Создание нового вертикального решения на базе существующего

Содержание

1 ОБЩАЯ ИНФОРМАЦИЯ

1.1 Термины и сокращения

Сокращение/Термин	Наименование/Определение
Агент	Программный объект, проводящий сбор данных по заданным алгоритмам и их передачу в центр сбора данных, настраиваемый локально или удаленно из центра сбора данных и коммуницирующий с другими объектами или центром сбора данных
Администратор	В настоящем документе: лицо, наделённое правами для осуществления деятельности в административной части ИС
АС	Автоматизированная система
Браузер, веб-браузер	Программное обеспечение на компьютере или мобильном устройстве пользователя, предназначенное для просмотра веб-страниц, содержания веб-документов, управления веб-приложениями, размещёнными в Интернете
Веб-интерфейс	Совокупность средств, при помощи которых пользователь взаимодействует с сайтом или любым другим приложением через веб-браузер
Веб-приложение	Клиент-серверное приложение, в котором клиентом выступает веб-браузер, а сервером — веб-сервер
Веб-сервер	Сервер (программное обеспечение), принимающий запросы от клиентов, обычно веб-браузеров, и выдающий им ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными
ВИС КНО	Ведомственная информационная система автоматизации контрольно-надзорной деятельности контрольно-надзорных органов
ГИС ТОР КНД	Государственная информационная система «Типовое облачное решение по автоматизации контрольно-надзорной деятельности
ГОСТ	Государственный стандарт
Датчик	Любое устройство, подключаемое к Системе, передающее измерения какого-либо параметра в Систему. В терминах прикладных сервисов дистанционного контроля в рамках экологического надзора в области охраны атмосферного воздуха: Газоанализатор, или комплекс газоанализаторов, которые передают данные, как правило через специализированный контроллер
ЕИС КНД	Единая информационная среда контрольно-надзорной деятельности
ЕСИА	Федеральная государственная информационная система «Единая система идентификации и аутентификации в инфраструктуре, обеспечивающей информационно-технологическое взаимодействие информационных систем, используемых для предоставления государственных и муниципальных услуг в электронной форме»

Сокращение/Термин	Наименование/Определение
Интерфейс	Совокупность возможностей, средств, способов, методов и правил взаимодействия двух объектов, в частности человека с системой, устройством или программой для обмена информацией между ними
Инцидент	В терминах прикладных сервисов дистанционного контроля в рамках экологического надзора в области охраны атмосферного воздуха: Сущность, агрегирующая в себе последовательность однотипных событий. Инцидент активен, пока продолжают поступать однотипные события. При появлении инцидента ответственный сотрудник получает соответствующее уведомление
ИС	Информационная система
ИС ПСД, Система	информационная система «Единая государственная платформа сбора данных, промышленного интернета вещей и инструментов анализа объективных данных о наблюдаемых объектах»
Инстанс	(англ. Instance) – Экземпляр чего-либо
КИА	Контрольно-измерительная аппаратура, обобщенное название различных устройств и комплексов автоматического и автоматизированного сбора величин измеряемых параметров и средств связи для передачи для передачи этих величин на сервера Системы
Клиент	В контексте сетевой архитектуры «клиент-сервер»: программное обеспечение, являющееся заказчиком услуг сервера (поставщика услуг)
Клиент-сервер	Вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами
КНД	Контрольно-надзорная деятельность
КНО	Контрольно-надзорный орган
Контент	Информационное наполнение веб-страницы, сайта, экрана мобильного приложения
Кроп	Небольшое изображение, полученное путем нарезки исходного большого изображения по определенным правилам
Общемировое время	Время в UTC(фр.: Temps Universel Coordonné) стандарте
ОС	Операционная система
ОЭ	Опытная эксплуатация
ПДК	Предельно допустимая концентрация
Платформа	Набор базовых сервисов, включающий в себя: подсистему работы с источниками данных, подсистему работы с данными и подсистему хранения данных
Профиль сообщения	Совокупность данных, описывающих сообщение

