

**АКЦИОНЕРНОЕ ОБЩЕСТВО
НАУЧНО-ПРОИЗВОДСТВЕННЫЙ ЦЕНТР «ЭЛВИС»**

УТВЕРЖДЕН
РАЯЖ.00593-01 32 01-ЛУ

**ЦИФРОВАЯ ПЛАТФОРМА УПРАВЛЕНИЯ ИНЦИДЕНТАМИ
И ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ
NEST-M**

Руководство системного программиста

РАЯЖ.00593-01 32 01

Листов 73

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

2023

АННОТАЦИЯ

Руководство системного программиста РАЯЖ.00593–01 32 01 содержит сведения и инструкции, необходимые для работы системного программиста с цифровой платформой управления инцидентами и принятия решений NEST-M РАЯЖ.00593–01.

Ниже приведено описание разделов документа.

Раздел 1 «Общие сведения о программе» содержит описание цифровой платформы управления инцидентами и поддержки принятия решений NEST-M, ее функциональные возможности, основные понятия с соответствующими определениями, используемые при эксплуатации цифровой платформы, и требования к программным средствам, необходимым для выполнения программы.

Раздел 2 «Структура программы» содержит описание структуры и архитектурных особенностей цифровой платформы управления инцидентами и поддержки принятия решений NEST-M, состав и назначение ее компонентов.

Раздел 3 «Установка программы» содержит описание последовательности действий для установки цифровой платформы управления инцидентами и поддержки принятия решений NEST-M и запуска программных средств, необходимых для ее работы.

Раздел 4 «Управление работой программы» содержит описание последовательности действий для обновления или удаления цифровой платформы управления инцидентами и поддержки принятия решений NEST-M.

Раздел 5 «Написание автоматизаций бизнес-логики» содержит последовательности действий и инструкции для подготовки и настройки интерфейсов цифровой платформы управления инцидентами и поддержки принятия решений NEST-M для работы на конкретном объекте, включая такие этапы, как:

- сбор информации об устройствах для создания объектов;
- сбор информации для написания карточки инцидента;
- создание зон ответственности;
- автоматизация создания объектов на интерактивной карте;
- автоматизация создания события;
- автоматизация создания инцидента;
- создание прототипов;
- создание ролей, пользователей в цифровой платформе;
- просмотр кодов и схем сообщений;
- выявление неисправностей;
- запись данных в базу данных.

Раздел 6 «Проверка программы» содержит инструкции для проверки цифровой платформы управления инцидентами и поддержки принятия решений NEST-M.

«Перечень сокращений» содержит описание сокращений, используемых в документе Руководство системного программиста РАЯЖ.00593–01 32 01.

СОДЕРЖАНИЕ

1	Общие сведения о программе	5
1.1	Описание ЦП NEST-M	5
1.2	Функциональные возможности	6
1.3	Основные понятия.....	6
1.4	Требования к программным средствам	8
2	Структура программы.....	9
3	Установка программы.....	18
3.1	База данных.....	19
3.2	Kafka	19
3.3	Backend	19
3.4	Frontend	21
3.5	Добавление тайлов	22
3.6	Добавление триггеров, прототипов и приоритетов	23
3.7	Запуск установки адаптеров.....	23
3.7.1	Адаптер Тополь	24
3.7.2	Адаптер Sylphide	26
3.7.3	Настройка интеграции Сильфиды	28
3.7.3.1	Установка библиотеки jq	28
3.7.3.2	Настройка интеграции с Сильфидой	28
3.7.3.3	Инициация логина сразу через Сильфиду.....	28
4	Управление работой программы	30
4.1	Удаление программы	30
4.2	Обновление программы.....	30
5	Написание автоматизаций бизнес-логики	31
5.1	Подготовительный этап.....	31
5.1.1	Сбор информации для написания бизнес-логики для создания объектов на плане и событий	31
5.1.2	Сбор информации для написания карточки инцидента	33
5.2	Создание зон ответственностей (scope)	37
5.3	Автоматизация создания POI.....	37
5.4	Автоматизация создания события	52
5.5	Автоматизация создания инцидента	56

5.6	Создание прототипов	58
5.7	Создание ролей, пользователей в ЦП NEST-M	63
5.7.1	Создание ролей	63
5.7.2	Создание пользователей	65
5.8	Просмотр кодов и схем сообщений	66
5.9	Инструкция по выявлению неисправностей	67
5.9.1	Возникновение нескольких сообщений в Kafka вместо одного	67
5.9.2	Отсутствует инцидент в модуле интерфейсов обработки инцидента	68
5.9.2.1	Отсутствие сообщения в kafka о создании инцидента	68
5.9.2.2	Сообщение в Kafka о создании инцидента	68
5.10	Запись автоматов, прототипов, зоны ответственности, файлов в БД	68
5.10.1	Запись автоматов, прототипов, зон ответственности с помощью утилиты VSCode	68
5.10.2	Запись автоматов, прототипов, зон ответственности с помощью утилиты Postman	69
6	Проверка программы	71
ПЕРЕЧЕНЬ СОКРАЩЕНИЙ		72

1 ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

1.1 Описание ЦП NEST-M

Цифровая платформа управления инцидентами и поддержки принятия решений NEST-M (далее - ЦП NEST-M) является комплексом программных средств, предназначенных для формирования единой информационной среды с целью координации работы специалистов по обеспечению безопасности различного уровня для повышения эффективности их взаимодействия.

ЦП NEST-M предназначена для эксплуатации субъектами хозяйственной деятельности в сфере промышленности, транспорта, торговли и функционально обеспечивает решение следующих задач:

- повышение безопасности сотрудников при производстве работ и исполнении функциональных обязанностей;
- снижение уровня угроз от актов незаконного вмешательства в работу субъекта хозяйственной деятельности;
- автоматизированное и централизованное управление интегрированными подсистемами;
- снижение совокупной стоимости владения интегрированными подсистемами;
- прогнозирование возникновения нештатных ситуаций на основе обработки и анализа больших данных.

ЦП NEST-M предназначена для интеграции со следующими классами охранных комплексов (далее – присоединенные системы):

- системы видеонаблюдения (далее СВН);
- системы контроля доступа (далее –СКУД);
- радиолокационные станции (далее – РЛС);
- охранно-пожарные системы (далее – ОПС);
- системы охраны периметра (далее - СОП);
- системы тревожно-вызывной сигнализации (далее - СТВС).

Целью интеграции являются:

- импорт данных о событиях, генерируемых присоединенными системами для последующей обработки в рамках ЦП NEST-M;
- управления устройствами, входящими в состав присоединенных систем в объеме, предоставляемом интерфейсом взаимодействия по единым алгоритмам, задаваемым в ЦП NEST-M;
- мониторинг технического состояния присоединенных систем по единым алгоритмам, задаваемым в ЦП NEST-M.

Архитектура позволяет также обеспечить присоединение к ЦП NEST-M специализированных систем промышленного оборудования, обеспечивающих:

- выявление и устранение потерь в производственных процессах;
- повышение стабильности технологических процессов;
- оптимизация расхода энергии;

- прогнозирование отказов оборудования;
- с целью обработки, обобщения и визуализации полученных от них данных.

1.2 Функциональные возможности

ЦП NEST-M обеспечивает выполнение следующих функций:

- ЦП NEST-M решает задачу автоматической генерации инцидентов на основе получаемых от присоединенных систем событий (в т.ч. по сложным, увязывающим события от нескольких систем, алгоритмам);
- ЦП NEST-M решает задачу визуализации оперативной информации на геопривязанной карте;
- ЦП NEST-M поддерживает совместную обработку сгенерированных инцидентов с распределением их между операторами по заданным алгоритмам. Работа операторов поддерживается с предоставлением справочной и нормативной информации в контексте инцидента. Обработка инцидентов операторами построена по иерархическому принципу с поддержкой как горизонтального взаимодействия операторов при решении задачи, так и поддержкой механизмов эскалации;
- ЦП NEST-M обеспечивает возможность автоматической генерации отчетов произвольной формы о процессе обработки как по конкретному инциденту, так и по произвольной группе инцидентов.

1.3 Основные понятия

При описании работы с программным обеспечением в настоящем документе используются следующие понятия с соответствующими определениями с соответствующими определениями.

База данных (БД) - информация, необходимая для работы системы хранится в базах данных на сервере баз данных SQL Server. Эту информацию можно условно разделить на постоянную и переменную. К постоянной информации относится информация о конфигурации и настройках ЦП NEST-M, а также информация о пользователях. К переменной относится информация о событиях, происходящих в системе в том числе об инцидентах, произошедших на объекте контроля;

Пользователь - лицо, которое использует ЦП NEST-M для выполнения конкретной функции.

Учетная запись (аккаунт) – это хранимая совокупность данных о пользователе, необходимая для его аутентификации и предоставления доступа к компонентам системы.

Оператор - это пользователь ЦП NEST-M, управляющий клиентским приложением. Для каждого оператора должны быть определены имя, пароль и роль аккаунта в базе данных. Количество операторов в системе не ограничено. Оператор не имеет доступа к базе данных (менеджеру конфигураций). Роль аккаунта оператора так же назначается администратором, имеющим доступ к базе данных;

Администратор – это роль учетной записи, имеющая доступ к менеджеру конфигураций, где можно добавлять, редактировать и удалять информацию из базы данных системы;

Авторизация доступа - позволяет разграничить доступ пользователей к различной информации при работе с ЦП NEST-M. При работе с SQL Server на сервере БД должны быть созданы учетные записи пользователей, и указаны для них методы авторизации. Доступ к ЦП NEST-M имеет зарегистрированный оператор, для которого назначено имя и пароль доступа. Также для каждого оператора назначается роль учетной записи для обработки и создания инцидентов. Полномочия могут быть установлены на добавление, чтение или редактирование информации.

Событие - ситуация (происшествие), возникшая в точке контроля, уведомление о которой передается в клиентское приложение ЦП NEST-M, и требующая внимания или действий от оператора в дальнейшем.

Инцидент - автоматически создается системой или оператором в клиентском приложении ЦП NEST-M на основании события или в последствии эволюции другого инцидента. У каждого инцидента есть свой маршрут обработки, набор доступных действий и т.д.;

Отчет - документированные сведения о параметрах инцидента, действиях оператора по данному инциденту, с возможностью фильтрации по свойствам, сохранения и печати (разделяются на отчеты по списку, по инциденту и по пользователю);

Роли учетных записей – разграничение прав учетных записей пользователей, разрешающие/запрещающие работать пользователю с различными модулями системы и зонами ответственности.

Триггеры - определяет правила формирования сообщений на основании других сообщений.

Зона ответственности – наборы правил, группирующие события, инциденты и РОІ по их ключевым признакам с целью организации одно- или многоуровневой структуры независимого управления и мониторинга охраняемых объектов для пользователей с различающимися функциями, правами, территориальным размещением и допуску к принятию решений.

Прототип инцидента – структура данных, задающая жизненный цикл инцидента: создание, обработка, эскалация (увеличение приоритета), деэскалация и закрытие инцидента

Приоритет инцидента – призван отразить важность инцидента относительно других. Каждому инциденту при настройке присваивается свой приоритет. Для оператора инциденты высокого приоритета отображаются всегда выше, чем инциденты с более низким приоритетом.

Присоединенная система – внешняя (относительно ЦП NEST-M) информационная система, поставляющая информацию о событиях и объектах в ЦП NEST-M через интерфейсы взаимодействия, которой также могут быть направлены команды

Адаптер - обеспечивает передачу данных между ЦП NEST-M и присоединенными системами

Форма ручных событий – это форма, с помощью которой оператор может создать собственное событие, которое нельзя зафиксировать автоматическими средствами обеспечения безопасности.

Наборы ручных событий – объединение ручных событий в единую категорию

Объект – охранные устройства, информация о которых была получена от присоединенных систем через соответствующие модули ЦП NEST-M.

3D объект – загруженная в систему 3D модель «здания», отображаемая на карте.

Зона – область, отображаемая на карте, к которой присоединена определенная группа объектов.

1.4 Требования к программным средствам

Для обеспечения работы серверной части ПО ЦП NEST-M требуется подключение к:

- базе данных PostgreSQL версии не ниже 11;
- Apache Kafka 3.0.x.

На сервере должен быть установлен Docker.

Дистрибутив ЦП NEST-M может быть установлен на ОС AstraLinux 1.7.1, Ubuntu 20.04.

На клиентских ПК (АРМ операторов) взаимодействие с ЦП NEST-M осуществляется с использованием браузеров Chrome версии 116 и выше.

