**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **Аналитическое хранилище для KION**  **Итоговый документ** | | |
|  |  | |
| Исполнители:  Чертанов Денис Алексеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Ф.И.О /  Сафонов Николай Юрьевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Ф.И.О /  марта  25  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.  Чертанов Д. А.  \_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_  Сафонов Н. Ю.  \_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_  Заказчик:  Демиденко Савва Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Ф.И.О /  МТС. Тета  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Компания /  марта  25  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |
|  | | |
|  | |  |

**2019**

**СОДЕРЖАНИЕ И СТРУКТУРА ДОКУМЕНТА**

**1.Введение**

1.1. Наименование программы

1.2. Краткая характеристика области применения

**2. Команда**

2.1. Командные роли и распределение ответственности

2.2. Матрица ответственности

**3.Управление проектом**

3.1. Метрики проекта

3.2. Диаграмма Ганта (базовый план и итоговый состав работ)

3.3. Использование инструментов (Canvas) или систем управления проектом (Trello, Jira)

**4. Предметная область и описание продукта**

4.1. Постановка проблемы

4.2. Анализ предметной области, рамки и границы проекта

4.3. Назначение приложения

**5. Описание потенциальных пользователей**

5.1. Цель заказчика и ключевые стейкхолдеры

5.2. План управления коммуникациями

**6. Анализ конкурентов**

**7. Функциональные требования**

**8. Другие требования к продукту (нефункциональные)**

**9. Описание архитектуры приложения и используемых технологий**

**10. Первоначальный прототип**

**11. Используемые технологии**

**12. Предложения по монетизации/коммерциализации**

**13. Будущее развитие проекта**

**14. Отчеты о взаимодействии с заказчиком** (Шаблон 1-6 с подписью)

**15.Отзыв заказчика**

**16. Руководство пользователя**

**17. Список используемой литературы**

# 1. ВВЕДЕНИЕ

## Наименование программы

Аналитическое хранилище для KION.

## 1.2. Краткая характеристика области применения

Системы сбора и хранения данных о действиях пользователей широко используются в сфере анализа данных. Без таких систем, например невозможна работа рекомендательных алгоритмов. Помимо рекомендаций данные пользователей могут широко использоваться в системах для автоматизированной аналитики. Данные пользователей позволяют бизнесу принимать решения для того, чтобы оставаться актуальным и помогают быстро отслеживать изменение трендов. В последнее время данными пользуются не только крупные бизнесы, но и авторы контента, которые на основе статистики могут более точно определять предпочтения их аудитории. На основе большого количества собираемых данных можно разделить пользователей на группы и выдавать таргетированную и эффективную рекламу. Во всех перечисленных сценариях использования необходима надежная система сбора и хранения данных пользователей, обладающая высокой пропускной способностью, гибкой масштабируемостью и отказоустойчивостью.

# 2. КОМАНДА

## 2.1. Командные роли и распределение ответственности

|  |  |
| --- | --- |
| Сафонов Николай | Разработчик, тестировщик, девопс |
| Чертанов Денис | Разработчик, тестировщик, девопс |

## 2.2. Матрица ответственности

Table

Description automatically generated

# 3. УПРАВЛЕНИЕ ПРОЕКТОМ

## 3.1. Метрики проекта

* Нагрузка на запись новых событий — 8 000 RPS;
* Нагрузка на чтение — 250 RPS;
* Доступность сервиса — 99,9 %.

## 3.2. Диаграмма Ганта

Graphical user interface, application

Description automatically generated

## 3.3. Использование инструментов или систем управления проектом

В качестве репозитория системы контроля версий использовался Github. Для ведения бэклога задача использовался встроенная в Github система с проектами. Для ведения ресурсов проекта использовался Notion.

# 4. ПРЕДМЕТНАЯ ОБЛАСТЬ И ОПИСАНИЕ ПРОДУКТА

## 4.1. Постановка проблемы

В рамках проекта необходимо решить проблему эффективного сбора и хранения потока событий пользователей под высокой нагрузкой. Для решения инженерной задачи необходимо предложить набор компонентов, который будет удовлетворять требованиям системы.

## 4.2. Анализ предметной области, рамки и границы проекта

Предметная область - системы для сбора и анализа данных пользователей. По данным компании IDC за 2021 год общие расходы по миру на системы для обработки больших данных и аналитики составили $215,7 млрд, что по сравнению с 2020 годом составляет прирост в 10.1%. В период с 2021 по 2025 год прогнозируемый годовой прирост размера этого рынка составит 12.8%, что делает разработки в этой области актуальными и перспективными.