Сокращение/Термин	Наименование/Определение
Регион мониторинга	Географический регион, город, район, в рамках которого технически и организационно осуществляется мониторинг контролируемых параметров
Реестр Российского ПО	Единый реестр российских программ для электронных вычислительных машин и баз данных
Сервер	В контексте сетевой архитектуры «клиент-сервер»: программное обеспечение, являющееся поставщиком услуг заказчикам (клиентам)
Сервис мониторинга воздуха	Прикладной сервис дистанционного экологического надзора в области охраны атмосферного воздуха, являющийся частью единой государственной платформы сбора данных, промышленного интернета вещей и инструментов анализа объективных данных о наблюдаемых объектах в составе платформы исполнения государственных функций
Сервис мониторинга объектов культурного наследия	Прикладной сервис сбора и анализа информации в рамках государственного надзора за состоянием, содержанием, сохранением, использованием, популяризацией и государственной охраной объектов культурного наследия, являющийся частью единой государственной платформы сбора данных, промышленного интернета вещей и инструментов анализа объективных данных о наблюдаемых объектах в составе платформы исполнения государственных функций
Сервис контроля вырубок лесов и использования с/х земель	Сервис контроля несанкционированных вырубок леса и использования земель сельскохозяйственного назначения с использованием фотоаналитики на основе машинного обучения единой государственной платформы сбора данных, промышленного интернета вещей и инструментов анализа объективных данных о наблюдаемых объектах в составе платформы исполнения государственных функций
Сигнальная сеть КИА	Сеть из датчиков и средств передачи данных. Сеть формирует равномерное покрытие территории, с определенным (регулярным) шагом между КИА. Может употребляться в формулировке «Сигнальная сеть», «Сигнальная сеть датчиков», «сеть датчиков»
Событие	В терминах прикладных сервисов дистанционного контроля в рамках экологического надзора в области охраны атмосферного воздуха: Фиксация факта превышения концентрации или достижения иных условий с обязательным уведомлением ответственных сотрудников
СУБД	Система управления базами данных
ТЗ	Техническое задание
Тенант	(англ. Tenant - арендатор) Экземпляр виртуального приложения в мультиарендной архитектуре с ограниченным доступом только до своей конфигурации и своего набора данных
Фреймворк	Программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта. «Фреймворк» отличается от понятия библиотеки тем, что библиотека может быть

Сокращение/Термин	Наименование/Определение
	использована в программном продукте просто как набор подпрограмм близкой функциональности, не влияя на архитектуру программного продукта и не накладывая на неё никаких ограничений. В то время как «фреймворк» диктует правила построения архитектуры приложения, задавая на начальном этапе разработки поведение по умолчанию — «каркас», который нужно будет расширять и изменять, согласно указанным требованиям. Также, в отличие от библиотеки, которая объединяет в себе набор близкой функциональности, — «фреймворк» может содержать в себе большое число разных по тематике библиотек.
ЧТЗ	Частное техническое задание
API	Application Programming Interface (интерфейс программирования приложений) — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах
JSON	(англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript
Multitenancy	Мультиарендность. В мультиарендной архитектуре программные приложения работают одновременно с несколькими конфигурациями и наборами данных нескольких организаций, а каждая организация-клиент работает со своим экземпляром виртуального приложения (тенантом), видя только свою конфигурацию и свой набор данных
POD	Базовая исполнительная единица приложения в среде Kubernetes - наименьшая и самая простая единица в объектной модели Kubernetes, которая может быть создана и развернута
RBAC	Управление доступом на основе ролей
XML	eXtensible Markup Language (расширяемый язык разметки) — язык для создания структурированных машиночитаемых документов

1.2 Условия реализации новых прикладных сервисов

Для создания нового вертикального решения можно взять за основу уже функционирующее вертикальное решение из состава Информационной системы «Единая государственная Платформа сбора данных, промышленного интернета вещей и инструментов анализа объективных данных о наблюдаемых объектах».

Лица, выполняющие работы по созданию нового или доработке существующего вертикального решения, должны знать принципы работы со следующими технологиями и библиотеками:

- серверная часть:
  - Java 8;

- Maven;
- Spring Boot;
- Hibernate;

- клиентская часть:
  - HTML/CSS/JavaScript;
  - TypeScript;
  - NPM;
  - Angular 8.

Пример реализации прикладного сервиса можно посмотреть в следующих репозиториях:

- https://gitlab.vms.torcloud.digital.gov.ru/gis-tor-knd/iot/iot-solution-backend
- https://gitlab.vms.torcloud.digital.gov.ru/gis-tor-knd/iot/iot-solution-vologda-backend

2 ИНТЕРФЕЙС ВЗАИМОДЕЙСТВИЯ С ПЛАТФОРМОЙ

2.1 Использование метаинформации

Для работы серверной части постоянно требуется метаинформация, хранящаяся в базовой функциональности Платформы. Метаинформация доступна в административной панели Платформы (структура, устройства и др.).

