный контейнер Docker, что упрощает развертывание и масштабирование.

RabbitMQ – Он используется для реализации асинхронной обработки сообщений в микросервисной архитектуре. Это брокер сообщений, который используется для обмена сообщениями между микросервисами.

API Gateway – это сервер, который работает как шлюз для ваших API. Он получает запросы от клиентов и перенаправляет их на соответствующие сервисы. Ocelot – это API Gateway, который можно использовать в .NET Core приложениях. Он поддерживает динамическое маршрутизацию, балансировку нагрузки, аутентификацию и другие функции.

Итогом работы является разработанная на базе языка программирования C# микросервисная архитектура, которая обеспечивает гибкость, масштабируемость и независимость компонентов. Каждый микросервис в архитектуре взаимодействует через API Gateway, который управляет маршрутизацией и балансировкой нагрузки. Для обмена данными между микросервисами используется асинхронная обработка сообщений с помощью RabbitMQ. База данных для каждого микросервиса – это PostgreSQL, который обеспечивает высокую производительность и надежность хранения данных. Для удобства разработки и управления микросервисами используется оркестратор контейнеров Docker.

В результате работы была достигнута гибкая и масштабируемая архитектура, которая позволяет легко добавлять новые функции и обновлять существующие, не влияя на работу всей системы. Это подтверждает эффективность микросервисной архитектуры в современных приложениях и позволяет командам разработчиков более эффективно работать над большими проектами. Также следующие темы:

1. Микросервисная архитектура: принципы, преимущества и недостатки.
2. API Gateway в микросервисной архитектуре: роль, выбор и реализация.
3. Оркестрация контейнеров в микросервисной архитектуре: Kubernetes и его применение.
4. Асинхронная обработка сообщений в микросервисной архитектуре: RabbitMQ и его применение.
5. Базы данных в микросервисной архитектуре: выбор и оптимизация.
6. Контейнеризация и виртуализация в микросервисной архитектуре: Docker и Kubernetes.

Д. В. Василенко
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **А. И. Кучеров**, ст. преподаватель

ХОД РАЗРАБОТКИ ПИЛОТНОЙ ВЕРСИИ МИКРОСЕРВИСНОГО ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА УСЛУГ

В настоящее время мир бизнеса и технологий проходит через процесс динамического развития, при котором бизнес-модели и технологии неизбежно взаимодействуют друг с другом. Одним из направлений этого процесса является микросервисная архитектура, которая становится все более популярной в сфере разработки программного обеспечения.

В этой статье мы рассмотрим процесс разработки веб-приложения на микросервисной архитектуре для конгломератного холдинга, используя ряд инструментов и технологий, таких как PostgreSQL, EntityFrameworkCore, Mediatr, FluentValidation, и другие.

Стек технологий:

База данных (БД) PostgreSQL – открытая реляционная СУБД, которая обеспечивает высокую надежность, масштабируемость и производительность.

EntityFrameworkCore – популярный ORM для .NET, который позволяет разработчикам работать с базами данных с помощью объектов .NET.

Unit of Work – шаблон проектирования, который группирует операции, выполняемые с базой данных, в единое целое.

Mediatr – библиотека для упрощения взаимодействия между компонентами системы.

FluentValidation – библиотека для валидации объектов и запросов.

PredicatesBuilder – библиотека для построения предикатов для запросов к БД.

OperationResult – шаблон для возврата результатов операций, соответствующий стандарту RFC7807.

Microsoft Identity – фреймворк для аутентификации и авторизации пользователей.

Vertical Slice Architecture – архитектурный стиль, который структурирует приложение по функциональным возможностям.

Minimal API – новый стиль API в ASP.NET Core, который позволяет создавать более лаконичные и эффективные API.

OpenIddict Auth2.0 – библиотека для реализации OpenID Connect и OAuth 2.0.

Nimble Framework (Microservice Template) – шаблон для создания микросервисов.

Swagger – инструмент для создания документации к API.

AppDefinitions – шаблон для приложений в микросервисной архитектуре.

Domain Events – шаблон проектирования, который используется для обмена сообщениями между различными частями системы.

Domain-Driven Design (DDD) – методология проектирования, которая центируется на домене и использует его язык.

Приложение будет построено на основе микросервисной архитектуры, где каждый сервис будет отвечать за свою часть функциональности. Это позволит масштабировать отдельные части системы независимо друг от друга, улучшит устойчивость и обеспечит быстрое внедрение новых функций.

RabbitMQ используется для обмена сообщениями между сервисами, а Ocelot – для маршрутизации запросов к соответствующим сервисам. Это позволяет создать гибкую и масштабируемую систему, где каждый сервис может быть развернут и масштабирован независимо друг от друга.

Разработанное приложение на микросервисной архитектуре позволит эффективно управлять и отслеживать движения персонала, автоматизировать заполнение документов и предоставлять подробную информацию о работе в холдинге. Стек технологий, используемый в приложении, обеспечивает высокую производительность, масштабируемость и гибкость системы.

С. В. Васильев
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **В. Н. Кулинченко**, ст. преподаватель

ПРИМЕНЕНИЕ MICROSOFT ENTITY FRAMEWORK CORE В РАМКАХ РАЗРАБОТКИ WEB API НА БАЗЕ ФРЕЙМВОРКА ASP.NET CORE

Entity Framework Core – это фреймворк, который разработчики программного обеспечения могут использовать для доступа к базам данных (БД). EF Core представляет