ООО «Отус онлайн-образование»

Курс «Scala-разработчик»

Сафронов Валерий Евгеньевич

Сервис управления наливом и взвешиванием нептепродутов

ВЫПУСКНОЙ ДИПЛОМНЫЙ ПРОЕКТ

Выполнил: студент группы

[scala-dev-mooc-2023-09](https://github.com/ScalaOtus/scala-dev-mooc-2023-09)

|  |  |
| --- | --- |
|  | Руководитель курса   \_\_\_\_\_\_\_\_/Воронец Алексей/  (подпись)     (И.О. Фамилия) |

«\_\_» апреля 2024 г.

КУРСК – 2024

СОДЕРЖАНИЕ

ГЛАВА I. Цели проекта.

ГЛАВА II. Планируемый функционал и компоненты.

ГЛАВА III. Используемые технологии.

ГЛАВА |V. Реализованный функционал

ГЛАВА V. Архитектура системы.

ГЛАВА VI. Детали реализации.

ГЛАВА VII. Функционал тестирования. Выполнение теста.

ГЛАВА VIII. Выводы.

1. **Цели проекта.**

Цель проекта - разработать фуллстэк приложение (бэкенд и фронтенд) реализующее часть функционала системы управления наливом и взвешиванием нефтепродуктов.

Приложение должно выполнять следующий функционал:

-принимать восходящий поток данных от оборудования автоматизированных весовых платформ (железнодорожной и автомобильных) по протоколу TCP;

- отображать состояние платформ пользователю в веб-интерфейсе браузера (взаимодействи браузера и бэкенда по протоколам Http и WebSocket);

- обрабатывать алгоритм взвешивания с применением карт доступа;

- отдавать поток событий в топики KAFKA;

- сохранять поток событий в таблицах базы данных;

- предоставлять доступ по протоколу Http к сохраненным в БД событиям для внешних сервисов;

- логировать события обработки в журнале;

**II. Планируемый функционал и компоненты.**

В проекте планировалось реализовать слабосвязанные компоненты разертываемые в единую работоспособную систему используя контейнер инверсии управления и внедрения зависимостей (IoK контейнер).

Весь функционал прокета разбивается на следующие крупные блоки.

1. Бэкенд
   1. Бизнес-модули
   2. Модули тестирования
2. Фронтенд (SPA приложение для браузера)
3. Вспомогательный код тестирования (скрипты)

Запланированные к реализации компоненты и сервисы (по блокам)

1. a)

* + 1. Конфигурация бэкенда в формате HOCON и сервисы (классы) чтения конфигурации.
    2. Модуль контейнера инверсии управления и внедрения зависимостей- описывающий создание экземпляров компонентов и сервисов и биндинг экземпляров на классы интерфейсовы.
    3. Модели текстовых протоколов в соответствии с которыми данные поступают от оборудования - паттерны регулярных выражений описывающих протокол и механизм получать эти паттерны по имени.
    4. Экстракторы протоколов - scala-экстракторы позволяющие парсить строки протоколов и преобразовывать их в ADT - кейс классы.
    5. Web модели - включают кейс классы данных для сериализации и имплиситные Json-врайтеры - предоставляют функционал Json-сериализации в компонентах, где такая сериализация требуется.
    6. DB модели - кейс классы на которые отображаются данные таблиц базы данных.
    7. DB схема - классы описывающие схему базы данных.
    8. Классы TCP серверов.
    9. Классы WebSocket сервера.
    10. Web контроллер для работы по Http - предоставляет асинхронные http - обработчики (на основе Future) и потоковые обработчики на основе стримов.
    11. Web контроллер для акцепта соединений по протоколу Web-socket.
    12. Актор WebSocket - сервера
    13. HTTP роутер.
    14. Сервис DB слоя - содержит логику запросов на DSL FRM (функционально реляционного мэппинга). Предоставляет интерфейс на основе фьючеров и стримов.
    15. Глобальные хранилища.
    16. Сервисы бизнеслогики:
        1. Диспетчеры
        2. Парсеры протоколов
        3. Стейт-машины
        4. Менеджер диспетчеров
    17. Классы TCP - клиентов
    18. Менеджер TCP - клиентов
    19. Классы UDP сервера: UDP сервер, менеджер UDP сервера, NetWorker

1. SPA приложение для браузера
2. Набор BASH скриптов организующих длительный тест имитирующий потоки данных от оборуддования.