2 СТРУКТУРА ПРОГРАММЫ

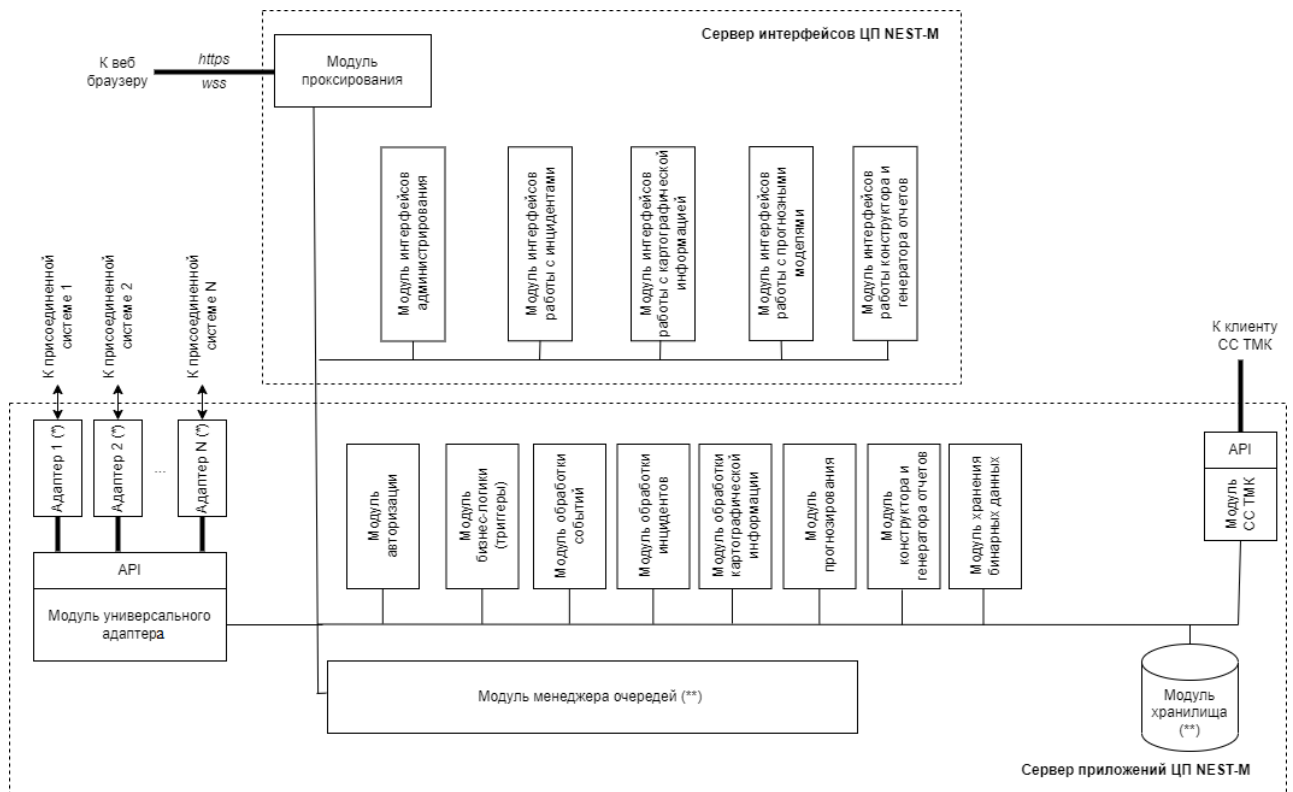
Структура ЦП NEST-M разработана в рамках идеологии микросервисной архитектуры, как способа разработки программного обеспечения, характеризующегося наличием в системе набора небольших, слабо связанных и легко изменяемых компонентов - микросервисов.

Микросервисы в составе ЦП NEST-M представлены модулями. Модули в составе сервера и АРМ ЦП NEST-M реализуются как docker-контейнеры на основе операционной системы AstraLinux Special Edition версии не ниже 1.7.

Модули в составе ЦП NEST-M разделены по функциональному назначению на:

- модули в составе сервера интерфейсов ЦП NEST-M (реализуют функциональность пользовательских интерфейсов ЦП NEST-M);
- модули в составе сервера приложений ЦП NEST-M (реализуют функциональность приема и передачи информации, ее обработки и хранения).

Структурная схема ЦП NEST-M приведена ниже (см. рисунок 1).



На схеме:

— Внутренняя TCP/IP сеть ЦП NEST-M

— Внешняя TCP/IP сеть

↔ Взаимодействие компонентов

(*) Не входит в состав ЦП NEST-M

(**) Модуль может быть реализован как внешний относительно ОС сервера приложений

Рисунок 1

Состав и назначение компонентов ЦП NEST-M, представленных на схеме, приведены в таблице 1.

Таблица 1 - Состав и назначение компонентов цифровой платформы NEST-M

Наименование компонента	Краткое наименование в документе	Назначение	Особенности реализации
1. Компоненты в составе сервера приложений ЦП NEST-M			
Модуль хранилища	МХ	Организация хранилища структурированных данных в виде таблиц	Реализуется на базе СУБД PostgreSQL. Может развертываться как в составе сервера приложений ЦП NEST-M, так и в виде отдельного кластера в инфраструктуре развертывания ЦП NEST-M
Модуль менеджера очередей	Менеджер очередей, МО	Организация потоков данных между модулями системы через централизованную среду. Является центральным элементом событийно-ориентированной архитектуры ЦП NEST-M. Позволяет изолировать и упорядочить обработку потоков данных в отдельных модулях комплекса	Реализуется с помощью пакетов Kafka-Zookeeper и Kafka-Broker. Может развертываться как в составе сервера приложений ЦП NEST-M, так и в виде отдельного кластера в инфраструктуре развертывания ЦП NEST-M.

Наименование компонента	Краткое наименование в документе	Назначение	Особенности реализации
Модуль хранения бинарных данных	МХБД	Сохранение и выдача бинарных данных через запросы по протоколу HTTP	
Модуль универсального адаптера	универсальный адаптер, УА	Интеграция внешних информационных систем с ЦП NEST-M через их интерфейсы взаимодействия	Адаптеры присоединенных систем разрабатываются индивидуально для отдельных типов присоединенных систем с целью согласования программной логики API присоединенной системы и универсального адаптера ЦП NEST-M. Они не входят в состав ЦП NEST-M и могут развертываться как на платформе сервера приложений ЦП NEST- M так и в инфраструктуре развертывания ЦП NEST-M
Адаптеры присоединенных систем	адаптеры ПС		
Модуль авторизации	МА	Авторизация и аутентификация пользователей ЦП NEST- M в системе и организация разграничения предоставления для них	Реализуется с помощью пакета Keycloak.

Наименование компонента	Краткое наименование в документе	Назначение	Особенности реализации
		прав доступа к ресурсам платформы	
Модуль бизнес-логики	МБЛ	Обработка сообщений из потоков менеджера очередей на основе заранее сконфигурированных правил. Позволяет генерировать новые сообщения на основе обработанных.	
Модуль обработки событий	МОС	Формирование и сохранение в БД событий, созданных на основе сообщений из потоков менеджера очередей. Рассылка новых сообщений и поиск архивных по запросам операторов.	
Модуль обработки инцидентов	МОИ	Формирование и сохранение в БД инцидентов, созданных на основе сообщений из потоков менеджера очередей. Рассылка и обработка новых инцидентов и поиск архивных по запросам операторов.	
Модуль обработки картографической информации	МОКИ	Реализации функциональности: -подготовка, формирование и	Подготовка и формирование данных для визуализации карт реализуется с

Наименование компонента	Краткое наименование в документе	Назначение	Особенности реализации
		хранение данных для визуализации карт; -организации работы с объектами картографической информации	помощью пакета Sharp для Node.js. Визуализация карт реализуется с помощью пакета TileServer-GL
Модуль прогнозирования	МП	Реализация функциональности работы с прогнозными моделями, предназначенными для прогнозирования возникновения нештатных ситуаций	
Модуль конструктора и генератора отчетов	МКГО	Реализация функциональности конструктора и генератора отчетов	Реализуется с помощью библиотеки генерации отчетов и создания документов FastReport .NET (с ПО дизайнера отчетов FastReport Online Designer в составе)
Модуль системы сборов результатов технического мониторинга и контроля объектов транспортной инфраструктуры	МССТМК	Реализация функциональности интеграции цифровой платформы с ГИС «Система сбора результатов технического мониторинга и контроля объектов транспортной инфраструктуры»	

Наименование компонента	Краткое наименование в документе	Назначение	Особенности реализации
2. Компоненты в составе сервера интерфейсов ЦП NEST-M			
Модуль проксирования	МПРОКС	Ретрансляция запросов пользователей из внешней сети, поступающих по протоколу https, к соответствующим модулям ЦП NEST-M.	Реализуется с помощью пакета nginx
Модуль интерфейсов администрирования	МИА	Реализация пользовательских интерфейсов (экранных форм), предназначенных для управления и конфигурирования работы цифровой платформы	
Модуль интерфейсов работы с инцидентами	МИРИ	Реализация пользовательских интерфейсов (экранных форм), предназначенных для поддержки функциональности отображения, обработки и сопровождения событий и инцидентов	
Модуль интерфейсов работы с картографической информацией	МИРКИ	Реализация пользовательских интерфейсов (экранных форм) предназначенных: -для работы с интерактивной картой охраняемого объекта (объектов) с целью визуализации данных о размещении охранных	

Наименование компонента	Краткое наименование в документе	Назначение	Особенности реализации
		<p>устройств, их текущего состояния, а также для отражения данных, необходимых для мониторинга оперативной обстановки на охраняемом объекте в визуальной форме;</p> <p>-для поддержки функциональности отображения событий (экранные формы), предназначенных для поддержки функциональности отображения событий</p>	
<p>Модуль интерфейсов работы с прогнозными моделями</p>	<p>МИП</p>	<p>Реализуются пользовательские интерфейсы (экранные формы), предназначенные для поддержки функциональности работы с прогнозными моделями и управления их функционированием</p>	
<p>Модуль интерфейсов работы конструктора и генератора отчетов</p>	<p>МИКГО</p>	<p>Для реализации пользовательских интерфейсов (экранных форм), предназначенных для поддержки функциональности генерации отчетов</p>	<p>Реализуется с помощью библиотеки генерации отчетов и создания документов FastReport .NET (с ПО дизайнера отчетов FastReport Online Designer в составе)</p>

Модуль проксирования производит преобразование внешних запросов (работа с SSL сертификатами), поступающих от браузера пользователя, и перенаправление запросов на выдачу статического и динамического контента на соответствующие модули, входящие в состав ЦП NEST-M.

Модули интерфейсов из состава сервера интерфейсов ЦП NEST-M:

- интерфейсов работы с инцидентами;
- интерфейсов работы с картографической информацией;
- интерфейсов работы с прогнозными моделями;
- интерфейсов работы конструктора и генератора отчетов;
- интерфейсов администрирования.

Модули интерфейсов взаимодействуют с соответствующими модулями сервера ЦП NEST-M:

- модулем бизнес-логики;
- модулем обработки событий;
- модулем обработки инцидентов;
- модулем обработки картографической информации;
- модулем прогнозирования;
- модулем конструктора и генератора отчетов;
- модулем хранения бинарных данных;
- модулем авторизации.

В состав каждого модуля в составе сервера интерфейсов ЦП NEST-M включается собственный веб-сервер для отдачи статического контента, выдачи динамического контента, генерируемого в рамках данного модуля или перенаправления запросов к соответствующим модулям в составе сервера приложений ЦП NEST-M. В состав каждого модуля в составе сервера приложений ЦП NEST-M включается собственный веб-сервер для реализации API.

Все приложения, реализующие функциональность интерфейсов (экранные формы) пользователя, являются SPA (одностраничными) приложениями. Веб-сервер отправляет статические файлы модулей, выполняемые на стороне клиента (браузера).

Для сборки приложений, реализуемых в модулях сервера интерфейсов ЦП NEST-M, используется программная платформа Node.JS версии 16 и выше.

Основой взаимодействия модулей в составе сервера приложений ЦП NEST-M являются потоки данных (очереди), поддерживаемые модулем менеджера очередей. Данная архитектура позволяет изолировать и упорядочить обработку потоков данных в отдельных модулях (событийно-ориентированной архитектуры). Среды разработки функциональности модулей в составе сервера ЦП NEST-M, применяемые при этом библиотеки и пакеты ПО, рассмотрены в соответствующих разделах настоящего документа.

В составе MX ЦП NEST-M развертывается СУБД PostgreSQL версии не ниже 11.10 на базе дистрибутива из состава ОС AstraLinux Special Edition согласно документации.

Взаимодействие отдельных модулей ЦП NEST-M друг с другом осуществляется по стеку протоколов TCP/IP в пределах выделенного замкнутого адресного пространства платформы. Дополнительно для платформы может быть осуществлено разделение сетевого доступа для подключения к следующим функциональным сетевым сегментам:

- пользовательский сегмент (через модуль проксирования);
- технологический сегмент систем безопасности (через модуль универсального адаптера);
- сегмент внешних информационных систем (через модуль СС ТМК).

3 УСТАНОВКА ПРОГРАММЫ

Установка программы осуществляется следующим образом:

1) копируем все необходимые ресурсы на сервер, например, в папку «NEST-M»;

2) команды выполняем из-под учетной записи с правами администратора, перед дальнейшими шагами можно вызвать bash из-под администратора

```
sudo bash
```

3) переходим в папку installer. Она находится по пути /НОМЕР_ВЕРСИИ/installer

```
cd installer
```

4) вызываем из папки installer скрипт install.sh

```
./install.sh
```

При выполнении команды может возникнуть ошибка «Permission denied» (см. рисунок 2). В таком случае нужно дать разрешение на запуск файла install.sh.



