САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №5

Выполнил:

Фирсов Илья

K3441

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

Задача

По выбранному варианту необходимо будет реализовать миграцию написанного REST API на микросервисную архитектуру.

Ход работы

1. **Анализ бизнес-доменов**
   - Разделение функциональности на независимые сервисы
   - Определение границ ответственности
   - Анализ зависимостей между компонентами
2. **Проектирование микросервисной архитектуры**
   - Определение контрактов API между сервисами
   - Проектирование асинхронной коммуникации
   - Выбор технологий для каждого сервиса
3. **Реализация сервисов**
   - Создание отдельных микросервисов
   - Настройка межсервисного взаимодействия
   - Реализация отказоустойчивости
4. **Инфраструктура и оркестрация**
   - Docker Compose для локальной разработки
   - Сетевое взаимодействие между сервисами
   - Мониторинг и логирование

---

1. Бизнес-логика и архитектура CoolThing

Проект CoolThing - система автоматизации маркетинга в Telegram с двумя основными режимами работы:

Режим 1: Готовые сценарии (Scripts Mode)

Администратор создает вариативные текстовые сценарии в веб-интерфейсе, которые затем отыгрываются ботами в целевых чатах.

Режим 2: Активный поиск (Parser Mode)

Боты-парсеры мониторят множество чатов на ключевые слова, при обнаружении целевой

аудитории менеджеры через LLM предлагают релевантные услуги в личных сообщениях.

Структура микросервисов:

```
coolthing/
├── admin/                  # FastAPI веб-приложение управления
├── bot/                    # Telegram бот для target messages
├── manager/                # LLM менеджер для персонализированной рекламы
├── parser/                 # Парсер чатов на ключевые слова
├── notifier/               # FastAPI + Telegram бот для уведомлений
├── scripts/                # Координатор выполнения скриптов
├── scripts_worker/         # Исполнитель скриптов в чатах
├── scripts_watchdog/       # Docker менеджер для scripts_worker
├── scripts_watcher/        # FastAPI API для отслеживания процессов
├── metrics_worker/         # FastAPI аналитика и метрики
├── watchdog/               # Менеджер контейнеров ботов
├── database/               # PostgreSQL + Alembic миграции
├── rabbit_mq/              # RabbitMQ message broker
├── fluentd/                # Централизованное логирование
└── monitoring-setup/       # Prometheus + Grafana
```

2. Сервисы и их бизнес-ответственность

Admin Service (FastAPI)

Ответственность: Веб-приложение для управления системой, создания сценариев и просмотра аналитики

Технологии: FastAPI, aio-pika, SQLAlchemy, Beanie (MongoDB), Telegram Bot API Функциональность:

- CRUD операции с ботами, кампаниями, сценариями через REST API

- Публикация NewActiveScript событий в RabbitMQ для запуска кампаний • Управление GPT настройками и шаблонами в базе данных
- Dashboard с аналитикой кампаний и метриками Bot Service (Telegram Bot + RabbitMQ)

Ответственность: Получение target messages из очереди и отправка в Telegram

Технологии: python-telegram-bot, aio-pika, Telethon

Функциональность:

- Consumer RabbitMQ сообщений типа NewTargetMessage
- Отправка персонализированных сообщений в Telegram
- Обработка ответов пользователей
- Логирование взаимодействий

Manager Service (LLM + RabbitMQ)

Ответственность: Обработка лидов через GPT для генерации персонализированных предложений

Технологии: OpenAI GPT API, aio-pika, Telethon, SQLAlchemy

Функциональность:

- Consumer NewTargetMessage из parser
- Анализ профиля пользователя через Telethon
- Генерация персонализированных предложений через GPT
- Отправка результатов в bot сервис

Parser Service (Telethon + NLP)

Ответственность: Мониторинг Telegram чатов, поиск ключевых слов, генерация лидов

Технологии: Telethon, NLTK, aio-pika, SQLAlchemy

Функциональность:

- Автоматическое присоединение к целевым чатам
- Real-time мониторинг сообщений
- NLP обработка текста (SnowballStemmer)

- Публикация NewTargetMessage в RabbitMQ

Notifier Service (FastAPI + Telegram Bot)

Ответственность: Централизованная система уведомлений и алертов

Технологии: FastAPI, python-telegram-bot, Beanie

Функциональность:

-Получение системных событий (BotBannedEvent, ScriptCrashEvent)

-Отправка уведомлений администраторам через Telegram

-REST API для отправки кастомных уведомлений

-Логирование всех алертов

Scripts Service (RabbitMQ Coordinator)

Ответственность: Координация выполнения скриптов, получение команд от admin

Технологии: aio-pika, Beanie, SQLAlchemy

Функциональность:

-Consumer NewActiveScript из admin

-Подготовка данных для выполнения скрипта

-Координация между scripts_watchdog и scripts_worker

-Отправка статусов в scripts_watcher

Scripts Worker Service (Telethon Executor)

