Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Дисциплина: Проектная документация **Техническое задание**

**Работу выполнил студент группы P34131:**Чжоу Хунсян

**Преподаватель:**Маркина Татьяна Анатольевна

2023 г.

г. Санкт-Петербург

Содержание

[**Техническое задание**](#_dk1kbp1ccyix) **3**

[Наименование](#_xitgw0xogbwx) 3

[Назначение](#_u79tx7nqecrn) 3

[Основания для разработки](#_t8v0s9ljaeyz) 3

[Функции](#_vjlpjmaqtvwz) 4

[Структура](#_llqtdqug7x3f) 4

[Обработчики REST API](#_l12cetot5v0) 4

[Основные сервисы](#_80efdoc333zt) 5

[Ключевые компоненты хранилища](#_txz7c8c89sk) 5

[Сеть обработки данных (Spark)](#_bfvr4dsc1cuc) 5

[Пользовательский интерфейс](#_g83j4dmr3owz) 6

[Надежность, безопасность, условия эксплуатации](#_c2hichv407vn) 6

[Документация](#_ndlikz8lomjx) 7

[Стадии и этапы разработки](#_sxuzru76j538) 7

[Порядок контроля и приема](#_a7j3sdvlovlc) 7

# **Техническое задание**

## **Наименование**

Разработка предложения для детекции транспортных потоков с помощью компьютерного зрения

## **Назначение**

Разрабатываемая система предназначена для автоматического анализа транспортных потоков на основе видеоданных с дорожных камер. Система должна определять интенсивность движения, классифицировать транспортные средства и передавать полученные данные для адаптивного управления светофорными объектами.

## **Основания для разработки**

Документами, на основании которых производится разработка, служат:

1. Документация InfluxDB [Электронный ресурс] – URL: <https://docs.influxdata.com/influxdb/>
2. Документация MariaDB [Электронный ресурс] – URL: <https://mariadb.com/kb/en/documentation/>
3. Документация Spark [Электронный ресурс] – URL: <https://spark.apache.org/docs/latest/>
4. Документация ReactJS [Электронный ресурс] – URL: <https://reactjs.org/docs/getting-started.html>
5. Shivakumar R., Goniwada. Cloud Native Architecture and Design: A Handbook for Modern Day Architecture and Design with Enterprise-Grade Examples. – М.: Apress, 2021 г.

## **Функции**

Платформа будет иметь следующие функции:

* Прием и хранение данных: sdLitica сможет принимать и сохранять данные временных рядов в Influxdb тремя способами:
  + через CSV-файлы;
  + как входящий поток данных в реальном времени используя веб-сокеты;
  + через Json TimeSeries формат отправляемых со сторонних серверов по REST API.
* Извлечение данных: Пользователи смогут извлекать данные временных рядов из Influxdb для анализа.
* Анализ данных: sdLitica будет использовать Spark для анализа данных, предоставляя пользователям различные статистические алгоритмы и алгоритмы машинного обучения для анализа данных временных рядов.
* Визуализация данных: Программное обеспечение предоставит пользователям ряд возможностей визуализации, включая диаграммы, графики и другие визуализации.
* Управление метаданными: sdLitica будет использовать MariaDB для хранения метаданных, связанных с пользователем и данными о временных рядов, и управления ими.

## **Структура**

### **Обработчики REST API**

Интерфейс управления REST API - обработчики и службы, которые предоставляют возможность выполнять операции управления, такие как регистрация пользователя, управление профилем, авторизация (получение токена) и так далее.

REST API Analytics Front-end - обработчики и сервисы, которые предоставляют возможность управлять аналитическими работами для текущего пользователя. Этот API позволит просматривать историю конкурирующих запросов с уважаемыми результатами и отслеживать ход текущих работ. API предоставляет возможность запускать вычисление выбранных видов аналитических вычислений для загруженных данных.

REST API Data-Push-интерфейсные обработчики и сервисы, позволяющие конечному пользователю загружать (в различных форматах, таких как CSV или JSON) или передавать данные временных рядов в систему в реальном времени.

### **Основные сервисы**

Ядро платформы - ключевые службы управления, используемые для обработки управленческих данных в системе, таких как информация о пользователях, метаданные загруженных пользовательских данных (временные ряды), метаданные для результатов аналитики и так далее.