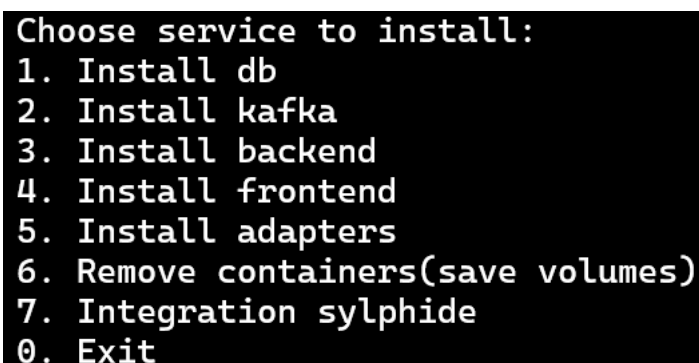
```
vladimir@MacBook-Pro-Vladimir:~/work/git-elv  
zsh: permission denied: ./install.sh  
vladimir@MacBook-Pro-Vladimir:~/work/git-elv
```

Рисунок 2

Для этого можно, например, воспользоваться командой:

```
chmod 777 ./install.sh
```

5) после запуска скрипта install.sh предложены пакеты, необходимые для установки (см. рисунок 3). Нужно последовательно установить все пакеты.



```
Choose service to install:  
1. Install db  
2. Install kafka  
3. Install backend  
4. Install frontend  
5. Install adapters  
6. Remove containers(save volumes)  
7. Integration sylphide  
0. Exit
```

Рисунок 3

3.1 База данных

При вводе "1" будет предложено поднять и настроить контейнер с БД Postgres. Необходимо ввести логин и пароль администратора БД (см. рисунок 4).

По умолчанию:

- Логин: postgres;
- Пароль: postgres.

```
Enter database username to connect [postgres]:  
Enter database password to connect [postgres]:
```

Рисунок 4

Начнётся настройка контейнера с БД Postgres (см. рисунок 5). В итоге, настройка будет успешно завершена.

```
[+] Running 2/2  
  ✓ Container db Started  
  ✓ Container adminer Started
```

Рисунок 5

3.2 Kafka

При вводе "2" будет предложено поднять и настроить контейнер Kafka. Для этого необходимо ввести IP-адрес компьютера (см. рисунок 6).

```
Enter address kafka to connect [nestorsb-nest-control  
-control-web-126.elvees.com]: 10.102.0.116
```

Рисунок 6

Начнётся настройка контейнера Kafka (см. рисунок 7). В итоге, настройка будет успешно завершена.

```
[+] Running 3/3  
  ✓ Container zookeeper-0 Started  
  ✓ Container broker-0 Started  
  ✓ Container kafka-ui Started
```

Рисунок 7

3.3 Backend

При вводе "3" будет предложена настройка всех backend-сервисов:

1) необходимо ввести адрес БД (см. рисунок 8);

```
Enter database address to connect [nestorsb-nest-control-control-web-126.elvees.com]: 10.102.0.116
```

Рисунок 8

2) необходимо ввести логин пользователя (см. рисунок 9). По умолчанию: postgres;

```
Enter database username to connect [postgres]:
```

Рисунок 9

3) необходимо ввести пароль (см. рисунок 10). По умолчанию: postgres;

```
Enter database password to connect [postgres]:
```

Рисунок 10

4) необходимо указать адрес для подключения к Kafka в виде IP_АДРЕС_МАШИНЫ:9092 (см. рисунок 11);

```
Enter kafka address and port to connect [nestorsb-nest-control-control-web-126.elvees.com:9092]: 10.102.0.116:9092
```

Рисунок 11

5) необходимо ввести логин и пароль от сервиса авторизации (см. рисунок 12 и рисунок 13). По умолчанию:

- логин: admin;
- пароль: admin;

```
Enter keycloak user name [admin]:
```

Рисунок 12

```
Enter keycloak password [admin]:
```

Рисунок 13

б) необходимо указать адрес сервера, на котором установлен frontend (см. рисунок 14). А именно IP-адрес текущего сервера при односерверной установке.

```
Enter frontend address [nestorsb-nest-control-control-web-126.elvees.com]: 10.102.0.116
```

Рисунок 14

Начнётся настройка backend (см. рисунок 15). В итоге, настройка будет успешно завершена.

```
[+] Running 11/11
  ✓ Volume "installer_plan-data" Created
  ✓ Container tileserver-gl Started
  ✓ Container keycloak1802 Started
  ✓ Container tilemaker Started
  ✓ Container filestore-service Started
  ✓ Container business-service Started
  ✓ Container monitor-service Started
  ✓ Container import-service Started
  ✓ Container control-service Started
  ✓ Container timeline-service Started
  ✓ Container report-service Started
```

Рисунок 15

3.4 Frontend

При вводе "4" будет предложена установка всех frontend-сервисов:

1) необходимо ввести IP-адрес сервера на котором развернут backend (см. рисунок 16). А именно IP-адрес текущего сервера при односерверной установке;

```
Enter api hostname address name [nestorsb-nest-control-control-web-126.elvees.com]: 10.102.0.116
```

Рисунок 16

2) необходимо указать адрес сервера, на котором установлен frontend (см. рисунок 17). А именно IP-адрес текущего сервера при односерверной установке;

```
Enter hostname address name [nestorsb-nest-control-control-web-126.elvees.com]: 10.102.0.116
```

Рисунок 17

3) вводим широту и долготу начальной точки на карте (см. рисунок 18);

```
Enter start lat coord [56.007118]:  
Enter start lng coord [37.156594]:
```

Рисунок 18

Начнётся настройка frontend (см. рисунок 19). В итоге, настройка будет успешно завершена.

```
[+] Running 4/4  
✔ Container monitor-web Started  
✔ Container control-web Started  
✔ Container web-server Started  
✔ Container setup-web Started
```

Рисунок 19

После установки всех пакетов можно выйти из install.sh, нажав "0" в меню (см. рисунок 20).

```
Choose service to install:  
1. Install db  
2. Install kafka  
3. Install backend  
4. Install frontend  
5. Install adapters  
6. Remove containers(save volumes)  
7. Integration sylphide  
0. Exit
```

Рисунок 20

3.5 Добавление тайлов

Для добавления тайлов требуется выполнить:

1) файл tiles.mbtiles нужно скопировать по пути /var/opt/elvees.com/installer/tiles. Для этого можно использовать команду:

```
cp ./TILES_PATH/tiles.mbtiles /var/opt/elvees.com/installer/tiles
```

В случае ошибки доступа требуется добавить права на запись. Для этого можно использовать команду:

```
chmod 777 /var/opt/elvees.com/installer/tiles/
```

2) нужно перезагрузить тайловый сервис с помощью команды:

docker restart tileserver-g1

В итоге система будет установлена. Это можно проверить, перейдя на адреса:

- IP_сервера/control;
- IP_сервера/monitor.

Система потребует ввести логин и пароль. По умолчанию:

- Для пользователя логин и пароль: nest-user;
- Для администратора логин и пароль: nest-admin.

3.6 Добавление триггеров, прототипов и приоритетов

Для добавления триггеров, прототипов и приоритетов необходимо перейти в папку stage-3 и запустить скрипт setup.sh.

```
./setup.sh IP_адрес_машины_на_которой_установлен_бек
```

В случае возникновения ошибки доступа необходимо дать разрешения на запуск. Например, с помощью команды:

```
chmod 777 ./setup.sh
```

3.7 Запуск установки адаптеров

Для запуска установки адаптеров требуется выполнить:

1) в install.sh включить адаптеры систем:

- Тополь;
- Sylphide;
- Synesys.

Для начала установки нужно запустить из папки installer скрипт install.sh:

```
./install.sh
```

2) в открывшемся окне необходимо ввести "5" и нажать Enter;

3) нужно ввести IP-адрес сервера, на котором развернут backend, а именно IP-адрес текущего сервера при односерверной установке (см. рисунок 21).

```
Enter backend address name [nestorsb-nest-control-control-web-126.elvees.com]: 10.102.0.116
```

Рисунок 21

Начнётся настройка адаптеров (см. рисунок 22). В итоге, настройка будет успешно завершена.

```
[+] Running 3/3
  ✓ Container topol-adapter      Started
  ✓ Container sylphide-adapter   Started
  ✓ Container nestd-adapter      Started
```

Рисунок 22

3.7.1 Адаптер Тополь

Для настройки адаптера Тополя требуется:

1) перейти по пути `/var/opt/elvees.com/installer`:

```
cd /var/opt/elvees.com/installer
```

2) начать редактировать файл `adapters.appsettings.config.json`:

```
nano adapters.appsettings.config.json
```

3) ввести массив адресов, где будет расположен Тополь в формате ['IP_номер1', 'IP_номер2'] (см. рисунок 23);

```
//Адрес устройства (или имитатора), поставщика данных  
"TopolServers": [ "http://localhost:5050/mock/topol",  
"http://localhost:5060/mock/topol"]
```

Рисунок 23

4) при необходимости поменять фильтр:

- кодов событий (см. рисунок 24);
- типов датчиков (см. рисунок 25);
- идентификаторов устройств (см. рисунок 26);


```
// Фильтр по кодам событий  
"IncludeEventCodes": [  
  100,  
  101  
],
```

Рисунок 24

```
// Фильтр по типам входов |  
"IncludeInputTypes": [  
  100  
],
```

Рисунок 25

```
// Фильтр по идентификаторам устройств  
"IncludeDeviceUids": [  
  "a596ff2c-d771-47ed-a090-3032d027ee01"  
  // "844bf57b-3bfd-4753-815d-e2643cfbe787",  
  // "a596ff2c-d771-47ed-a090-3032d027ee01",  
  // "1491529d-c180-4f89-981b-06f2780484a5",  
  // "f08e71d4-0532-4920-93ee-b54b86b7ae3f",  
  // "fe9ff377-aa7f-4e47-ac40-bcb955b70920",  
  // "6e76246e-f6f9-4aeb-b645-e387dbfbf4e8",  
  // "a38a332e-ac99-422d-95e5-244d570c40b4",  
]
```

Рисунок 26

5) сохранить редактируемый файл `adapters.appsettings.config.json`;

6) перезапустить адаптер Тополя с помощью команды:

```
docker restart topol-adapter
```

В итоге адаптер Тополя будет настроен, а в Модуле интерфейсов работы с картографической информацией в левом списке будут отображаться объекты Тополя (см. рисунок 27).

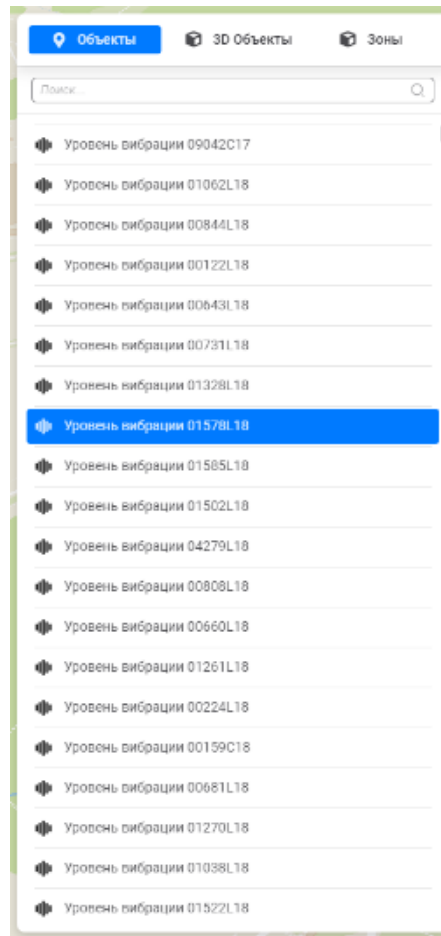


Рисунок 27

3.7.2 Адаптер Sylphide

Для настройки адаптера Sylphide:

1) нужно перейти по пути `/var/opt/elvees.com/installer`:

```
cd /var/opt/elvees.com/installer
```

2) начать редактировать файл `adapters.appsettings.config.json`:

```
nano adapters.appsettings.config.json
```

3) ввести адрес сервера с Sylphide в графу «SylphideAddress» (см. рисунок 28);

```
"SylphideOptions": {  
  "SylphideAddress": "http://sylphide-orsb.elvees.com/",  
  "IdentityServerPort": 8040,  
  "PresentationServerPort": 8081,  
  "VideoServerPort": 8090,  
  "Address": "http://10.102.0.120:5851/services/import",  
  "Name": "Sylphide",  
  "Code": "Sylphide",  
  "Actor": "sylphide",  
  "DisplayName": "Sylphide",  
  "Login": "nest",  
  "Password": "12345",  
  "StatusFilter": [  
    "unresolved",  
    "unconfirmed",  
    "confirmed"  
  ],  
  "MaxDegreeOfParallelism": 4,  
  "ConnectionTimeout": 100,  
  "ArchiveMargin": "00:00:15"  
}
```

Рисунок 28

- 4) сохранить редактируемый файл `adapters.appsettings.config.json`;
- 5) перезапустить адаптер Sylphide с помощью команды:

`docker restart sylphide-adapter`

В итоге адаптер Sylphide будет настроен, а в Модуле интерфейсов работы с картографической информацией в левом списке будут отображаться объекты Sylphide (см. рисунок 29).