Для уменьшения задержек при постоянном получении частей метаданных реализован кэш-механизм, при котором все метаданные с настраиваемой периодичностью загружаются в серверную часть. Разные виды кэшей доступны как компоненты Spring и могут быть использованы в любом сервисе.

Таблица 1 – Описание кэшей Spring Boot

Название	Описание
`DevicesCache.java`	Кэш статусов устройств (выделяется из кэша структуры)
`DigitalTwinsCache.java`	Кэш устройств (теги устройств выделяются из структуры и для них загружаются данные из эндпоинта базовой функциональности Платформы devices
`SensorsCache.java`	Кэш датчиков (выделяется из кэша структуры)
`TagsCache.java`	Кэш структуры (метаданных)
`DeviceThresholdsExceedStatusCache.java`	Кэш порогов

Структура метаинформации в кэше аналогична структуре в базовой функциональности Платформы. Однако для выполнения функциональности серверной части часто нужно приведение этих данных к виду, удобному для использования в конкретном сервисе. Такие операции сервис выполняет самостоятельно.

2.2 Постоянное хранилище и миграции (изменения схемы базы данных)

В базовой функциональности Платформы хранятся метаинформация и данные измерений, а также обеспечивается поддержка работоспособности при высокой нагрузке, при получении большого объема новых данных и обработке правил для каждого измерения. При срабатывании правила в серверную часть приложения отправляется факт превышения, и уже серверная часть принимает решение, считать ли данный факт инцидентом, или это превышение связано с уже существующим инцидентом.

Инциденты сохраняются в базе данных Postgres. Для приложение важно наличие готовой базы, параметры подключения к которой прописываются в файле `application.yml`, и наличие схемы. Структура таблиц и полей будет создана приложением при первом запуске приложения.

Работы с базой данных осуществляется через JPA API и библиотеку Hibernate, а также при поддержке данных технологий со стороны Spring.

За создание таблиц и полей отвечает Liquibase – специальная библиотека, в которой создание структуры базы описывается в виде шагов-изменений. Список существующих таблиц в базе приведен ниже (Таблица 2).

Таблица 2 – Список таблиц

Название	Описание
`databasechangeloglock`	Таблица для Liquibase миграций
`databasechangelog`	Таблица для Liquibase миграций
`comments`	Комментарии к инцидентам
`pollution_sources`	Список возможных источников для каждого инцидента
`device_infos`	Информация по устройствам
`computationinfo`	Результаты расчетов в Тайфун и Эколог-Город
`incidents`	Таблица с инцидентами

2.3 Прием фактов превышения от базовой функциональности Платформы

При срабатывании правила базовая функциональность Платформы отправляет факт превышения в серверную часть приложения, и уже приложение решает, нужно ли сохранять данный факт превышения как инцидент, или нужно добавить данный факт к уже существующему инциденту. Для этого используется метод контроллера (эндпоинт `/api/v1/event`): `ru.waveaccess.iot.incidentapp.api.EventController#createIncidentFromEvent`.

Логика создания инцидента описана в `ru.waveaccess.iot.incidentapp.service.IncidentService#createNewIncidentOrExtendOld`.

2.4 Запрос оперативных данных от базовой функциональности Платформы

Запрос оперативных данных (за 24 часа) выполняется в хранилище Tarantool, расположенный внутри базовой функциональности Платформы и доступный через промежуточный слой, проверяющий права и преобразующий данные. Доступ осуществляется посредством сервиса TarantoolStorageService и служит для получения значений измерений.

2.5 Запрос исторических данных от базовой функциональности Платформы

Запрос исторических данных выполняется в хранилище Clickhouse, расположенное внутри базовой функциональности Платформы и доступный через промежуточный слой, проверяющий права и преобразующий данные. Доступ осуществляется посредством сервиса `ClickhouseStorageService` и служит для получения значений измерений за определенный период времени.

3 ЛОГИЧЕСКИЕ МОДУЛИ И КОМПОНЕНТЫ ПРИКЛАДНОГО СЕРВИСА

3.1 Общие положения

Для серверной части используется стандартная структура проекта Maven, а также стандартная архитектура Spring Boot.

