Техническое задание: Сервис аналитики посещаемости

Проект: Сервис аналитики посещаемости сайта (аналог Google Analytics).  
Цель: Создание API для сбора, обработки и анализа данных о посещении страниц сайта. Использование технологий для обеспечения высокой производительности, масштабируемости и мониторинга.

# Функциональные требования:

1. Прием событий:  
 - Прием событий о посещении страниц через HTTP API.  
 - Сохранение информации о посещении: URL страницы, время посещения, пользовательский агент, сессия пользователя и т.д.  
  
2. Кеширование:  
 - Использование Redis для кеширования статистики посещаемости популярных страниц, чтобы снизить нагрузку на базу данных.  
  
3. Асинхронная обработка:  
 - Использование Kafka для асинхронной отправки и обработки событий посещений страниц.  
  
4. Сохранение данных:  
 - Сохранение событий о посещении страниц в базе данных PostgreSQL с использованием индексов для ускорения запросов.  
  
5. Мониторинг:  
 - Сбор метрик с помощью Prometheus (например, количество обработанных событий, время отклика API).  
 - Визуализация данных в Grafana для мониторинга производительности системы.  
  
6. Логирование:  
 - Логирование запросов и ошибок с помощью ELK Stack (Elasticsearch, Logstash, Kibana).  
  
7. API интерфейсы:  
 - Создание простого API для получения статистики о посещаемых страницах (например, топ-страницы по посещаемости за день).  
  
8. Масштабируемость:  
 - Контейнеризация системы с использованием Docker.  
 - Использование Kubernetes для оркестрации и масштабирования сервисов.  
  
9. Производительность:  
 - Оптимизация работы с БД с использованием партиционирования данных.

# Технические требования:

1. Backend:  
 - Платформа: ASP.NET Core 6.0+  
 - Основной функционал: Прием событий, асинхронная обработка, кеширование, сохранение данных в БД.  
  
2. База данных:  
 - PostgreSQL: схема для хранения информации о посещении страниц, индексы, партиционирование данных.  
  
3. Очереди сообщений:  
 - Kafka для асинхронной обработки событий посещений.  
  
4. Кеширование:  
 - Redis для хранения популярных данных.  
  
5. Мониторинг:  
 - Prometheus для сбора метрик.  
 - Grafana для визуализации метрик.  
  
6. Логирование:  
 - ELK Stack (Elasticsearch, Logstash, Kibana) для логирования системы.  
  
7. Контейнеризация:  
 - Docker и Kubernetes для оркестрации и масштабируемости.

# API интерфейсы:

1. POST /api/events  
 - Описание: Регистрирует событие посещения страницы.  
 - Запрос: JSON-объект с данными о посещении страницы (userId, sessionId, pageUrl, referrerUrl, timestamp, userAgent, device).  
 - Ответ: HTTP 200 OK при успешной регистрации события.

# Производительность и масштабируемость:

1. Использование очередей Kafka для асинхронной обработки событий.  
2. Кеширование в Redis для популярной статистики и снижения нагрузки на БД.  
3. Оптимизация работы с БД с индексами и партиционированием для работы с большими объемами данных.  
4. Контейнеризация с Docker и использование Kubernetes для масштабируемости.

# Мониторинг и логирование:

1. Использование Prometheus для сбора метрик (время отклика, количество событий, нагрузка на компоненты системы).  
2. Визуализация данных с помощью Grafana.  
3. Логирование с помощью ELK Stack для отслеживания и анализа ошибок системы.