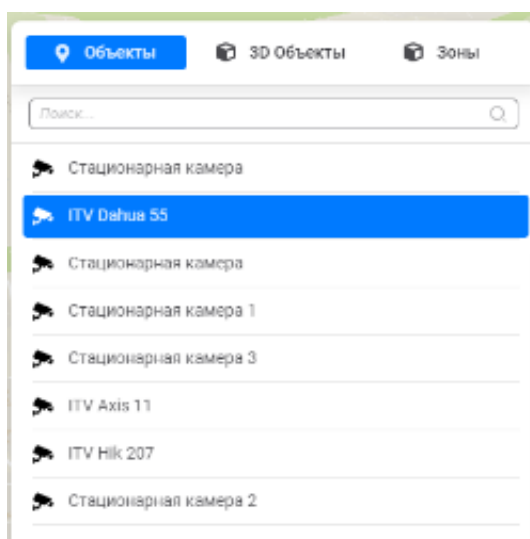


Рисунок 29

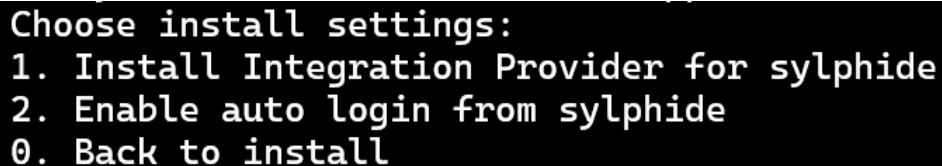
3.7.3 Настройка интеграции Сильфиды

3.7.3.1 Установка библиотеки jq

Для настройки интеграции с программным обеспечением для обработки и анализа видеоданных «Сильфида VMS» РАЯЖ.00551-01 (далее – Сильфида) требуется библиотека jq, которую нужно скопировать на сервер и установить с помощью скрипта `install.sh` в папке `installer`:

- 1) копируем архив с установщиком jq на сервер;
- 2) далее выполняем команды:

```
chmod 777 jq-linux64  
mv jq-linux64 /bin/jq
```
- 3) вводим "7" нажимаем Enter;
- 4) будет переход в отдельный раздел инсталлятора, появится меню (см. рисунок 30).



```
Choose install settings:  
1. Install Integration Provider for sylphide  
2. Enable auto login from sylphide  
0. Back to install
```

Рисунок 30

После каждой введенной цифры и исполнения пункта будет возврат в это меню до тех пор, пока не будет осуществлен выход (введен 0).

3.7.3.2 Настройка интеграции с Сильфидой

Внутри 1 пункта (см. рисунок 30) необходимо ввести:

- адрес сервера с ЦП NEST-M;
- логин подключения к Keycloak: *admin*;
- пароль подключения к Keycloak: *admin*;
- адрес Сильфиды, где находится identityserver: *sylphide-orsb.elvees.com*;
- порт Сильфиды, где находится identityserver 8040: *zauum* в *installer*;
- секрет для соединения с identityserver: *zauum* в *installer*.

После этой настройки на окне входа в ЦП NEST-M появится раздел «Войти через» и будет размещена кнопка «Сильфида». Чтобы войти через Сильфиду в ЦП NEST-M, нужно будет нажать на нее. После нажатия этой кнопки будет выполнен переход на страницу входа в Сильфиду. После логина в ней - переход в конкретное окно NEST- M.

3.7.3.3 Инициация логина сразу через Сильфиду

Внутри 2 пункта (см. рисунок 30) необходимо ввести:

- адрес сервера с ЦП NEST-M;
- логин подключения к Keycloak: *admin*;
- пароль подключения к Keycloak: *admin*.

Эта настройка используется в дополнение к пункту (1). Вместо окна входа в ЦП NEST-M и отдельной кнопки для "Войти через" пользователь будет сразу направлен на страницу логина Сильфиды. Остальной процесс - прежний.

Замечание: если при настройке интеграции с Сильфидой была произведена настройка через IP-адрес, то авторизоваться необходимо через IP-адрес. Аналогично и через доменное имя.

4 УПРАВЛЕНИЕ РАБОТОЙ ПРОГРАММЫ

4.1 Удаление программы

Для удаления программы необходимо выполнить последовательность действий:

1) ЦП NEST-M устанавливается по пути /var/opt/elvees.com. Для удаления необходимо удалить папку /elvees.com:

```
rm -rf /var/opt/elvees.com
```

2) далее необходимо удалить контейнеры приложения с помощью команды:

```
docker rm $(docker ps -aq) -f
```

3) затем удалить образы контейнеров с помощью команды:

```
docker rmi $(docker images -q) -f
```

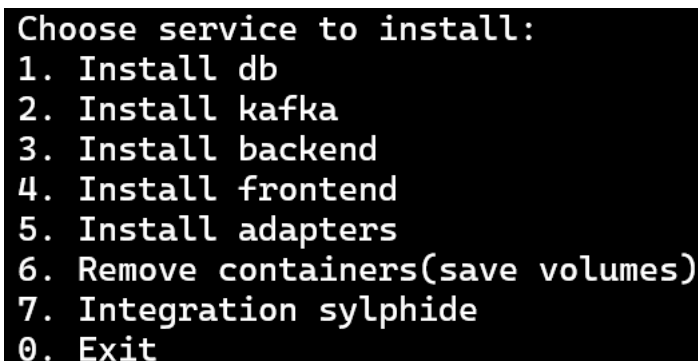
4) после удалить volume с помощью команды:

```
docker volume rm $(docker volume ls -q)
```

4.2 Обновление программы

Для обновления программы необходимо выполнить последовательность действий:

1) необходимо в папке installer запустить скрипт install.sh (см. рисунок 31).



```
Choose service to install:
1. Install db
2. Install kafka
3. Install backend
4. Install frontend
5. Install adapters
6. Remove containers(save volumes)
7. Integration sylphide
0. Exit
```

Рисунок 31

2) ввести "6" и нажимаем Enter. После этого будут удалены контейнеры frontend и backend. При этом контейнеры БД (postgres), Kafka, и адаптеры не пострадают. А данные, хранящиеся в БД, удалены не будут;

3) после этого следуем инструкциям по установке контейнеров backend и frontend. Они описаны в подразделах 3.3 и 3.4.

5 НАПИСАНИЕ АВТОМАТИЗАЦИЙ БИЗНЕС-ЛОГИКИ

5.1 Подготовительный этап

5.1.1 Сбор информации для написания бизнес-логики для создания объектов на плане и событий

Перед написанием бизнес-логики устройства (POI) на карте необходимо выяснить следующие моменты:

- тип устройства, который будет отображаться на плане;
- перечень доступных действий, отображаемых у POI на вкладке «Управление» (см. рисунок 32). Перечень доступных действий и схемы их команд зависят от присоединенной системы;

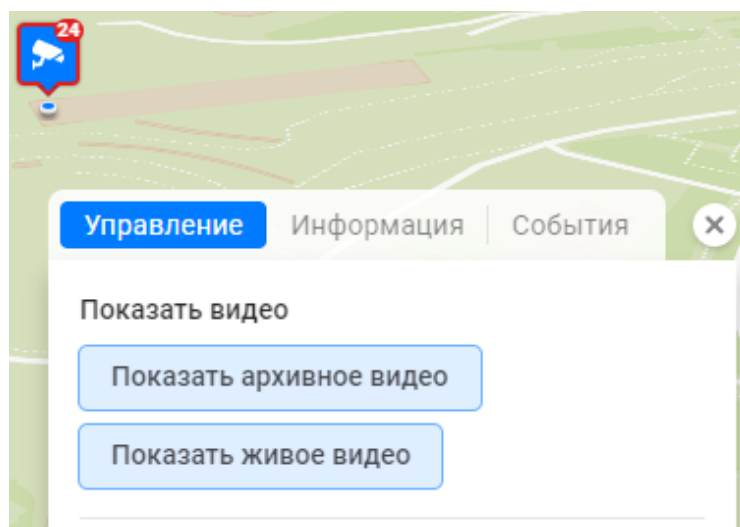


Рисунок 32

- перечень информации, отображаемой у POI на вкладке «Информация» (см. рисунок 33);

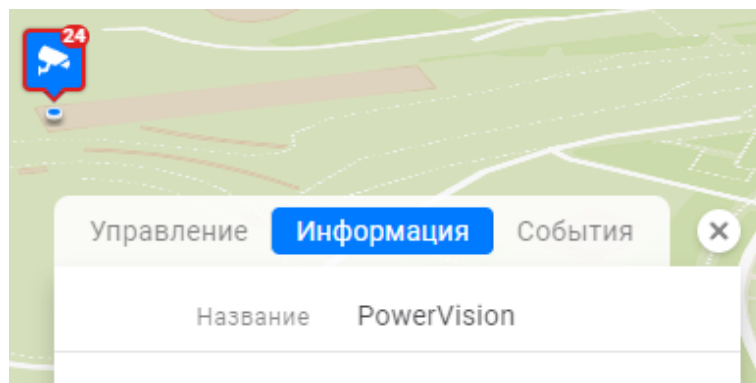


Рисунок 33

– перечень событий, отображаемых у ROI на вкладке «События» (см. рисунок 34)

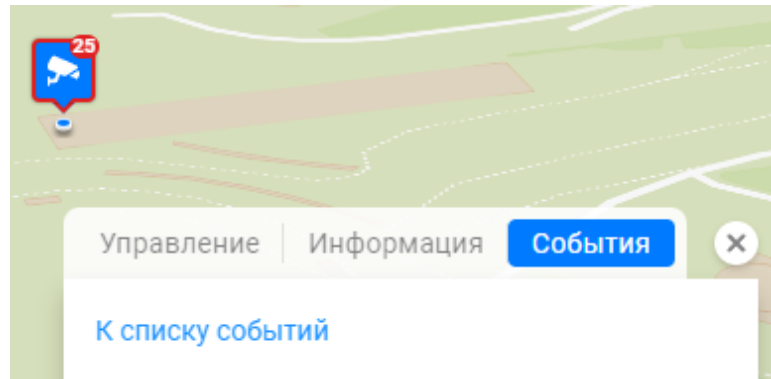


Рисунок 34

и отображаемых в таблице «События» (см. рисунок 35).

Автофильтр					
События	Название	Дата и время	Идентификатор	Статус	
PowerVision	UserAlarm	28.09.2023 14:25:22	275	●	
PowerVision	UserAlarm	28.09.2023 14:24:22	274	●	
PowerVision	UserAlarm	28.09.2023 14:22:18	273	●	
PowerVision	UserAlarm	28.09.2023 14:21:57	272	●	
PowerVision	UserAlarm	28.09.2023 14:20:31	271	●	

Рисунок 35

– перечень полей и информации, отображаемой в окне дополнительной информации события в таблице «события» (см. рисунок 36).

Дополнительные данные		
События	Информация о событии	
Турникет-2 вход	Идентификатор	fef48784-1401-4a8c-8073-5101950e9f68
Турникет-1 вход	Название события	Тревога
Турникет-2 вход	Время возникновения события	31.01.2022 13:18:23
Турникет-2 вход	Время регистрации события в ЦП NEST-M	27.01.2024 11:53:37
Уровень вибрации 00054L	Статус	Тревога
Уровень вибрации 00712L	Датчик	Уровень вибрации
Уровень вибрации 01290L	Прибор	Прибор без имени
Уровень вибрации 01329L	Тип прибора	ДД-1
Уровень вибрации 00054L	Адрес прибора	172.21.242.111
Уровень вибрации 00712L	Серийный номер прибора	00054L18
Уровень вибрации 01290L	<input type="button" value="Создать отчет"/>	
Уровень вибрации 03630L		
Уровень вибрации 01329L		
Уровень вибрации 00054L		
Уровень вибрации 00712L		

Рисунок 36

5.1.2 Сбор информации для написания карточки инцидента

Перед написанием бизнес-логики для карточки инцидента необходимо выяснить следующие моменты:

- сообщения участвующие в формировании и обработке карточки инцидента. Например, «Нажата планка антипаники», «Отжата планка антипаники»;
- перечень полей этих сообщений (схема сообщений), которых будут отображаться в карточке инцидента, а также значений полей (если такое условие есть), которые будут использоваться в качестве условий формирования инцидента. Например: eventName (будет выводиться в карточке в виде названия события), events_Id, sensors_Id, dateTime и так далее.

Схему входящего сообщения, можно посмотреть в БД в строке в таблицы схем сообщений по адресу http://имя_сервера_NEST-M:5080. После этого:

- 1) открыть DB import;
- 2) выбрать «schemas»;
- 3) посмотреть схему в столбцах properties и descriptions, где schema_code = «event» (см. рисунок 37).