3.2 Высокоуровневая структура проекта

Серверная часть вертикального решения состоит из общей части (`iot-solution-backend`), которая находится в отдельном репозитории, и части, специфичной для данного вертикального решения и региона, например, `iot-solution-vologda-backend`.

Специфичная часть состоит из родительского проекта, для которого родителем указан `spring-boot-starter-parent`, и вложенного модуля `incident-service`, который и является серверной частью веб-приложения.

3.3 Интерфейс серверной части

Для работы клиентской части веб-приложения используются эндпоинты, которые находятся в контроллерах в пакете `ru/waveaccess/iot/incidentapp/api`.

Контроллеры выполнены в виде компонентов Spring `@RestController`, это позволяет упростить их программирование (Таблица 3).

Таблица 3 – Описание контроллеров

Название	Описание
`AuthController.java`	Получение информации от текущем пользователе
`CommentController.java`	Создание и просмотр комментариев из карточки инцидента
`DebugController.java`	Служебные методы для разработки
`DeviceController.java`	Получение статусов устройств
`EventController.java`	Создание фактов превышения и получение различных данных измерений по фильтрам
`IncidentController.java`	Получение списка инцидентов по различным фильтрам
`ReferenceController.java`	Получение значений справочников
`ReportController.java`	Получение отчетов
`SensorController.java`	Получение обобщенной информации
`SettingController.java`	Получение настроек
`VersionController.java`	Получение версии приложения
`WebAppController.java`	Получение структуры размещения датчиков и другой информации, необходимой для рабочего стола

3.4 Общая часть всех вертикальных решений (`iot-solution-backend`)

Общая часть автоматически с настроенной периодичностью запрашивает метаинформацию и обновляет кэш (локальная версия метаинформации), а также предоставляет

общие классы модели и клиенты для доступа к базовой функциональности Платформы. Метаинформация – структура, представляющая собой иерархический список объектов (справочники, объекты контроля и др.). Такую структуру можно увидеть в административной панели базовой функциональности Платформы.

Общая часть всех вертикальных решений состоит из подпроектов – Maven-модулей (Таблица 4).

Таблица 4 – Состав общей части всех вертикальных решений (`iot-solution-backend`)

Название	Описание
`cache-service`	Классы для автоматической загрузки и предоставления метаинформации.
`common-model`	Общие классы модели.
`common-service`	Общие сервисы.
`platform-client`	Классы модели и клиенты (сервисы) для доступа к базовой функциональности Платформы.

Список клиентов (сервисов) для доступа к функциональности Платформы представлен ниже (Таблица 5).

Таблица 5 – Список клиентов (сервисов)

Название	Описание
`ClickhouseStorageService.java`	Доступ к постоянному хранилищу данных измерений
`DigitalTwinsService.java`	Получение информации об устройствах
`HttpGatewayService.java`	Получение информации об агенте и отправка значений измерений (используется в эмуляторах и агентах)
`RegistryService.java`	Получение метаинформации
`TarantoolStorageService.java`	Получение оперативной информации (значение измерений за 24 часа)
`UserInfoService.java`	Получение информации о пользователе

Клиенты, предназначенные для доступа к функциональности Платформы, являются обертками над Open API интерфейсом Платформы. Подпроекты собираются в виде библиотек в формате jar файлов и доступны для включения в сборку основной серверной части через локальное Maven-хранилище (папка `.m2` в директории текущего пользователя).

3.5 Spring Boot и архитектура проекта

В архитектуре проекта используется стандартная архитектура Spring Boot, а именно разделение кода на модель, конфигурацию, контроллеры, сервисы, репозитории, клиенты и утилиты. Конфигурация, контроллеры, сервисы, репозитории и клиенты являются компонентами (бинами) Spring Boot и связываются друг с другом через внедрение зависимостей, предоставляемое Spring Boot.

Описание основных пакетов Spring Boot представлено ниже (Таблица 6).

Таблица 6 – Основные пакеты Spring Boot

Название	Описание
`api`	Содержит контроллеры для REST запросов
`config`	Содержит конфигурации Spring Boot
`constants`	Значения разных констант
`exception`	Исключения
`model`	Классы модели
`repository`	Репозитории (объекты доступа к данным)
`service`	Сервисы (в виде Spring компонентов)
`utils`	Утилиты

3.6 Аутентификации и авторизация

3.6.1 Обработка запросов от клиентской части

Если аутентификация и авторизация не была выполнена, клиентское приложение осуществляет переход на страницу аутентификации и авторизации МЗИ. После успешного прохождения клиентское приложение получает файл cookies с токеном безопасности, который использует для всех запросов. Прокси МЗИ при вызове эндпоинта проверят наличие необходимых прав, приложению нет необходимости выполнять это самостоятельно.

