А. П. Пашук, А. Б. Гуринович

# РАЗРАБОТКА СИСТЕМЫ СЕМАНТИЗАЦИИ ТЕКСТОВОЙ ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

*Рассматривается реализация сложной системы автоматизированной обработки текстовой информации в виде взаимосвязанных самостоятельных модулей. Предлагается использование контейнеров для ускорения и упрощения процессов развертывания системы.*

## Введение

Для сложных систем, включающих в себя множество модулей, возникают проблемы с управлением кодом, его поддержкой и тестированием. Для избежания таких проблем такие системы представляют в виде относительно самостоятельных модулей. Это позволяет покрыть тестами каждый из модулей, что значительно увеличит стабильность системы в целом и позволит обновлять модули по мере необходимости, избегая необходимости обновления всей системы.

## I. Микросервисная архитектура

В общем случае, микросервисная архитектура представляет собой один из видов сервисно-ориентированной архитектуры - подхода к разработке программного обеспечения в виде распределенных, заменяемых модулей, имеющих собственные стандартизированные интерфейсы для взаимодействия с другими модулями.

Упрощенная архитектура разрабатываемой системы приведена на рисунке 1. На схеме микросервисы объединены в логические блоки, которые на практике представляют собой группы серверов, на которых разворачиваются данные микросервисы.

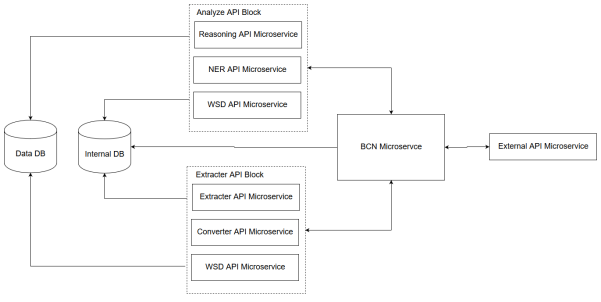


Рис. 1 – Схема системы семантизации в микросервисной архитектуре

Стоит отметить, что для упрощения на приведенной схеме не отображены зависимости микросервисов, такие как очередь задач (Celery или RQ), хранилище данных (Redis) и др.

## II. Развертывание микросервисов в контейнерах

Для быстрого и безопасного развертывания микросервисов можно использовать Docker [1]. Docker - это инструмент для развертывания изолированных контейнеров для выполнения каких-либо задач. Ниже приведен пример конфигурации Docker-контейнера для простого микросервиса, написанного на Flask.

```
FROM debian:latest

RUN apt-get update
RUN apt-get install -y python3 python3-pip

COPY ./ /opt/app/

RUN pip3 install -r /opt/app/requirements.txt

EXPOSE 80
ENTRYPOINT ["/opt/app/server.sh"]
```

В большинстве случаев каждый микросервис имеет зависимости, что делает невозможным последовательный запуск каждого микросервиса. В этом случае удобно использовать инструмент docker-compose, позволяющий явно указать зависимости каждого сервиса и запускать в указанном порядке.

## III. Выводы

Представление монолитного программного обеспечения в виде нескольких самостоятельных модулей позволяет упростить разработку и поддержку сложной системы. Развертывание микросервисов с использованием Docker-контейнеров значительно ускоряет процесс доставки программного продукта, а также упрощает его перенос на новые сервера.

1. Enterprise Application Container Platform | Docker. Mode of access: https://www.docker.com/. – Date of access: 12.03.2019.

*Пашук Александр Владимирович*, аспирант кафедры систем управления БГУИР, pashuk@bsuir.by.

*Научный руководитель: Гуринович Алевтина Борисовна*, доцент кафедры ВМиП БГУИР, кандидат физико-математических наук, gurinovich@bsuir.by