Adminer 4.8.1

DB: import
Схема: public

SQL-запрос: `SELECT * FROM "schemas" where module_id='95802d84-31e9-4624-9078-b4e5008f85c2' LIMIT 50`

id	module_id	schema_code	display_name	
43e9e100-f99c-4b7b-a993-c3eb95c11c0c	95802d84-31e9-4624-9078-b4e5008f85c2	Alarm	Тревога	{"compId": "string", "termId": "string", "ev
306660d8-5a81-44dd-825a-da8f925ee645	95802d84-31e9-4624-9078-b4e5008f85c2	Computer	Компьютер	{"ipAddress": "string", "computers_Id": "st
65d55f61-fe17-4c55-8067-a22c4f30b796	95802d84-31e9-4624-9078-b4e5008f85c2	Sensor	Сенсор	{"name": "string", "type": "string", "typeAI
e309c32b-fd37-49ab-99f5-9a53b708af50	95802d84-31e9-4624-9078-b4e5008f85c2	ArmTerminal	Поставить на охрану терминал	{"operId": "String", "termId": "String", "ev
08187f33-de45-42e3-9595-6b3ff6474299	95802d84-31e9-4624-9078-b4e5008f85c2	Terminal	Терминал	{"am": "string", "name": "string", "ipAddr
e92c556f-59e8-4fe6-adc1-becb7523d104	95802d84-31e9-4624-9078-b4e5008f85c2	Event	Событие	{"jid": "string", "propusk": "string", "dateT
23fe797d-19de-4772-b22e-a4e5988817c6	95802d84-31e9-4624-9078-b4e5008f85c2	UnblockDoor	Разблокировать дверь	{"operId": "String", "termId": "String", "ev
b12b07b1-db32-419d-b9ce-1863b104122c	95802d84-31e9-4624-9078-b4e5008f85c2	BlockDoor	Заблокировать дверь	{"operId": "String", "termId": "String", "ev
9f5afd3-8eb5-49a3-8f5e-c63d57fdeb5	95802d84-31e9-4624-9078-b4e5008f85c2	UnalarmTerminal	Снять тревогу с терминала	{"operId": "String", "termId": "String", "ev
941da904-f05f-4f61-b9f0-d9744254ebcd	95802d84-31e9-4624-9078-b4e5008f85c2	DisarmSensor	Снять с охраны датчик	{"operId": "String", "termId": "String", "ev
026f19a0-9e80-4672-8f79-51220e242093	95802d84-31e9-4624-9078-b4e5008f85c2	ArmSensor	Поставить на охрану датчик	{"operId": "String", "termId": "String", "ev
cd8fd350-05f7-4606-8270-8e4cbddc4b5c	95802d84-31e9-4624-9078-b4e5008f85c2	UnalarmSensor	Снять тревогу с датчика	{"operId": "String", "termId": "String", "ev
56e4ca7c-282d-452e-b8e3-fd2198754d0b	95802d84-31e9-4624-9078-b4e5008f85c2	CloseDoor	Закрыть дверь	{"operId": "String", "termId": "String", "ev
7ac6e6ab-0d13-4b46-9d2a-d401461ef4b9	95802d84-31e9-4624-9078-b4e5008f85c2	OpenDoor	Открыть дверь	{"operId": "String", "termId": "String", "ev
3270cb18-76ea-42f4-9e4b-1d485cfa773d	95802d84-31e9-4624-9078-b4e5008f85c2	DisarmTerminal	Снять с охраны терминал	{"operId": "String", "termId": "String", "ev

15 строк (0.000 s) Редактировать, Explain, Экспорт

Рисунок 37

Посмотреть, какие значения в сообщении приходят от сторонней системы можно в модуле менеджера очередей сообщений (Kafka) по адресу *http://имя сервера NEST-M:9090/ui/clusters/local/all-topics/nest.messages/messages?keySerde=String&valueSerde=String&limit=100*

Используя фильтры отсортировать, отфильтровать, и выбрать интересующую строку с значением (Value) (см. рисунок 38), содержащим:

"schemaCode": "message.Имя_модуля"

UI for Apache Kafka 56fa824 v0.7.1

Topics / nest.messages

Overview Messages Consumers Settings Statistics

Seek Type: Offset Offset Partitions: All items are selected. Key Serde: String Value Serde: String

Submit

Search + Add Filters

DONE 1 ms 65 KB 100 messages consu

Offset	Partition	Timestamp	Key Preview	Value Preview
1131	0	09.08.2023, 14:37:13	e354114a-6892-4054-99b4-09676036286c	{"schemaCode":"message.Senesys.Event","id":"e...

Рисунок 38

Если раскрыть строку, то можно увидеть полученное событие из сторонней системы (см. рисунок 39). В нем указано, какие поля и с какими значениями сообщение было получено.

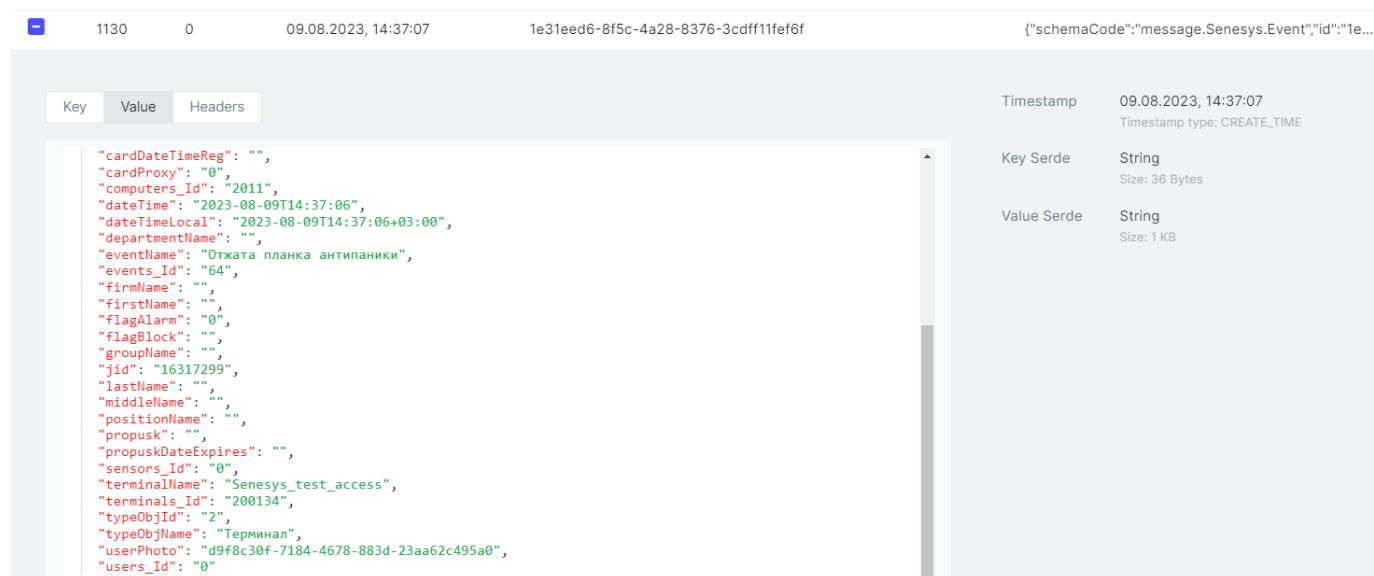


Рисунок 39

- с) количество ролей, участвующих обработке карточки инцидента. Должно быть указано в задании на карточку инцидента;
- д) ветвление (последовательность обработки карточки инцидента ролями). Должно быть указано в задании на карточку инцидента;
- е) шаги, описание (содержимое) шагов и их последовательность у каждой роли, а также перечень возможных действий у каждой роли.
Например, действия (открыть видео, показать объект на плане), наличие ветвления (эволюции), закрытие инцидента и т. д.

Перечень действий, а также параметры команд действий, поддерживаемых в ЦП NEST-M можно посмотреть в БД по адресу http://имя_сервера_NEST-M:5080. После этого:

- 1) открыть DB import;
- 2) выбрать «schemas»;
- ф) перечень полей, отображаемых в карточке инцидента, значения которых не получаются из сторонних систем. При этом они задаются статическими или в зависимости от условий возникновения карточки инцидента.
Например, поля, в которых указываются позывные радиостанций, телефоны вызываемых служб и прочее.

Перечень полей, должен быть указан в задании на карточку инцидента.

На основе данных, полученных из пунктов а)-ф), составить наглядную схему карточки инцидента.

Ниже как пример, приведена наглядная блок-схема инцидента «Нажата планка антипаники» (см. рисунок 40), По ней видно, что:

- карточку инцидента обрабатывают 3-роли (Оператор, Старший оператор, Начальник смены);
- есть два ветвления: Реальный инцидент или Ложное срабатывание. Также видно описание полей и их последовательность;
- карточка инцидента будет состоять из 6 прототипов.

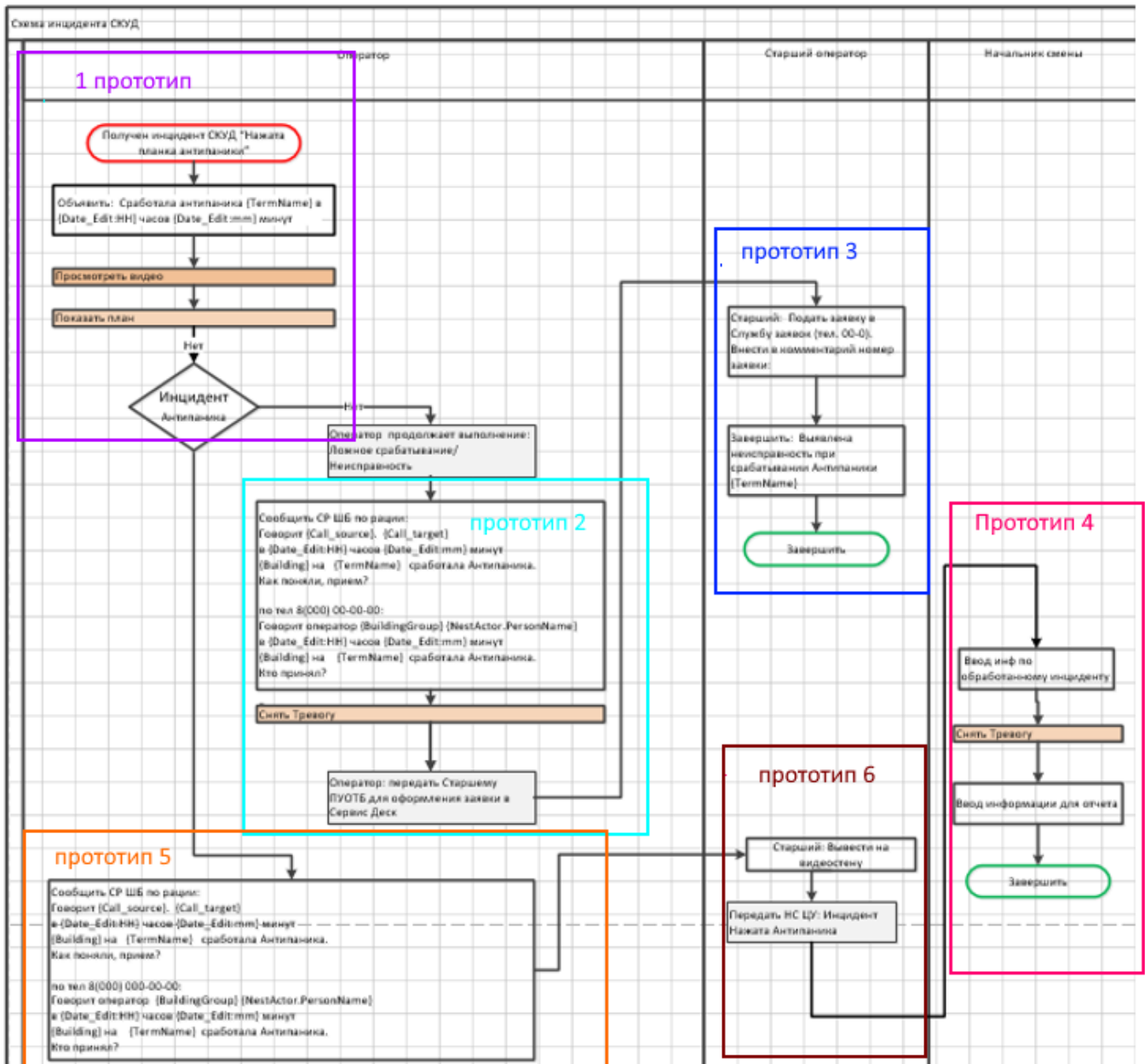


Рисунок 40

5.2 Создание зон ответственностей (scope)

Для того чтобы оператору ЦП NEST-M отображались на карте POI события, а в Модуле обработки инцидентов - инциденты, которые ему должны быть доступны, оператору должна быть выдана роль. Она должна включать в себя зону ответственности, в которую входят теги этих POI, событий и инцидентов прототипов.

Пример сообщения на добавление зоны ответственности с кодом "nest-user" включающую в себя теги "test" и "Интроскоп-ик:Wekey":

```
{
  "code": "nest-user",
  "filters": [
    {
      "templates": [
        {
          "template": "test"
        },
        {
          "template": "Интроскоп-ик:Wekey"
        }
      ]
    }
  ]
}
```

5.3 Автоматизация создания POI

Для создания или изменения POI через модуль менеджера очередей сообщений (Kafka) используется сообщение с кодом `poi.write`.

Структура сообщения:

```
{
  "schemaCode": "poi.write",
  "data": {
    "poiRead": #####,
    "poiWrite": #####,
    "navigateParameters": #####
  }
}
```

а) структура "poiRead":

"poiRead" определяет, как будут искаться POI для изменения. Особый режим позволяет создавать новый POI, если ни один не будет найден:

- 1) **id** - идентификатор обновляемого POI - прямая ссылка на POI для изменения;
- 2) **externalId** - внешний идентификатор POI;
- 3) **tags** - массив тегов, каждый из которых должен быть у POI.