3.6.2 Выполнение фоновых задач

Для выполнения фоновых задач (например, обновление кэша) серверная часть самостоятельно аутентифицируется и авторизуется в МЗИ, получает токен и использует его для фоновых запросов. При устаревании токена выполняется его автоматическое обновление.

3.7 Конфигурация приложения

Основной файл конфигурации:

`src/main/resources/application.yml`.

В файле конфигурации прописаны основные настройки приложения, которые могут отличаться для различных установок (Таблица 7).

Таблица 7 – Конфигурация настроек приложения

№	Параметр	Пример значения	Комментарий
1	`kc.base-url`		Адрес модуля защиты информации (МЗИ)
2	`kc.client-id`	`apps_3_wa`	ИД клиента
3	`kc.client-secret`	`********`	Секрет
4	`spring.datasource.url`	`jdbc:postgresql://postgres-vol:5432/iot`	Адрес базы данных
5	`spring.datasource.hikari.username`	`postgres`	Имя пользователя базы

№	Параметр	Пример значения	Комментарий
6	spring.datasource.hikari.password	*******	Пароль к базе
7	mail.registry.base-url	http://iotstage.ru/registry/v1	Адрес регистра
8	mail.http-gateway.base-url	http://api.iotstage.ru/http-gateway/v1	Адрес HTTP точки доступа
9	mail.tarantool-storage.base-url	http://iotstage.ru/operational-storage/api/v1	Адрес временного хранилища данных
10	mail.clickhouse-storage.base-url	http://iotstage.ru/long-term-storage/api/v1	Адрес постоянного хранилища данных
11	mail.digital-twins.base-url	http://iotstage.ru/digital-twins/api/v1	Адрес сервиса цифровых двойников
12	login.clientInfoUrl	https://iotstage.ru/identity-manager/api/v1/user_info	Адрес точки получения информации и пользователе
13	login.userInfoUrl	https://iotstage.ru/keycloak/auth/realms/iot/protocol/openid-connect/userinfo	Адрес точки получения информации о пользователе
14	Iot.mail.url	https://culture.iotstage.ru/incident	Адрес приложения для отправки в электронной почте

Дополнительно к основной изменяемой конфигурации из файла `application.yml` в пакете `src/main/java/ru/waveaccess/iot/incidentapp/config` находятся неизменяемые конфигурации Spring Boot (Таблица 8).

Таблица 8 – Неизменяемые конфигурации Spring Boot

Название	Описание
CustomJwtGrantedAuthoritiesConverter.java	Преобразование JWT токена
FeignConfig.java	Конфигурация Feign клиентов
OAuth2SecurityConfiguration.java	Конфигурация разрешений для различных эндпоинтов
OAuth2SecurityConfigurationBase.java	Конфигурация разрешений
RoleMappingConfig.java	Конфигурация настроек безопасности
SwaggerConfig.java	Конфигурация Swagger
WebSocketConfiguration.java	Конфигурация для уведомлений для браузера по веб-сокетам

3.8 Сборка серверной части проекта

а) Собрать общую часть (из в директории общей части) командой:

```
mvn clean install
```

б) Собрать серверную часть конкретного вертикально решения (из в директории конкретного решения) командой:

```
mvn clean package
```

в) Приложение собирается в виде исполняемого jar файла со встроенным контейнером сервлетов Tomcat и всеми зависимостями и может быть запущено командой:

```
java -jar <название jar файла>
```

4 КЛИЕНТСКАЯ ЧАСТЬ ПРИЛОЖЕНИЯ

Клиентская часть является типовым Angular 8 приложением, со сборкой и загрузкой зависимостей через `npm`. Реализацию клиентской части новых вертикальных решений исполнитель может выполнять на базе любого современного фреймворка веб-приложений.

4.1 Структура проекта и сборка

а) Перед сборкой необходимо загрузить зависимости командой:

```
npm install
```

б) Выполнить сборку командой:

```
npm run build
```

в) Зависимости проекта прописаны в файле `package.json`.

Основные директории клиентской части приложения описаны ниже (Таблица 9).