## 4.3. Назначение приложения

Назначение сервиса - сохранять и отдавать под высокой нагрузкой историю просмотров пользователя. Также система должна предоставлять инструмент для анализа данных рекомендательным отделом.

# 5. ОПИСАНИЕ ПОТЕНЦИАЛЬНЫХ ПОЛЬЗОВАТЕЛЕЙ

## 5.1. Цель заказчика и ключевые стейкхолдеры

Цель заказчика - сохранять и анализировать историю просмотров пользователей.

## 5.2. План управления коммуникациями

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Предмет коммуникации | Цель | Дата | Средство коммуникации | Ответственный | Участники |
| Собеседование | Попасть на проект | 21.10.21 | Zoom | Чертанов Д.А. | Чертанов Д.А.,  Сафонов Н.Ю. |
| Прототип архитектуры | Согласовать прототип архитектуры | 04.11.22 | Zoom | Чертанов Д.А. | Чертанов Д.А.,  Сафонов Н.Ю. |
| Архитектура | Согласовать архитектуру | 17.02.22 | Zoom | Чертанов Д.А. | Чертанов Д.А.,  Сафонов Н.Ю. |
| Тестирование | Приёмка результатов тестирования | 23.03.22 | Zoom | Чертанов Д.А. | Чертанов Д.А.,  Сафонов Н.Ю. |

# 6. АНАЛИЗ КОНКУРЕНТОВ

**Ivi**

Сервис Ivi использует собственную систему сбора и хранения действий пользователей. Цель их системы - предоставление инструмента анализа данных для аналитиков. Для этих целей используется Clickhouse. Такая система не подходит для решения нашей задачи, поскольку она спроектирована без учета сценария чтения истории пользователя по ключу под высокой нагрузкой. Clickhouse хорошо подходит для выполнения аналитических запросов по большому объему данных, но по результатам наших измерений не очень хорошо справляется с чтением большого количества широких строк с низкой задержкой из-за того, что хранит данные по колонкам.

Diagram

Description automatically generated

**Spotify**

Сервис Spotify собирает поток событий пользователей с помощью своей системы доставки событий. Их система использует кластер Hadoop над HDFS для обработки последних поступивших событий. Такая система не подходит для решения нашей задачи. Hadoop может использоваться для оффлайн аналитики с использованием таких инструментов как например HQL, но не подходит для сценария получения истории пользователя по ключу, который является критическим для решения поставленной задачи.

Diagram

Description automatically generated

**Netflix**

Сервис Netflix составляет персональные рекомендации. Для этого сервис собирает события пользователей и обрабатывает их в своей системе. В архитектуре системы выделяются два сценария работы с данными пользователей - онлайн и оффлайн вычисления. При проектировании системы учитывалось использование данных в рекомендательных алгоритмах на основе машинного обучения, в связи с чем архитектура состоит из большого количества компонентов, которые избыточны для решения нашей задачи.

Diagram

Description automatically generated

# 7. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1. Возможность отправить событие в систему с помощью POST запроса;
2. Возможность аутентификации с помощью JWT;
3. Возможность осуществлять SQL запросы по всем событиям.

# 8. ДРУГИЕ ТРЕБОВАНИЯ К ПРОДУКТУ (НЕФУНКЦИОНАЛЬНЫЕ)

1. Нагрузка на запись новых событий — 8 000 RPS;
2. Нагрузка на чтение — 250 RPS;
3. Доступность сервиса — 99,9 %;
4. Должны храниться все события за все время.

# 9. ОПИСАНИЕ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ И ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

Система разбита на три основных слоя:

1. Слой, отвечающий за сбор событий. Компонент состоит из группы одинаковых сервисов. Основная задача каждого сервиса принять запрос с описанием события, провалидировать формат и содержание запроса и быстро переложить запрос в персистентное хранилище. Источники событий отправляют события в один из сервисов этого компонента.
2. Шина событий. Этот компонент представляет из себя поток событий, из которого могут читать все подсистемы, которым требуется информация о поступающий событиях. Задачей этого компонента является временное хранение событий (порядка нескольких дней) и предоставление доступа к свежим событиям другим подсистемам.
3. Слой хранения событий. Задачей этого слоя является перманентное хранение всей истории событий и предоставление sql-доступа клиентам и аналитикам или системам, которым требуется аналитика по событиям.

