API OpenAI позволяет отправлять текстовые запросы и получать ответы от языковых моделей. Запросы могут быть настроены для выполнения различных задач, таких как генерация текста, суммирование, перевод, ответы на вопросы и многое другое. При отправке есть возможность указать различные параметры, такие как длина ответа, температура (которая определяет уровень креативности или случайности в ответах), контекст (предшествующий текст, который модель должна учитывать при генерации ответа) и др.

Ответы от API возвращаются в формате JSON и содержат сгенерированный текст вместе с иной полезной информацией, такой как статус запроса или использованные токены. Разработчики должны обрабатывать эти ответы, чтобы извлекать нужную информацию и интегрировать ее в приложение.

Важно учитывать, что использование API OpenAI обычно подразумевает оплату в зависимости от количества запросов и объема использованных токенов (единиц обработанного текста). Поэтому необходимо отслеживать использование API для управления расходами, особенно при масштабировании приложения. OpenAI предлагает различные инструменты и информационные таблицы для мониторинга использования токенов и оптимизации стоимости.

При работе с программным интерфейсом OpenAI важно уделять внимание вопросам безопасности и конфиденциальности. Это включает в себя защиту API-ключа, обеспечение безопасности данных пользователей и соблюдение принципов этичного использования искусственного интеллекта. Компания OpenAI предоставляет рекомендации и лучшие практики по обеспечению безопасности и соблюдению этических норм при использовании их технологий.

API OpenAI предоставляет разработчикам доступ к передовым инструментам искусственного интеллекта, открывая новые возможности для создания инновационных приложений. Однако успешное использование технологии требует понимания основ работы с API, а также внимательного отношения к вопросам стоимости, безопасности и конфиденциальности.

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

ТЕХНОЛОГИИ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Давайте начнем с краткого введения в микросервисную архитектуру. Микросервисная архитектура – это методология разработки, которая структурирует приложения как набор небольших, независимых и взаимодействующих между собой сервисов. Каждый сервис выполняет определенную функцию и может быть развернут, обновлен и масштабирован независимо друг от друга.

В основной части статьи мы рассмотрим технологии, которые могут быть использованы для реализации микросервисной архитектуры, и в частности, посвятим Entity Framework Core(EF), JWT и базе данных PostgreSQL, Asp Net Core ,Unit Tests, Docker, RabbitMQ, Api Gateway (Ocelot).

EF – Эта технология используется для взаимодействия с базой данных. Каждый микросервис, который работает с данными, будет использовать EF для выполнения операций CRUD (создание, чтение, обновление, удаление) с базой данных.

В JWT содержится три части: заголовок, полезные данные и подпись. Заголовок содержит информацию о типе токена и алгоритме шифрования, полезные данные содержат данные о пользователе или другой информации, которую необходимо передать, а подпись используется для проверки подлинности токена.

PostgreSQL – Каждый микросервис, который работает с данными, будет использовать PostgreSQL для хранения и извлечения данных.

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.