Таблица 9 – Директории клиентской части приложения

Директория	Описание
`src`	Содержит весь исходный код проекта.
`src/app`	Основные сущности фреймворка – модули, компоненты, сервисы, директивы и т.д.
`src/assets`	Статический контент – изображения, шрифты и т.п.
`src/environments`	Конфигурации для различных сред запуска (test, prod …).
`src/libs`	Новые или дополненные необходимой функциональностью библиотеки.
`src/styles`	Глобальные стили, которые применяются ко всему приложению.
`src/app/common`	Конфигурационные файлы и константы.
`src/app/core`	Директивы, модели, сервисы, валидаторы, pipes и пр.
`src/app/interceptors`	Перехватчики http запросов.
`src/app/modules`	Модули приложения.
`src/app/modules/incident`	Модуль для работы с инцидентами и их детальной страницей.
`src/app/modules/registers`	Модуль для работы со справочниками и их детальными страницами.
`src/app/modules/reports`	Модуль для работы с отчетами.
`src/app/modules/sources`	Модуль для работы со страницей рабочего стола.
`src/app/modules/shared`	Общие компоненты; shared модуль может быть импортирован в любой другой модуль.

Приложение использует Lazy Loading – т.е. модули будут загружаться в зависимости от выбранного маршрута, что помогает сократить время загрузки страницы.

Основная маршрутизация описана в файле `src/app/app-routing.module.ts`.

4.2 Аутентификация и авторизация

Если в ответ на запросы клиента приходят ошибки `401 Unauthorized` или `403 Forbidden`, клиент перенаправляет браузер на страницу ввода логина и пароля МЗИ. В обратном редиректе клиент получает токен, который впоследствии использует во всех запросах. Токен добавляется к запросам в http-перехватчике `src/app/interceptors/main.interceptor.ts`.

В перехватчике также происходит обработка кода ответа от сервера.

4.3 Рабочий стол

Компонент, отвечающий за раздел «Рабочий стол»: `src/app/modules/sources/pages/sources-page`.

Главные элементы на рабочем столе – это карта, графики, таблицы и круговые диаграммы. Для отображения карты используется библиотека Leaflet с различными плагинами.

Для кластеризации объектов используется плагин `leaflet.markercluster`, для поиска по адресу – `leaflet-geosearch`.

Основные компоненты рабочего стола описаны ниже (Таблица 10).

Таблица 10 – Основные компоненты рабочего стола

Компонент	Описание
`modules/sources/components/monitored-object-table`	Таблица с данными по объекту мониторинга за 60 минут
`modules/shared/components/device-graph`	График по датчику с отображением текущих данных, среднего, максимального, минимального и пороговых значений. За отображение графиков отвечает библиотека для графического изображения и визуализации данных ngx-echarts
`modules/shared/components/sensor-guage`	Круговая диаграмма с текущими данными датчика. Круговые диаграммы построены на основе библиотеки для создания динамических интерактивных визуализации данных D3.
`modules/shared/components/incident-linear-diagram`	Сводная информация по городу/области.

4.4 Инциденты

В разделе «Инциденты» представлен список инцидентов, разбитый по статусам. Компонент, отвечающий за раздел «Инциденты»: `src/app/modules/incident/pages/incident-page`.

Детальная страница инцидента: `src/app/modules/incident/pages/incident-info-page`.

Основные компоненты детальной страницы инцидента представлены ниже (Таблица 11).

Таблица 11 – Основные компоненты детальной страницы инцидента

Компонент	Описание
`modules/incident/components/incident-comments`	Работа с комментариями: отображение/отправка комментария.
`modules/incident/components/incident-journal`	Журнал инцидента: таблица с показаниями датчиков с момента начала инцидента.
`modules/shared/components/device-graph`	График по датчику с отображением текущих данных: среднего, максимального, минимального и пороговых значений.
`modules/incident/components/incident-map`	Отображение датчика на карте, по которому сработал инцидент.

Для получения данных по инцидентам используется сервис `src/app/core/services/incident.service.ts`.

4.5 Отчеты

В разделе «Отчеты» за каждый тип отчета отвечает отдельный компонент.

Основные компоненты раздела «Отчеты» представлены ниже (Таблица 12).