Graphical user interface

Description automatically generated

**Описание слоя сбора событий**

Для сбора событий используется один или несколько сервисов, написанных на языке Java на фреймворке Spring с использованием сервера Embedded Jetty для обработки запросов.

**Описание слоя транспортировки**

Для временного хранения событий и асинхронной отправки в слой хранения используется Kafka. Kafka широко используется в микросервисных архитектурах. Разработано большое количество коннекторов для перенаправления сообщений из Kafka в различные хранилища, благодаря чему ее очень просто встроить в систему. Также для Kafka существует клиент на Java поддерживающий асинхронную запись. Поэтому было принято решение использовать Kafka для транспортировки событий в хранилища.

**Описание слоя хранения**

Слой хранения разбит на два компонента для двух сценариев работы:

1. Выполнение аналитических запросов;
2. Быстрое получение истории событий пользователей.

Для аналитических запросов используется Clickhouse. Clickhouse специально разрабатывался под подобные сценарии работы и заточен под высокую пропускную способность аналитических запросов на большом массиве данных.

Хранилища должны поддерживать высокую пропускную способность на запись. Для второго сценария работы также необходима высокая скорость чтения массива данных по ключу. Для удовлетворения этих двух критериев выбор делался среди хранилищ на основе LSM деревьев. Среди готовых решений таких как HBase, Cassandra, ScyllaDB, InfluxDB, RocksDB с учетом других ограничений было принято решение использовать Cassandra.

# 10. ПЕРВОНАЧАЛЬНЫЙ ПРОТОТИП

Diagram

Description automatically generated

Изначально предполагалось, что один кластер Clickhouse сможет справиться со всеми сценариями работы, включая аналитические запросы и запросы на получение истории пользователя. В результате первичного тестирования выяснилось, что Clickhouse плохо подходит для второго сценария использования из-за относительно высокого показателя read amplification связанного с колоночным хранением строк в таблице. Поэтому для обеспечения второго сценария работы в систему добавлена Cassandra.

# 11. ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

1. Docker - Для контейнеризации сервисов и удобства развертывания;
2. Java - Для написания ресивер сервиса и вспомогательных инструментов;
3. Spring - Для написания ресивер сервиса;
4. Jetty - Веб сервер для ресивер сервиса;
5. Cassandra - Для хранения событий пользователей и обеспечения выполнения сценария чтения событий пользователя по ключу;
6. Kafka - Для асинхронной записи событий в хранилища;
7. Kafka connect - Для подключения Kafka к Cassandra;
8. Clickhouse - Для хранения событий пользователей и обеспечения выполнения аналитических запросов по событиям;
9. Prometheus - Для хранения метрик сервисов;
10. Grafana - Для визуализации метрик сервисов;
11. Yandex Tank - Для проведения нагрузочного тестирования;
12. h2load - Для проведения стресс тестирования;
13. Swagger - Для документации API.

# 12. ПРЕДЛОЖЕНИЯ ПО МОНЕТИЗАЦИИ/КОММЕРЦИАЛИЗАЦИИ

1. Построение рекомендательных систем поверх аналитического хранилища;
2. Использование собираемых данных аналитиками;
3. Продажа персонализированной рекламы.

# 13. БУДУЩЕЕ РАЗВИТИЕ ПРОЕКТА

1. Автоматизация части работы аналитиков (Автоматическое составление отчетов, срезов, метрик и прочего);
2. Интеграция системы с другими информационными системами для расширенной аналитики (Добавление внешних словарей, агрегаций и т.п.).

# 16. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## 16.1. Развертывание системы

Система состоит из множества несвязанных компонентов. Для удобства развертывания все компоненты контейнеризованы с использованием Docker. Запуск компонентов осуществляется в следующей последовательности:

1. Zookeeper
2. Kafka
3. Cassandra
4. Kafka Connect
5. Clickhouse
6. Receiver
7. Prometheus
8. Grafana

### 16.1.1. Zookeeper

1. Прописать переменные окружения в файле **zookeeper/env.bash**;
2. Выполнить команду **bash zookeeper/start-zookeeper.bash**.

### 16.1.2. Kafka

1. Прописать переменные окружения в файле **kafka/kafka/env.bash**;
2. Выполнить команду **bash kafka/kafka/start-kafka.bash**.