Если в "**poiRead**" были указаны **externalId** и **tags**, но не было **id**, то если такие POI не были найдены, изменяться будет новый только что созданный POI.

б) структура "**poiWrite**":

- 1) **externalId** - внешний идентификатор, который, если не пуст, будет записан POI;
- 2) **tags** - массив тегов, который будет добавлен или удален у POI;
- 3) **meta** - записывает метаданные POI;
- 4) **data** - записывает данные POI;
- 5) **state** - данные, которые будут записаны в инцидент маппером. Известные свойства:
 - **logicalState.alarm** – флаг, определяющий тревожный статус POI;
 - **logicalState.disarm** – флаг, определяющий снятый с охраны статус POI;
 - **logicalState.area.id** - идентификатор области, к которой привязан POI;
 - **logicalState.area.name** - имя области, к которой привязан POI;
 - **name** - строковое имя POI;
 - **sylphideId** - **sylphideId** камеры Sylphide, которая привязана к POI;
- 6) **position** - **latitude**, **longitude**, **altitude** и/или **direction** изменяются, если были переданы;
- 7) **features** - словарь с командами, которые будут у POI. Известные свойства:
 - **featureCode** - ключ кнопки в словаре. При передаче значения по ключу, кнопка будет перезаписана. Если передать null, то кнопка будет удалена;
 - **caption** - группирующий заголовок кнопки;
 - **content** - текст внутри кнопки;
 - **confirmation** - текст подтверждения;
 - **commandCode** - код сообщения, который будет сгенерирован кнопкой;
 - **commandMappings** - маппинги для формирования сообщения;
- 8) **presentation** – изменение представления POI на карте;
- 9) **delete** - если true, то помечает POI как удаленный. Убирает его из всех выборок для клиента и для обновления.

с) структура "**navigateParameters**". Пример:

```

{
  "schemaCode": "poi.write",
  "id": "fa88f643-dc75-4689-be49-408009295b6f",
  "createdUtc": "2023-08-10T13:31:33.3973206Z",
  "meta": {
    "moduleCode": "Sylphide",
    "schemaCode": "Camera",
    "actor": "sylphide",
    "externalId": "1",
    "moduleId": "0b08c639-ca05-4dd1-8f40-22178c285f71",
    "schemaId": "a8d11a76-3eec-408d-ba73-bb68941d1165",
    "adapterId": "fa9d15bf-549e-48d5-b8af-597598ef764e",
    "triggerCode": "sylphide-poi"
  },
  "data": {
    "poiRead": {
      "tags": [
        "CBH:Sylphide"
      ],
      "externalId": "1"
    },
    "poiWrite": {
      "tags": [
        "CBH:Sylphide",
        "systemType:CBH",
        "icon:camera"
      ],
      "features": {
        "live": {
          "caption": "Показать видео",
          "content": "Показать живое видео",
          "commandCode": "command.Sylphide.ShowLiveVideoFeatu
re",
          "featureCode": "live",
          "commandMappings": [
            {
              "cases": {
                "": {
                  "deviceId": "{poi.state.data.device
Id}",
                  "userName": "{actor.account}"
                }
              }
            }
          ]
        }
      },
      "enarm": {
        "caption": "Поставить на охрану",
        "content": "Поставить на охрану",
        "commandCode": "poi.write",

```

```

"featureCode": "enarm",
"commandMappings": [
  {
    "cases": {
      "": {
        "poiRead.id": "{poi.id}",
        "poiWrite.state.logicalState.disarm
": false
      }
    }
  }
],
},
"unarm": {
  "caption": "Снять с охраны",
  "content": "Снять с охраны",
  "commandCode": "poi.write",
  "featureCode": "unarm",
  "commandMappings": [
    {
      "cases": {
        "": {
          "poiRead.id": "{poi.id}",
          "poiWrite.state.logicalState.disarm
": true
        }
      }
    }
  ]
},
"archive": null
},
"externalId": "1",
"state": {
  "name": "Стационарная камера 2",
  "deviceId": 1,
  "externalId": "1",
  "compositionId": 2,
  "logicalState": {
    "alarm": false,
    "disarm": false
  }
},
"presentation": {
  "properties": [
    {
      "name": "Название",
      "value": "Стационарная камера 2"
    }
  ]
}
]

```



```

    }
  }
}

```

Для автоматического создания системой сообщений используется Автоматизация модуля бизнес-логики.

Автоматизация (автоматизация, автоматика, автомат) - структура данных, которая позволяет реализовать бизнес-логику. Для чтения или настройки реализован стандартный CRUD.

Node (нод, нода, узел) - структура данных, которая позволяет хранить промежуточные состояния незавершенных автоматов, агрегировать данные и запускать запланированные реакции.

Структура данных автоматизации:

- 1) **id** - уникальный идентификатор, вычисляется при создании, реализует EntityBase;
- 2) **createdUtc** – дата/время создания, вычисляется при создании, реализует EntityBase;
- 3) **insertedUtc** - дата/время вставки в БД, вычисляется при вставке в БД, реализует EntityBase;
- 4) **editedUtc** - дата/время последнего изменения, вычисляется при изменении, реализует дата/время;
- 5) **deletedUtc** - дата/время удаления, вычисляется при удалении, реализует EntityBase;
- 6) **code** - уникальный строковый идентификатор, реализует CodeEntityBase, используется при создании, изменении, удалении или запросе;
- 7) **description** - текстовое описание, реализует CodeEntityBase;
- 8) **isEnabled** - флаг активности автомата;
- 9) **reactions []** - набор реакций:

- **code** - текстовый код реакции, записывается в метаданные;
- **isEnabled** - флаг активности реакции;
- **schemaCodeTriggers []** - набор кодов схем сообщений для запуска реакции;
- **dateTimeTriggers []** - набор макросов для вычислений даты срабатывания реакций;
- **conditions []** - условия срабатывания реакций, где :

isRequired - флаг необходимости условия,

isSufficient - флаг достаточности условия,

expression - макрос для вычисления проверяемой части,

equals [] - массив макросов, одно вычисленное значение которых должно совпасть с вычисленной проверяемой частью,

notEquals [] - массив макросов, все значения которых не должны совпадать с вычисленной проверяемой частью,
contains [] - массив макросов, один из которых должен содержать вычисленную проверяемую часть,
notContains [] - массив макросов, ни одно из вычисленных значений которых не должны содержать вычисленную проверяемую часть;

– **handlers []** - набор обработчиков реакции, где:

code - код обработчика реакции, записывается в метаданные,

isEnabled - флаг активности обработчика,

conditions [] - набор условий для выполнения обработчика, где:

isRequired - флаг необходимости условия,

isSufficient - флаг достаточности условия,

expression - макрос для вычисления проверяемой части,

isNull - проверка на null (не реализовано),

isNotNull - проверка на не null (не реализовано),

equals [] - массив макросов, одно вычисленное значение которых должно совпасть с вычисленной проверяемой частью,

notEquals [] - массив макросов, все значения которых не должны совпадать с вычисленной проверяемой частью,

contains [] - массив макросов, один из которых должен содержать вычисленную проверяемую часть,

notContains [] - массив макросов, ни одно из вычисленных значений которых не должны содержать вычисленную проверяемую часть,

createNode - флаг определяет, будет ли создана нода при реакции на сообщение,

updateNodes - флаг определяет, будут ли обновляться ноды при реакции на сообщение,

writeNodes - флаг определяет, что при реакции на сообщения будут обновляться все найденные ноды, но если ни одна не нашлась, новая нода будет создана,

terminateNodes - флаг определяет, что найденные обновленные ноды будут закрыты при реакции на сообщение,

searchNodeKeyMacro - макрос для вычисления ключа нод, по которому они будут искаться для обновления,

writeNodeKeyMacro - макрос, вычисляющий новое значение ключа обновляемых нод. Если вычисленное значение пустое, обновление ключа не производится,

writeMappings [] - маппинги для изменений данных найденных нод:

order - порядок,

description - описание,

switch - часть оператора Switch-Case,

cases - часть оператора Switch-Case,

resultSchemaCode - макрос для вычисления кода схемы сообщения, которое будет отправлено в очередь при срабатывании обработчика реакции,

resultMappings[] - маппинги для формирования данных отправляемого сообщения

order – порядок,

description – описание,

switch – часть оператора Switch-Case,

cases – часть оператора Switch-Case.

Пример:

```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "createdUtc": "2023-08-07T10:34:40.575Z",
  "insertedUtc": "2023-08-07T10:34:40.575Z",
  "editedUtc": "2023-08-07T10:34:40.575Z",
  "deletedUtc": "2023-08-07T10:34:40.575Z",
  "code": "string",
  "description": "string",
  "isEnabled": true,
  "reactions": [
    {
      "code": "string",
      "isEnabled": true,
      "schemaCodeTriggers": [
        "string"
      ],
      "dateTimeTriggers": [
        "string"
      ],
      "conditions": [
        {
          "isRequired": true,
          "isSufficient": true,
          "expression": "string",
          "isNull": ???,
          "isNotNull": ???,
          "equals": [
            "string"
          ],
          "notEquals": [
            "string"
          ],
          "contains": [
            "string"
          ],
          "notContains": [
```

```
        "string"
      ]
    }
  ],
  "handlers": [
    {
      "code": "string",
      "isEnabled": true,
      "searchNodeKeyMacro": "string",
      "conditions": [
        {
          "isRequired": true,
          "isSufficient": true,
          "expression": "string",
          "equals": [
            "string"
          ],
          "notEquals": [
            "string"
          ],
          "contains": [
            "string"
          ],
          "notContains": [
            "string"
          ]
        }
      ],
      "createNode": true,
      "updateNodes": true,
      "writeNodes": true,
      "terminateNodes": true,
      "writeNodeKeyMacro": "string",
      "writeMappings": [
        {
          "order": 0,
          "description": "string",
          "switch": "string",
          "cases": "string"
        }
      ],
      "resultSchemaCode": "string",
      "resultMappings": [
        {
          "order": 0,
          "description": "string",
          "switch": "string",
          "cases": "string"
        }
      ]
    }
  ]
}
```

```

    }
  ]
}

```

Для создания, обновления сообщения с SchemaCode - **poi.write**, в автоматизации (автомате) обязательно указывается:

- набор реакций (**reaction []**);
- набор кодов схем сообщений для запуска реакции (**reactions [].schemaCodeTriggers**);
- код схемы сообщения, которое будет отправлено в очередь при срабатывании обработчика реакции (**reactions [].handlers [].resultSchemaCode**), в случае создания/обновления POI - **poi.write**;
- маппинги для формирования данных отправляемого сообщения (**reactions [].handlers [].resultMappings []**).

Кодом схемы сообщения для запуска реакции (**schemaCodeTriggers**) может являться любое сообщение с любым кодом. Зачастую кодом схемы сообщения для запуска реакции является код (схема) входящего сообщения от присоединенной системы.

Результат выполнения автоматизации бизнес логики по созданию POI является появление в Kafka сообщения с названием кодом схемы "**schemaCode**": "**poi.write**". Если сообщение с названием "**schemaCode**": "**poi.write**" имеет правильную схему, то в БД запишется POI. Сигналом, что POI записалось в БД является появление в Kafka сообщения со схемой "**schemaCode**": "**poi.result**", показывающий какие изменения (создание, обновление) произошли касемо POI.

Проверить, что POI записалось в БД, можно посмотреть в самой БД, по адресу <http://имя сервера NEST-M:5080/>. После нужно:

- 1) Выбрать в DB «monitor»;
- 2) выбрать pois;
- 3) в таблице должна быть добавленная POI отфильтровав по id (найдя ID в схеме "**schemaCode**": "**poi.write**" в Kafka).

При написании POI нужно учесть, что название в **features** и наименование в **featureCode** должны совпадать, иначе сообщение не будет работать (см. рисунок 41):

```
"poiWrite.features": {  
  "live": {  
    "featureCode": "live",  
    "commandCode": "command.Sylphide.ShowLiveVideoFeature",  
    "caption": "Видео",  
    "content": "Показать online видео",  
    "commandMappings": [  

```

Рисунок 41

Если у нас подразумевается массив данных, в котором нужно брать какое-либо значение из функций, то нужно пометить переменную как макрос квадратными скобками [] (см. рисунок 42).

```
"commandMappings": [  
  {  
    "cases": {  
      "": {  
        "poiRead.id": "{poi.id}",  
        "poiWrite.state.logicalState.disarm": true,  
        "poiWrite.state.logicalState.alarm": false,  
        "apIds[]": "{poi.state.data.externalId}",  
        "accessPointMode": "LOCKED",  
        "poiWrite.data.device_name": "{poi.state.data.name}",  

```

Рисунок 42

Если мы запишем в виде `"apIds": [{"poi.state.data.externalId}"]`, то результатом будет строка (см. рисунок 43):

```
"data": {  
  "apIds": [  
    "{poi.state.data.externalId}"  
  ],  

```

Рисунок 43

И это даст нам ошибку синтаксическую, так как ожидается увидеть число, а не строчку слов (см. рисунок 44).

```

    "data": {
      "methodName": "SetUpModeAccessPoint",
      "message": "ERROR 6 SYNTAX ERROR"
    }
  }
}

```

Рисунок 44

Все названия, которые не указаны, но встречаются в других примерах создания POI, являются произвольными названиями, которые можно задавать самостоятельно. Например, **poiWrite.state.testovaja.zapis** – будет работать совершенно нормально, и запишет в state => testovaja => zapis значение.

