данных (в данном случае это PostgreSQL базы) и предоставляет GraphQL и REST API для работы с данными из БД. Данный подход также позволяет избежать избыточности данных и излишних связей между сервисами, а в некоторых случаях и дополнительно оптимизировать за счёт сокращения количества запросов и объёма передаваемых данных внутри системы.

А. И. Жвалевский
(*ГрГУ им. Я. Купалы, Гродно*)

РАЗРАБОТКА ИНФОРМАЦИОННО-АНАЛИТИЧЕСКОЙ СИСТЕМЫ ИСТОРИЧЕСКИХ ПАМЯТНИКОВ В РЕСПУБЛИКЕ БЕЛАРУСЬ

Задача исследования – спроектировать информационно-аналитическую систему для хранения, обработки и выдачи сведений об историко-культурных ценностях. При этом выдача данных должна основываться на введенном пользователем семантическом запросе – это такой запрос, результаты которого ранжируются в соответствии с семантической языковой моделью.

Цель семантического поиска исторических памятников – определять особенности запроса пользователя и предоставлять ему наиболее релевантные результаты. Из целей и задач вытекают проблемы, которые необходимо решить в ходе проектирования системы: проблема сбора и хранения данных и поиска среди этих данных на основе семантического запроса.

Для решения проблемы семантического поиска необходимо разработать модель, способную разбирать введенный запрос на понятные для системы команды и производить поиск на основе этих команд. За базу для решения данной проблемы используется BERT – нейронная сеть от Google, а точнее – Sentence-BERT (S-BERT). Должен использоваться асимметрический семантический поиск, так как он позволяет по краткому запросу найти абзац длиннее, отвечающий на запрос. Примером может служить запрос типа «Памятник 19-го века поэту», и вы хотите найти абзац «Памятник Александру Сергеевичу Пушкину, работы Александра Михайловича Опекушина, был установлен в Москве 6 июня 1880 года».

Для разработки серверной части использована микросервисная архитектура. Сервисы реализованы на платформе .NET 6 и языке C#. В качестве СУБД используется PostgreSQL 14. Для разработки клиентской части выбран JavaScript-фрэймворк Vue.js 3. GraphQL используется в качестве языка запросов и среды выполнения этих запросов. Микросервисы взаимодействуют между собой синхронно (REST, gRPC) и асинхронно (AMQP). Веб-приложение разворачивается с помощью технологии контейнеризации Docker. Для автоматизации развертывания, масштабирования и управления конвейеризированными приложениями используется Kubernetes. Аутентификация и авторизация реализована по стандарту OAuth 2.0 и OpenID Connect.

А. И. Жежко
(*ГГУ им. Ф. Скорины, Гомель*)

ОРГАНИЗАЦИЯ ВЗАИМОДЕЙСТВИЯ РАЗРАБОТЧИКОВ ВЕБ-ПРИЛОЖЕНИЙ В IT-КОМПАНИИ

При разработке программного продукта от программистов ожидается, что готовый продукт будет полностью соответствовать требованиям клиентов, иметь в себе все необходимые для полноценной работы функций. Так же немаловажную часть играет и соответствие продукта стандартам компании. Но если в малых компаниях над одним приложением работает от одного до нескольких человек, и для соблюдения требований достаточно одного технического задания, то уже в крупных IT-компаниях штат сотрудников исчисляется не одной сотней, и без полного контроля рабочего процесса деятельность невозможна.

Для упрощения контроля над разработкой и максимальной эффективности требуется использование специализированного программного обеспечения, позволяющее распределять задачи между сотрудниками, а руководителям контролировать, какие задачи выполнены и готовы к дальнейшим этапам, а какие требуют дополнительного внимания либо и вовсе внесения корректив.

Чаще всего небольшие компании попросту игнорируют подобные системы, в связи с малым количеством сотрудников, при котором достаточно небольшого списка задач, а то и вовсе держат всё в уме. Но на практике оказывается, что часть малоприметного базового