Ответственность: Выполнение скриптов в Telegram чатах

Технологии: Telethon, SQLAlchemy, Template engine

Функциональность:

- Выполнение вариативных сценариев в чатах
- Обработка шаблонов текста
- Мониторинг вовлеченности аудитории
- Отправка результатов в scripts_watcher

Scripts Watchdog Service (Docker Manager)

Ответственность: Динамическое создание Docker контейнеров для scripts_worker

Технологии: aiodocker, aio-pika, Docker API

Функциональность:

- Consumer NewActiveScript из scripts
- Создание Docker контейнеров scripts_worker
- Масштабирование количества воркеров
- Управление жизненным циклом контейнеров Scripts Watcher Service (FastAPI Monitor)

Ответственность: REST API для отслеживания прогресса выполнения скриптов

Технологии: FastAPI, Beanie

Функциональность:

- REST API для получения статуса скриптов
- Обновление прогресса выполнения
- Хранение истории выполнения
- Аналитика результатов кампаний Metrics

Worker Service (FastAPI Analytics)

Ответственность: Сбор и анализ метрик эффективности маркетинговых кампаний

Технологии: FastAPI, cloud dump services, analytics libraries

Функциональность:

- Сбор метрик из различных источников
- Агрегация данных по кампаниям
- Построение отчетов и дашбордов
- Выявление трендов и аномалий

Watchdog Service (Docker Bot Manager)

Ответственность: Управление жизненным циклом ботов в Docker контейнерах

Технологии: aiodocker, APScheduler, SQLAlchemy

Функциональность:

- Мониторинг состояния ботов
- Автоматический перезапуск упавших контейнеров
- Масштабирование количества ботов
- Управление ресурсами (CPU, память)

3. Бизнес-процессы и обоснование микросервисной архитектуры

Scripts Mode: Автоматизированное выполнение сценариев

Бизнес-процесс: Администратор создает вариативный сценарий → система автоматически выполняет его в целевых Telegram чатах → отслеживает вовлеченность аудитории → предоставляет аналитику результатов.

Обоснование разделения:

Admin Service - управление сценариями (FastAPI для веб-интерфейса)
Scripts Service - координация (RabbitMQ consumer для команд)
Scripts Watchdog - контейнер-менеджмент (Docker API для масштабирования)
Scripts Worker - выполнение (Telethon для Telegram API)
Scripts Watcher - мониторинг (FastAPI для REST API)

Parser Mode: Проактивный поиск и

персонализация

Бизнес-процесс: Парсер мониторит чаты на ключевые слова → генерирует лиды → менеджер анализирует профили через LLM → отправляет персонализированные предложения в личку.

Обоснование разделения:

Parser Service - поиск лидов (Telethon + NLTK для NLP)
Manager Service - LLM обработка (OpenAI API + Telethon)
Bot Service - доставка сообщений (Telegram Bot API)
Notifier Service - алерты (FastAPI + Telegram для уведомлений)

Обоснование микросервисной архитектуры

- Технологическая независимость - каждый сервис использует оптимальный стек
- Масштабируемость по нагрузке - сервисы масштабируются независимо
- Отказоустойчивость - сбой одного сервиса не влияет на всю систему
- Командная разработка - разные команды работают над разными сервисами
- Инкрементное развертывание - обновление сервисов без downtime

4. Docker Compose оркестрация

Docker Compose обеспечивает локальную разработку и тестирование микросервисной архитектуры. Конфигурация разделена на несколько файлов для

лучшей организации и переиспользования.

5.Инфраструктура и инструменты

Fluentd обеспечивает централизованное логирование, Prometheus + Grafana предоставляют мониторинг и алертинг. Docker Compose организует локальную разработку с правильными зависимостями между сервисами.

Сервисы реализуют health checks, graceful shutdown и event-driven коммуникацию для обеспечения высокой доступности и отказоустойчивости.

Вывод

Проект CoolThing демонстрирует зрелую микросервисную архитектуру с 11 сервисами, каждый из которых имеет четкую ответственность и использует соответствующие технологии.

Ключевые особенности архитектуры:

- Domain-driven design с разделением на Scripts Mode и Parser Mode
- Event-driven коммуникация через RabbitMQ для loosely coupled сервисов
- Docker контейнеризация с динамическим управлением контейнерами
- Многоуровневая архитектура с API, workers, monitors, coordinators

Вывод по работе

Микросервисная архитектура CoolThing включает:

- 11 сервисов с четким разделением по бизнес-ответственности
- RabbitMQ для асинхронной коммуникации между сервисами
- Docker Compose для локальной оркестрации
- Fluentd + Prometheus + Grafana для логирования и мониторинга
- Dynamic scaling через scripts_watchdog для worker контейнеров
- Event-driven взаимодействие вместо прямых API вызовов
- Health checks и graceful shutdown для отказоустойчивости