Ниже приведен пример автомата на создание POI, где кодом схемы запуска реакции является код входящего сообщение (код схемы - **message.Wekey.Event**) от присоединенной системы Wekey.

Код схемы и схема сообщения от присоединенной системы Wekey:

```

{
  "schemaCode": "message.Wekey.Event",
  "id": "46a60074-c18e-4f7f-add0-3f8d88bba45a",
  "createdUtc": "2023-08-20T18:16:42.3711988Z",
  "meta": {
    "moduleCode": "Wekey",
    "schemaCode": "Event",
    "actor": "wekey",
    "externalId": "cb1275bb-7009-40c7-80f5-6bda0457bd45",
    "moduleId": "dd57ace1-2c0a-4884-9178-9727c4f3b1ed",
    "schemaId": "7c1ec469-9e89-4c6a-a025-afb34a3a178b",
    "adapterId": "818b7523-fc2a-4133-b4f8-4b93248076dd"
  },
  "data": {
    "topic": [
      "Introscopy",
      "LimitExceeded"
    ],
    "utcTime": "2023-06-17T16:27:06.55432Z",
    "source": {
      "id": "009"
    },
    "data": {
      "picture": "9723dbaf-8c01-4d97-b362-4df17d9a5550",
      "account": "Иванов А.В.",
      "result": "1"
    }
  }
}
}
}

```



```

        "Интроскоп-ик:Wekey"
    ],
    "poiWrite.externalId": "{data.source.id}",
    "poiWrite.tags[]": [
        "Интроскоп-ик:Wekey",
        "systemType:Интроскоп-ик",
        "icon:device",
        "poiId:{poiId}"
    ],
    "poiWrite.state.externalId":
"{data.source.id}",
    "poiWrite.state.name": "Интроскоп №
{data.source.id}",
    "poiWrite.state.datetime_event":
"{data.utcTime}",
    "poiWrite.state.account":
"{data.data.account}",
    "poiWrite.state.result": "{data.data.result}",
    "poiWrite.state.picture":
"{data.data.picture}",
    "poiWrite.state.logicalState.alarm": true,
    "poiWrite.presentation.properties[]": [
        {
            "name": "Название устройства",
            "value": "Интроскоп № {data.source.id}"
        },
        {
            "name": "Система",
            "value": "Wekey"
        }
    ],
    "poiWrite.features": {
        "live": {
            "featureCode": "live",
            "commandCode":
"command.Sylphide.ShowLiveVideoFeature",
            "caption": "видео",
            "content": "Показать живое видео",
            "commandMappings": [
                {
                    "order": 1,
                    "description": "description-1",
                    "cases": {
                        "": {
                            "deviceId":
"{poi.state.data.deviceId}",
                            "userName":
"{actor.account}"
                        }
                    }
                }
            ]
        }
    }
}

```


5.4 Автоматизация создания события

События на плане можно отображать в нескольких местах:

- на вкладке «События» выделенного объекта на плане. Для этого необходимо чтобы объект (POI) уже был добавлен в БД;
- в таблице событий.

Добавление отображения события в таблице «Событий» и на вкладке «События» у POI производится путем написания автомата на создание сообщения создания события.

Для создания событий используется сообщение с кодом **event.new.xxx**, где xxx - условный код события.

Структура сообщения:

```
{
  "schemaCode": "event.new.xxx",
  "data": {
    "eventWrite": #####
  }
}
```

EventWrite - структура данных, определяющая данные создаваемого события. Она содержит следующие поля:

- **eventId** - внешний идентификатор события, опциональное поле.
- **source** - источник сообщения, опциональное поле.
- **name** - название события. Если значение не указано, вместо него используется код события.
- **status** - массив строк с обозначением события. Поддерживаются зеленый кружок «arm», красный кружок «alarm» и серый кружок «disarm».
- **eventDate** - дата и время возникновения события с передачей часового пояса
- **tags** - массив тегов, который будет присвоен событию
- **data** - сторонняя информация о событии
- **presentation** - представление POI и событий

Ниже приведен пример автомата на создание сообщения создания события, где кодом схемой сообщения реакции является код схема **"poi.result"**:

```
{
  "code": "wekey-events",
  "description": "wekey-events",
  "isEnabled": true,
  "reactions": [
```

```

{
  "code": "reaction-message.Wekey.Event-alarm",
  "isEnabled": true,
  "schemaCodeTriggers": [
    "poi.result"
  ],
  "conditions": [
    {
      "isRequired": true,
      "expression": "{data.poi.tags}",
      "contains": [
        "Интроскоп-ик:Wekey"
      ]
    },
    {
      "isRequired": true,
      "expression": "{data.poiWrite.state.logicalState.alarm}",
      "equals": [
        "True"
      ]
    },
    {
      "isRequired": true,
      "expression": "{data.poi.state.data.logicalState.disarm}",
      "equals": [
        "False"
      ]
    }
  ],
  "handlers": [
    {
      "code": "wekey-event-alarm",
      "isEnabled": true,
      "resultSchemaCode": "event.new.wekey-alarm",
      "resultMappings": [
        {
          "description": "отображение_тревожного_события",
          "cases": {
            "": {
              "eventWrite.tags[]": [
                "Интроскоп-ик:Wekey",
                "systemType:Интроскоп-ик",
                "poiId:{data.poi.id}"
              ],
              "eventWrite.eventId": "no_Id",
              "eventWrite.source":
"{data.poiWrite.state.name}",
              "eventWrite.name": "Превышен порог",
              "eventWrite.eventDate":
"{data.poiWrite.state.datetime_event}",

```



```
"reactionCode": "reaction-message.Wekey.Event-alarm",
"handlerCode": "wekey-event-alarm"
},
"data": {
  "eventWrite": {
    "name": "Превышен порог",
    "source": "Интроскоп № 009",
    "status": [
      "alarm"
    ],
    "tags": [
      "Интроскоп-ик:Wekey",
      "systemType:Интроскоп-ик",
      "poiId:1f57566a-0e6c-465f-bd2a-732580804983"
    ],
    "eventId": "no_Id",
    "eventDate": "2023-06-17T16:27:06.55432Z",
    "presentation": {
      "content": [
        {
          "source": "9723dbaf-8c01-4d97-b362-4df17d9a5550",
          "template": "image"
        }
      ],
      "properties": [
        {
          "name": "Время",
          "value": "2023-06-17T16:27:06.55432Z"
        },
        {
          "name": "Источник",
          "value": "Интроскоп № 009"
        },
        {
          "name": "Событие",
          "value": "Превышен порог"
        },
        {
          "name": "Оператор",
          "value": "Иванов А.В."
        }
      ]
    }
  }
}
}
}
}
```

5.5 Автоматизация создания инцидента

Для создания инцидентов используется сообщение с кодом схемы `accident.new.xxx`, где `xxx` - код прототипа создаваемого инцидента. Сигналом, что инцидент создан является появление в Kafka сообщения с кодом схемой `accident.accidentNew.xxx`.

Пример сообщения:

```
{
  "schemaCode": "accident.new.evo-test",
  "data": {
    "accidentWrite": {
      "tags": [
        "test",
        "evo-test"
      ],
      "data": {
        "prefix": "prefix",
        "test": "test"
      }
    }
  }
}
```

Структура:

- **schemaCode** - код сообщения в формате `accident.new.xxx`;
- **data.accidentWrite** - структура данных, по которой создается инцидент, где
- **tags** - тэги, которые будут записаны инциденту при его создании;
- **data** - данные, которые будут скопированы инциденту при его создании.

Пример автоматизации:

```
{
  "code": "wekey-accidents",
  "description": "wekey-accidents",
  "isEnabled": true,
  "reactions": [
    {
      "code": "reaction-poi.result-create-new-accidents",
      "isEnabled": true,
      "schemaCodeTriggers": [
        "poi.result"
      ],
      "conditions": [
        {
          "isRequired": true,
          "expression": "{data.poi.tags}",

```



```

    "contains": [
      "Интроскоп-ик:Wekey"
    ]
  },
  {
    "isRequired": true,
    "expression": "{data.poiWrite.state.logicalState.alarm}",
    "equals": [
      "True"
    ]
  },
  {
    "isRequired": true,
    "expression": "{data.poi.state.data.logicalState.disarm}",
    "equals": [
      "False"
    ]
  }
],
"handlers": [
  {
    "code": "wekey-create-new-accident",
    "isEnabled": true,
    "resultSchemaCode": "accident.new.wekey-prototype-new-
accident",
    "resultMappings": [
      {
        "description": "создание_нового_инцидента_wekey",
        "cases": {
          "": {
            "accidentWrite.tags[]": [
              "Интроскоп-ик:Wekey",
              "systemType:Интроскоп-ик",
              "poiId:{data.poi.id}"
            ],
            "accidentWrite.data.datetime_event":
"{data.poiWrite.state.datetime_event}",
            "accidentWrite.data.device_name":
"{data.poiWrite.state.name}",
            "accidentWrite.data.account_wekey":
"{data.poiWrite.state.account}",
            "accidentWrite.data.picture":
"{data.poiWrite.state.picture}",
            "accidentWrite.data.text_message":
"Обнаружение запрещенного предмета",
            "accidentWrite.data.poiId": "{data.poi.id}"
          }
        }
      }
    ]
  }
]

```

```

    }
  ]
}

```

Внимание: чтобы данные из сообщения инцидента (код **accident.accidentNew.xxx**), передались в прототип, необходимо, чтобы код прототипа xxx был указан в коде сообщения инцидента.

5.6 Создание прототипов

Чтобы инцидент отобразился у оператора в модуле интерфейсов работы с инцидентами, необходимо, чтобы был настроен прототип.

Для настройки прототипов используется CRUD /services/control/prototypes.

Структура:

- 1) **code** - уникальный код прототипа, по которому происходит редактирование прототипа и создание и изменение инцидента;
- 2) **description** - текстовое описание, которое используется при настройке;
- 3) **longName** – длинное название инцидента;
- 4) **shortName** - короткое название инцидента;
- 5) **soundnotificationsource** - ссылка или ID filestore звукового уведомления;
- 6) **dataSchema** - заявленные администратором свойства инцидента и их текстовое описание;
- 7) **allowTermination** - флаг, разрешающий завершение инцидента, если у него есть варианты развития;
- 8) **terminationComment** - макрос комментария завершения по умолчанию;
- 9) **prioriotyCode** - код приоритета, с которым будет создан инцидент;
- 10) **panels []** - набор панелей инцидента:
 - **displayorder** - порядковый номер панели;
 - **handler** - флаг, определяющий, что панель является шагом;
 - **content** - содержимое панели;
 - **confirmation** - макрос подтверждения выполнения шага;
 - **commandCode** - код сообщения, которое будет сгенерировано при выполнении команды;
 - **commandMappings** - параметры генерируемого сообщения;
- 11) **evolutions** - набор вариантов развития инцидента:
 - **code** - код варианта развития, используется при вызове **accident.merge**;
 - **displayOrder** - порядковый номер варианта эволюции;
 - **caption** - макрос подсказки варианта эволюции;

- **mappings** - формирование данных инцидента при его развитии;
- **priorityCode** - код приоритета, который будет назначен инциденту при обновлении;
- **prototypeCode** - код прототипа, который будет у инцидента после обновления.