### 16.1.3. Cassandra

1. Прописать переменные окружения в файле **cassandra/cassandra/env.bash**. Если нужно запустить кластер, то нужно выбрать какая нода будет первоначальным сидом. В файле **env.bash** для сида в переменной **CASSANDRA\_SEEDS** указывается ip адрес ноды на которой будет запущен сид. В остальных нодах также указывается ip адрес сида. Сид должен быть запущен до остальных нод. После запуска всех нод кластера кассандры можно менять адреса сидов через команду **docker exec**;
2. Ноды запускаются командной **bash cassandra/cassandra/start-cassandra.bash**.

### 16.1.4. Kafka connect

1. Прописать переменные окружения в файле **kafka/kafka-connect/env.bash**;
2. Запускаем выполнением команды **bash kafka/kafka-connect/start-kafka-connect.bash**;
3. Для добавления коннектора Cassandra выполняется команда **bash kafka/kafka-connect/add-cassandra-sink-connector.bash**.

### 16.1.5. Receiver

1. Прописать переменные окружения в файле **receiver-service/env.bash**;
2. Выполнить команду **bash receiver-service/start-receiver.bash**.

### 16.1.6. Clickhouse

1. Прописать переменные окружения в файле **clickhouse/env.bash**;
2. Выполнить команду **bash clickhouse/start-clickhouse.bash**.

## 16.2. Запись событий

Запись события осуществляется POST запросом в receiver сервис:

curl -X POST http://hostname:8081/api/user-event/save -d \  
'{  
 "user\_id" : 1,  
 "video\_id" : 1,  
 "event\_type" : 1,  
 "event\_time" : 1  
}'

## 16.3. Чтение событий

Чтение событий осуществляется напрямую из Cassandra. Например, можно прочитать события пользователя с помощью языка cql из клиента cqlsh:

select \*   
from kion.user\_event  
where user\_id = 1  
limit 100;

# 17. СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

# x

|  |  |
| --- | --- |
| 1. | The DataStax Blog [Электронный ресурс] // DataStax: [сайт]. [2015]. URL: https:/​/​www.datastax.com/​blog/​advanced-time-series-data-modelling (дата обращения: 12.02.2022). |
| 2. | Spotify’s Event Delivery – The Road to the Cloud (Part I) [Электронный ресурс] // engineering.atspotify: [сайт]. [2016]. URL: https:/​/​engineering.atspotify.com/​2016/​02/​spotifys-event-delivery-the-road-to-the-cloud-part-i/ (дата обращения: 10.02.2022). |
| 3. | Spotify’s Event Delivery – The Road to the Cloud (Part II) [Электронный ресурс] // engineering.atspotify: [сайт]. [2016]. URL: https:/​/​engineering.atspotify.com/​2016/​03/​spotifys-event-delivery-the-road-to-the-cloud-part-ii/ (дата обращения: 12.02.2022). |
| 4. | Spotify’s Event Delivery – The Road to the Cloud (Part III) [Электронный ресурс] // engineering.atspotify: [сайт]. [2016]. URL: https:/​/​engineering.atspotify.com/​2016/​03/​spotifys-event-delivery-the-road-to-the-cloud-part-iii/ (дата обращения: 13.02.2022). |
| 5. | Как мы в ivi переписывали etl: Flink+Kafka+ClickHouse [Электронный ресурс] // Хабр: [сайт]. [2018]. URL: https:/​/​habr.com/​ru/​company/​ivi/​blog/​347408/ (дата обращения: 15.02.2022). |
| 6. | System Architectures for Personalization and Recommendation [Электронный ресурс] // netflixtechblog: [сайт]. [2013]. URL: https:/​/​netflixtechblog.com/​system-architectures-for-personalization-and-recommendation-e081aa94b5d8 (дата обращения: 16.02.2022). |
| 7. | Processing billions of events/day [Электронный ресурс] // Plumbr: [сайт]. [2017]. URL: https:/​/​plumbr.io/​blog/​programming/​processing-billions-of-eventsday (дата обращения: 18.02.2022). |
| 8. | Storing and querying trillions of events [Электронный ресурс] // Plumbr: [сайт]. [2017]. URL: https:/​/​plumbr.io/​blog/​programming/​storing-and-querying-trillions-of-events (дата обращения: 17.02.2022). |

x