Таблица 12 – Основные компоненты раздела «Отчеты»

Компонент	Описание
`modules/reports/pages/report-page`	Базовый компонент для формирования конкретных типов отчетов
`modules/reports/components/incidents-report`	Отчет по инцидентам
`modules/reports/components/exceeding-report`	Отчет по фактам превышения
`modules/reports/components/sensor-statistics`	Отчет по статистике по датчикам

Для получения данных для отчетов используется сервис `src/app/core/services/report.service.ts`.

4.6 Справочники

В разделе «Справочники» за каждый справочник отвечает отдельный компонент.

Основные компоненты раздела «Справочники» представлены ниже (Таблица 13).

Таблица 13 – Основные компоненты раздела «Справочники»

Компонент	Описание
`modules/registers/pages/registers-page`	Базовый компонент для формирования различных справочников
`modules/registers/components/regions`	Справочник регионов

Компонент	Описание
modules/registers/components/devices	Справочник устройств
modules/registers/components/sensors	Справочник датчиков
modules/registers/components/monitored-objects	Справочник объектов мониторинга
modules/registers/components/thresholds	Справочник порогов

Для получения данных для справочников используется сервис src/app/core/services/register.service.ts.

4.7 Уведомления о новых инцидентах

При возникновении нового инцидента создается и отображается пуш-уведомление об инциденте. Для этого используются библиотеки Web Sockets:

- @stomp/ng2-stompjs;
- sockjs-client.

Основной сервис по работе с веб-сокетами – src/app/core/services/web-socket.service.ts. В данном сервисе описано подключение клиента к веб-сокету, а также подписка клиента на получение с сервера информации о новых инцидентах. Когда клиент получает эту информацию, в браузере пользователя отображается уведомление. Работа с уведомлениями описана в сервисе src/app/core/services/notify-service.ts.

Конфигурационный файл с настройками для работы веб-сокетов: src/app/common/config/stomp.config.ts.

5 СОЗДАНИЕ НОВОЙ ФУНКЦИОНАЛЬНОСТИ

Для добавления новых возможностей можно взять за основу уже реализованную функциональность: например, получение данных для рабочего стола, поиск и отображение инцидентов, генерация и просмотр отчетов, просмотр справочников и др.

Создание новой функциональности выполняется в следующей последовательности:

а) выбрать раздел, в котором будет размещаться новая функциональность;

б) создать необходимые для реализации данной функциональности Angular компоненты и сервисы;

в) создать необходимые эндпоинты и контроллеры;

г) создать реализацию в сервисах;

д) при необходимости сохранения в постоянную память создать новые типы данных или добавить поля в существующие;

е) при необходимости скорректировать или добавить вычисление дополнительных данных при получении факта превышения.

5.1 Сохранение новых типов данных в постоянную память

Для сохранения данных вертикального решения в постоянную память необходимо выполнить следующие шаги:

а) создать новую JPA сущность `@Entity` или добавить поля в существующую;

б) создать файл изменений Liquibase, в котором прописать создание нужных таблиц, полей и ограничений;

в) создать Spring репозитории с методами поиска;

г) начать использовать созданные сущности и репозитории в сервисах.

5.2 Дополнительная обработка при регистрации фактов превышения

Дополнительную обработку или вычисления при регистрации фактов превышения можно сделать в методах:

- `ru.waveaccess.iot.incidentapp.service.IncidentService#createNewIncidentOrExtendOld` (решение для Сервиса «Сбора и анализа информации в рамках государственного надзора за состоянием, содержанием, сохранением, использованием, популяризацией и государственной охраной объектов культурного наследия»);
- `ru.waveaccess.iot.incidentapp.service.IncidentService#extendOrCreateNewIncident` (решение для Сервиса «Сбора и анализа информации в рамках экологического надзора в области охраны атмосферного воздуха».).

6 СОЗДАНИЕ УТИЛИТЫ РЕГИСТРАЦИИ ВЕБ-АДРЕСОВ В МЗИ

В составе вертикального решения должен быть разработан модуль для конфигурации веб-адресов в МЗИ, собранный в виде отдельного приложения. Данный модуль предназначен для добавления разрешенных адресов и связывания их с пользователем и ролью в МЗИ для возможности фронтенд части приложения делать запросы из браузера пользователя к закрытой части сети, в которой находится система.

За основу может быть использован уже готовый модуль, который находится в бэкенд части каждого вертикального решения. Данный модуль регистрирует все предоставляемые бэкенд веб-ресурсы. Если в новом вертикально решении создаются новые веб-ресурсы, они должны быть добавлены в файл `api-docs-<тип вертикального решения>.yaml`.