жения, которые заранее извлекаются из медиа-источников, формируя каталог, из которого пользователи могут выбирать наиболее интересные для них источники.

Основные функции платформы: регистрация и авторизация пользователей; выбор пользователем языка интерфейса и языка для изучения; просмотр каталога доступных медиа-источников, с возможностью последующего выбора интересующего источника слов, словосочетаний и предложений; просмотр списков слов, словосочетаний и предложений; формирование документа на печать выбранного списка; экспорт выбранного списка в различные форматы для последующего использования в сторонних приложениях, в том числе в качестве словаря; изучение слов из выбранного источника, исключая ранее изученные слова; повторение изученных слов; просмотр пользователем статистики процесса обучения в виде таблиц и графиков, для анализа и самоконтроля.

Интерактивная платформа разработана на основе клиент-серверной архитектуры с использованием REST API, что позволило достичь высокой эффективности взаимодействия клиентской (Frontend) и серверной (Backend) частей. Серверная часть реализована на языке программирования Java с использованием Spring Framework и является прослойкой между клиентской частью и СУБД PostgreSQL. Backend принимает и обрабатывает запросы от клиента, взаимодействует с базой данных и в соответствии с заложенной бизнес-логикой возвращает результат клиенту. Клиентская часть реализована с использованием библиотеки ReactJS и предлагает интуитивно понятный и современный веб-интерфейс.

Разработанная интерактивная платформа предоставляет инструменты, которые облегчают изучение иностранных языков посредством погружения в иностранный язык через медиа.

Е. С. Ксензов

(ГГУ имени Ф. Скорины, Гомель)

СОЗДАНИЕ .NET МИКРОСЕРВИСОВ ДЛЯ WEB-ПРИЛОЖЕНИЯ

В современной разработке веб-приложений всё большее распространение получают микросервисные архитектуры, которые позволяют разбивать сложные системы на независимые, легко масшта-

бируемые компоненты. В данной работе рассмотрено создание микросервисов на платформе .NET для выполнения задач вне контура основного веб-приложения. Основной целью является повышение производительности, безопасности и масштабируемости системы за счёт распределения вычислительной нагрузки и изоляции критически важных процессов. Микросервисы позволяют вынести взаимодействие с внешними сервисами, такими как базы данных, API сторонних систем или очереди сообщений, в отдельные модули, не прибегая к использованию виртуальных машин, что значительно экономит ресурсы [1].

В рамках работы разработано несколько микросервисов, каждый из которых отвечает за выполнение определённой задачи. Например, проведение длительных вычислительных операций, обработку данных, поступающих от внешних систем, поддержание работоспособности основного вебприложения. Все микросервисы разработаны на платформе .NET с использованием современных технологий, таких как ASP.NET Core, Docker и Kubernetes. Это позволяет обеспечить высокую производительность, лёгкость развёртывания и масштабируемость. Каждый микросервис работает в изолированной среде, что снижает вероятность возникновения ошибок и упрощает отладку [2].

Использование микросервисов позволяет распределить вычислительную нагрузку между несколькими узлами, что особенно важно для высоконагруженных систем. Кроме того, изоляция процессов повышает безопасность, так как утечка данных или сбой в одном микросервисе не повлияют на работу всей системы. Взаимодействие между микросервисами осуществляется через REST API или сообщения в очереди, что обеспечивает гибкость и независимость компонентов. Это также упрощает внедрение новых функций и обновление существующих, так как изменения в одном микросервисе не требуют пересборки всего приложения.

Для реализации микросервисов использовались современные технологии и инструменты, такие как ASP.NET Core для создания RESTful API, Docker для контейнеризации и Kubernetes для оркестрации контейнеров. Это позволяет легко разворачивать и масштабировать микросервисы в различных средах, включая облачные платформы. Также использовались библиотеки для работы с очередями сообщений, такие как RabbitMQ, что позволяет организовать асинхронное взаимодействие между микросервисами.

В результате работы создана модульная система, которая демонстрирует преимущества микросервисной архитектуры для веб-приложений. Распределение нагрузки и изоляция процессов позволили повысить производительность и безопасность системы. В будущем планируется расширение функциональности микросервисов, а также интеграция с дополнительными внешними сервисами. Кроме того, рассматривается возможность использования микросервисного подхода для обеспечения импортонезависимости основного веб-приложения. Данный подход может быть применён в других проектах, где требуется высокая производительность и безопасность.

Литература

1 Cesar de la Torre. .NET Microservices: Architecture for Containerized .NET Applications / Cesar de la Torre, Bill Wagner, Mike Rousos. – Redmond, Washington: One Microsoft Way, 2023. – 335 p.

2 Smith, S. Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure / S. Smith. – Redmond, Washington: One Microsoft Way, 2019. – 113 p.

П. С. Кузьменко

(ГГУ имени Ф. Скорины, Гомель)

РАЗРАБОТКА КОМПИЛЯТОРА ДЛЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ

Сегодня различные проблемы решаются с помощью компьютера. Однако инструкции вычислительной машины, математические выражения и человеческие представления могут отличаться друг от друга. Для упрощения работы можно всё привести к одному уровню абстракции, создав язык программирования. Используя язык программирования можно в читабельной для человека форме написать математический алгоритм, а затем компилировать в понятный уже для компьютера набор инструкций.

Конечно, уже существует немало языков под различные нужды. Для быстрого и простого написания программ не редко используют Python, Lua, JavaScript, они и многие им подобные используют динамическую типизацию и выполняются, сразу не создавая отдельный