Ядро управления аналитикой - набор сервисов platfrom для обработки аналитических запросов и получения результатов аналитики для вошедшего в систему пользователя.

Data-Push Core - сервисы для преобразования данных из пользовательских форматов в общесистемный формат структур временных рядов, которые будут храниться в централизованном хранилище временных рядов.

### **Ключевые компоненты хранилища**

База данных SQL - используется для общего управления информацией и метаданными о загруженных пользовательских данных (временных рядах) и обработки запросов аналитики.

Хранилище NoSQL - используется для хранения бесплатных результатов выполнения аналитики. Поскольку разные виды выполнения могут приводить к разным результатам, хранилище NoSQL в свободной форме обеспечивает большую гибкость для их сохранения.

База данных временных рядов - это очень эффективное хранилище данных временных рядов. В этом хранилище такого рода данные могут храниться очень эффективным способом (с точки зрения потребления места) и могут быть легко найдены и быстро извлечены.

### **Сеть обработки данных (Spark)**

Сеть обработки данных - это распределенная система, ориентированная на распределенную обработку аналитических запросов к данным конечного пользователя. Это группа исполнителей, подобная Дженкинсу, которая управляется и синхронизируется с помощью эффективного кластера RabbitMQ на основе сообщений.

## **Пользовательский интерфейс**

Пользовательский интерфейс sdLitica будет состоять из следующих компонентов:

* Main menu: предоставит информацию о том, как пользоваться системой, и позволит перейти к различным разделам.
* Profile: У пользователей будет иметь свой собственный профиль, в котором они смогут управлять своими личными данными.
* Dashboards: пользователи смогут настроить свои собственные информационные панели для отображения данных временных рядов наиболее удобным для них способом.
* My buckets: пользователи смогут создавать хранилища для временных рядов и настраивать их, а также добавлять сами данные временных рядов и управлять ими.
* Load data: пользователи смогут загружать данные временных рядов, используя 3 метода: File, Streams через WebSocket, REST API - Json TimeSeries (JTS).
* Data explorer: пользователи смогут строить различные виды графиков и проводить аналитику над своими временными рядами, впоследствии результаты можно будет добавлять в свои группы раздела Dashboards.
* History: пользователь сможет отслеживать свою историю действий с временными рядами.

## **Надежность, безопасность, условия эксплуатации**

Надежность: sdLitica будет спроектирована таким образом, чтобы быть высоконадежной, гарантируя точное и эффективное хранение и анализ данных временных рядов.

Безопасность: платформа будет разработана с учетом соображений безопасности, гарантируя защиту конфиденциальных данных от несанкционированного доступа.

Условия эксплуатации: платформа будет спроектирована для работы в различных условиях, что гарантирует ее возможность использования в различных отраслях промышленности и областях применения.

Другие важные требования: платформа будет разработано таким образом, чтобы быть масштабируемой, позволяя пользователям легко расширять систему по мере необходимости.

## **Документация**

Будет разработана следующая документация:

* Руководство пользователя: В руководстве пользователя содержатся инструкции по использованию sdLitica, включая навигацию по пользовательскому интерфейсу, загрузку данных временных рядов, настройку информационных панелей и выполнение анализа.
* Конфигурация: В руководстве по настройке будут приведены инструкции о том, как настроить sdLitica для различных вариантов использования, включая изменение пользовательского интерфейса, настройку элементов управления доступом и настройку функций анализа данных.

## **Стадии и этапы разработки**

Разработка sdLitica будет проходить в следующие этапы:

* Дизайн и архитектура: будет выполнена разработка архитектуры платформы и компонентов, включая схему базы данных, пользовательский интерфейс и функции анализа данных.
* Реализация и кодирование: разрабатывается платформа, используя C# в качестве языка программирования и React JS для пользовательского интерфейса.
* Тестирование и гарантия качества: будет протестирована платформа, чтобы убедиться, что оно соответствует требованиям и не содержит ошибок.

Разработка будет вестись для каждого типа сервиса отдельно. Общие части примеров будут использованы в следующих.

## **Порядок контроля и приема**

* Контрольное тестирование: платформа будет протестирована, чтобы убедиться, что оно соответствует техническим требованиям и спецификациям.
* Оформление документации.
* Сдача заказчику (защита "ВКР").