**III. Используемые технологии.**

Технологии бэкенда:

1. Play Framework - базовый связующий фреймворк
2. Конфигурация в формате HOCON. Configuration API.
3. Google Guice - контейнер инверсии управления и внедрения зависимостей.
4. Асинхронный HTTP, потоковый HTTP.
5. Akka
   1. классические акторы - используются для реализации сетевого функционала - TCP серверы, TCP клиенты, UDP сервер, WebSocket сервер;
   2. типизированные акторы - используются для реализации сервисов бизнес-логики - диспетчеров, парсеров протоколов, стейт-машин;
6. Akka Streams - используются для следующих задач
   1. сериализация и отправка сообщений в фронтенд по протоколу WebSocket;
   2. сериализация и отправка сообщений в топики KAFKA;
   3. организации потокового взаимодействия базы данных и вэб-контроллера;
   4. оптимизация батч режима вставки данных в таблицы БД - разбиение на чанки и параллелизм;
7. Play Json - применяются различные способы сериализации ADT в т.ч. имплиситные Writes.
8. Scala Slick - библиотека функционально-реляционного мэппинга. API DB-слоя используют как потоки так и фьючеры.
9. Протокол WebSocket - может использоваться в качестве альтернативы HTTP для взаимодействия бэкенда и фронтенда.
10. Использование scala-экстракторов для парсинга протоколов.

Технологии фронтенда:

1. Язык разработки - TypeScript
2. Фреймворк мультиплатформенной и кроссплатформенной сборки c библиотекой компонентов - Quasar Framework.
3. Внутренний реактивный web фрэймворк - Vue JS 3.
4. Библиотека управления состоянием - Pinia
5. Сеть: AJAX библиотека Axios, WebSocket библиотека reconnecting-websocket.
6. Сборка и траспилирование: webpack, babel.

**IV. Реализованный функционал.**

Весь функционал запланированый в пунктах I,II реализован.

**V. Архитектура системы.**

Платформа ЖД весов

Микроконтроллерный блок управления

Микроконтроллерный блок управления

Платформа автомобильных весов 2

Датчик периметра

Датчик периметра

**Табло бегущей строки**

RFID

RFID

Платформа автомобильных весов 1

Датчик периметра

Датчик периметра

**Табло бегущей строки**

RFID

RFID

Платформа автомобильных весов 3

Датчик периметра

Датчик периметра

**Табло бегущей строки**

RFID

RFID

Микроконтроллерный блок управления

Микроконтроллерный блок управления

Локальная сеть / Интернет

TCP

TCP

TCP

Рисунок 1 - физические объекты системы

TCP

Скрипты имитации потока данных от оборудования

Сеть

UDP

UDP

TCP

Бэкенд

Модуль тестирования, UDP сервер, TCP клиенты

TCP серверы

TCP

Инфраструктура имитации оборудования

Фронтенд

HTTP

Web Socket

Рисунок 2 - общая схема компонентов программной системы и коммуникаций

СУБД

Конфигурация

Модуль Google Guice

TCPStorage Singleton

параметры конфигурации

Конффигурация TCP серверов

TCPServerBuilder Singleton

ActorRef созданного TCP сервера помещается в хранилище

TCP Server 1

TCP Server 3

TCP Server 4

TCP Server 2

Handler

Handler

Handler

Handler

ActorRef

ActorRef

ActorRef

ActorRef

TCP-серверы и их хэндлеры - типизированные акторы