Ниже приведена схема прототипа с кодом **wekey-prototype-new-accident** с эволюцией этого прототипа в прототип с кодом **wekey-prototype-gun**:

```
{
  "code": "wekey-prototype-new-accident",
  "dataSchema": {
    "data.datetime_event": "",
    "data.device_name": "",
    "data.poiId": "",
    "data.account_wekey": "",
    "data.picture": "",
    "data.text_message": ""
  },
  "soundNotificationSource": "mixkit-classic-short-alarm-993.wav",
  "longName": "{accident.data.text_message}\r\nМесто:
{accident.data.device_name}\r\nВремя обнаружения:
{accident.data.datetime_event}\r\nФИО оператора
интроскопа:{accident.data.account_wekey}",
  "shortName":
"{accident.data.text_message}\r\n{accident.data.device_name}",
  "priorityCode": "default",
  "panels": [
    {
      "displayOrder": 2,
      "handler": false,
      "content": [
        {
          "text": "Посмотреть архивное видео"
        }
      ],
      "commandCode": "command.Sylphide.ShowArchiveVideoFeature",
      "commandMappings": [
        {
          "cases": {
            "": {
              "deviceId": "{poi.state.data.deviceId}",
              "archivePoint": "{accident.data.datetime_event}",
              "userName": "{actor.account}"
            }
          }
        }
      ]
    }
  ],
}
```

```

{
  "displayOrder": 1,
  "handler": true,
  "content": [
    {
      "text": "Просмотреть изображение для определения типа
запрещенного предмета",
      "template": "image",
      "source": "{accident.data.picture}"
    }
  ]
},
{
  "displayOrder": 2,
  "handler": true,
  "content": [
    {
      "text": "Выбрать тип обнаруженного предмета"
    }
  ]
}
],
"evolutions": [
  {
    "displayOrder": 1,
    "code": "gun",
    "caption": "Огнестрельное оружие",
    "mappings": [
      {
        "cases": {
          "": {
            "datetime_event":
"{accident.data.datetime_event}",
            "device_name": "{accident.data.device_name}",
            "poiId": "{accident.data.poiId}",
            "account_wekey": "{accident.data.account_wekey}",
            "picture": "{accident.data.picture}",
            "text_message": "Обнаружено огнестрельное
устройство"
          }
        }
      }
    ]
  },
  {
    "priorityCode": "default",
    "prototypeCode": "wekey-prototype-gun"
  }
]
}

```

Схема прототипа с кодом **wekey-prototype-gun**:

```
{
  "code": "wekey-prototype-gun",
  "dataSchema": {
    "data.datetime_event": "",
    "data.device_name": "",
    "data.poiId": "",
    "data.account_wekey": "",
    "data.picture": "",
    "data.text_message": ""
  },
  "soundNotificationSource": "",
  "longName":
  "{accident.data.text_message}\r\nМесто:{accident.data.device_name}\r\nВремя
  обнаружения: {accident.data.datetime_event}\r\nФИО оператора интроскопа:
  {accident.data.account_wekey}",
  "shortName":
  "{accident.data.text_message}\r\n{accident.data.device_name}",
  "priorityCode": "default",
  "panels": [
    {
      "displayOrder": 2,
      "handler": false,
      "content": [
        {
          "text": "Посмотреть архивное видео"
        }
      ],
      "commandCode": "command.Sylphide.ShowArchiveVideoFeature",
      "commandMappings": [
        {
          "cases": {
            "": {
              "deviceId": "{poi.state.data.deviceId}",
              "archivePoint": "{accident.data.datetime_event}",
              "userName": "{actor.account}"
            }
          }
        }
      ]
    },
    {
      "displayOrder": 3,
      "handler": false,
      "content": [
        {
          "text": "Показать на карте"
        }
      ]
    }
  ]
}
```

```

"commandCode": "client.navigate",
"commandMappings": [
  {
    "cases": {
      "": {
        "navigateParameters.userName": "{actor.account}",
        "navigateParameters.poiId":
"{accident.data.poiId}"
      }
    }
  }
],
},
{
  "displayOrder": 0,
  "handler": false,
  "content": [
    {
      "text": "{accident.data.text_message}",
      "template": "image",
      "source": "{accident.data.picture}"
    }
  ]
},
{
  "displayOrder": 3,
  "handler": true,
  "content": [
    {
      "text": "СНЯТЬ тревоги на {accident.data.device_name}"
    }
  ],
  "commandCode": "poi.write",
  "commandMappings": [
    {
      "cases": {
        "": {
          "poiRead.id": "{accident.data.poiId}",
          "poiWrite.state.logicalState.alarm": false
        }
      }
    }
  ]
}
],
"evolutions": []
}

```

Внимание: чтобы данные из сообщения инцидента (код **accidentNew.xxx**), передались в прототип, необходимо, чтобы код прототипа xxx был указан в коде сообщения инцидента.

5.7 Создание ролей, пользователей в ЦП NEST-M

По умолчанию в ЦП NEST-M заведены следующие учетные записи:

- nest-admin;
- nest-user.

Отличие учетной записи nest-admin от nest-user, в том, что у nest-admin есть права, позволяющие:

- добавлять, изменять и удалять устройства на карте;
- добавлять, изменять и удалять 3D-объекты на карте;
- добавлять, изменять и удалять периметры на карте;
- открыть менеджер конфигураторов для настройки системы.

Создание пользователей, привязывание ролей и зон ответственности осуществляется в менеджере конфигураторов, который располагается по адресу *http://имя_сервера_NEST-M/setup*.

- Логин: nest-admin;
- Пароль: nest-admin.

5.7.1 Создание ролей

Для создания роли необходимо:

1) выбрать вкладку «Роли учётных записей» в левом меню. Откроется таблица с ролями, которые есть в системе (см. рисунок 45);

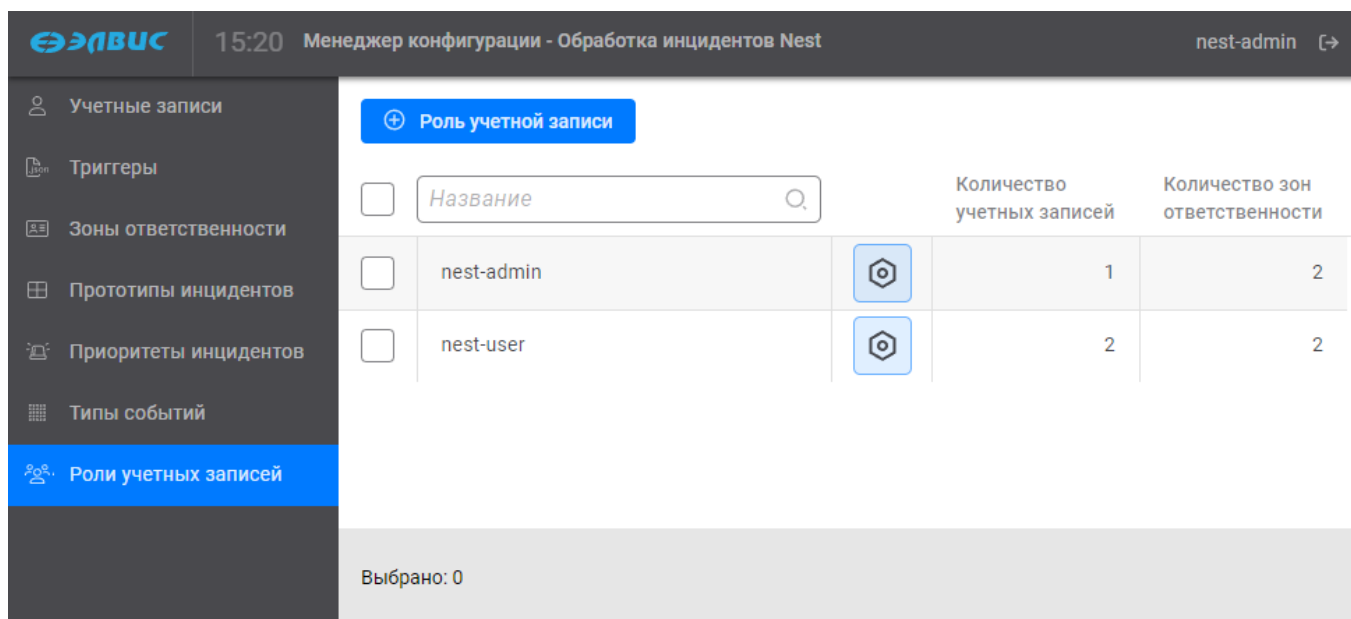


Рисунок 45

2) нажать на кнопку «+ Роль учётной записи» для добавления роли. Откроется форма для ввода информации о новой роли (см. рисунок 46);

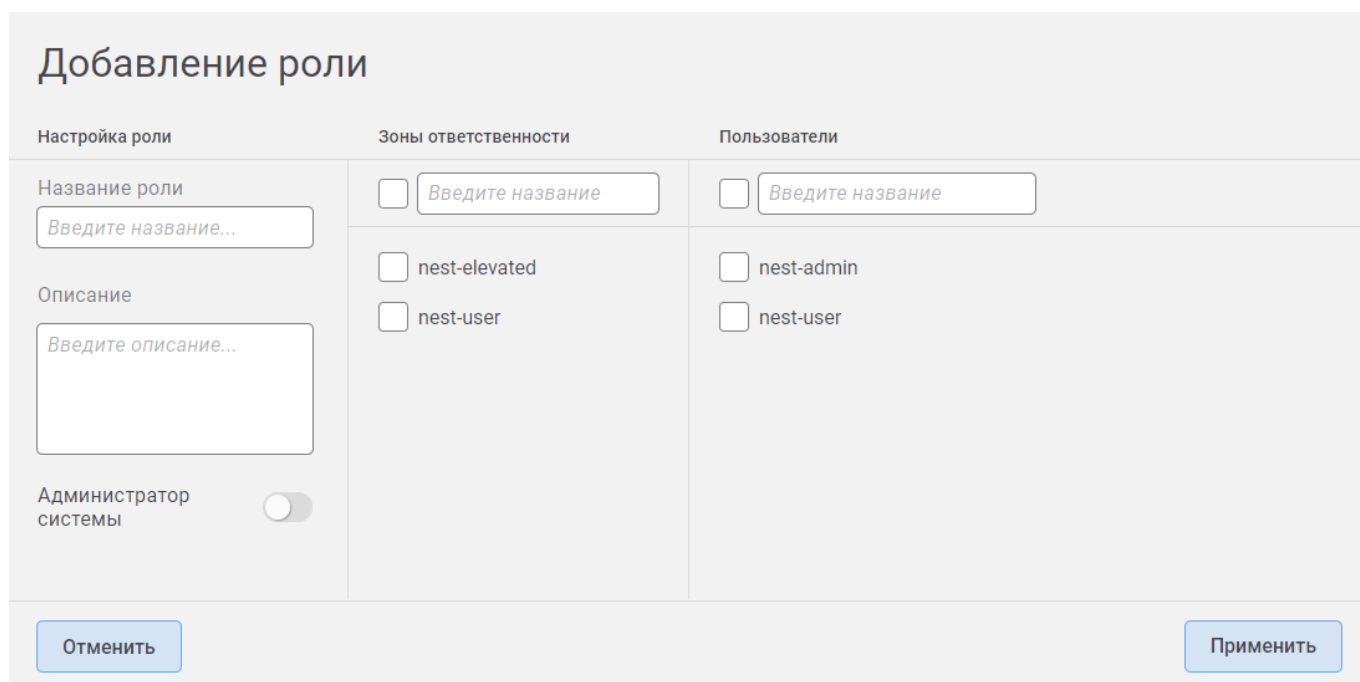


Рисунок 46

5.7.2 Создание пользователей

Для создания пользователя необходимо:

1) выбрать вкладку «Учётные записи» в левом меню. Откроется таблица с учётными записями, которые есть в системе (см. рисунок 47);

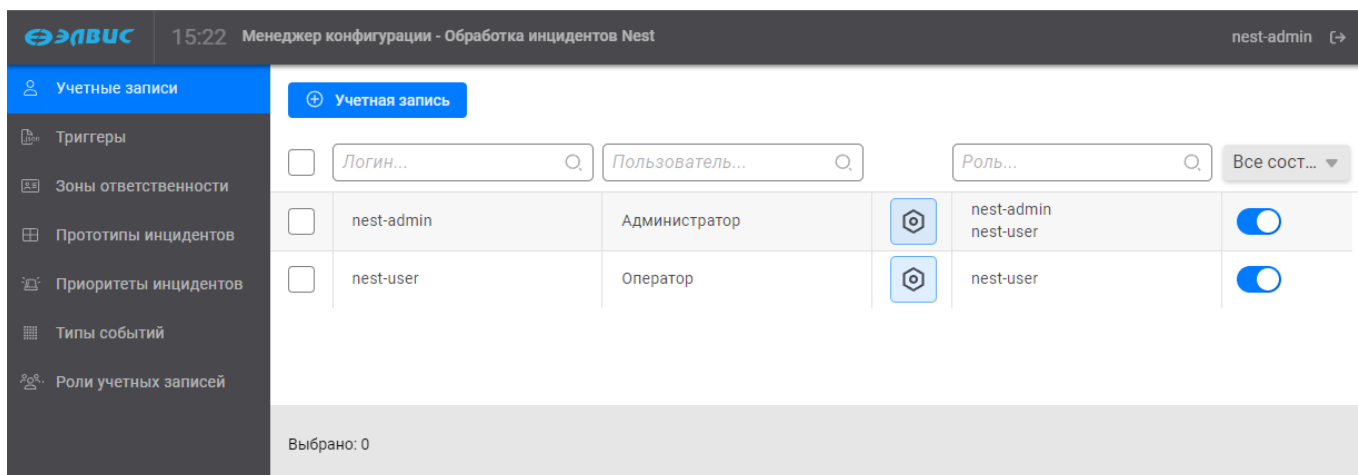


Рисунок 47

2) нажать на кнопку «+ Учётная запись» для добавления учётной записи. Откроется форма для ввода информации новой учётной записи (см. рисунок 48);

Добавление учётной записи

Логин	Имя персоны
<input type="text" value="Введите логин"/>	<input type="text" value="Введите имя персоны"/>
Роль	
<input type="text" value="Выберите роль"/>	
Пароль	Подтверждение
<input type="text" value="Введите пароль"/>	<input type="text" value="Введите пароль"/>

Рисунок 48

5.8 Просмотр кодов и схем сообщений

Для написания автоматизации модуля бизнес-логики необходимо знать коды схем сообщений, схемы этих сообщений, значения, передаваемые в этих сообщениях. Зачастую в первую очередь возникает необходимость посмотреть коды схем и схему входящего сообщения от присоединенной системы.

Схему входящего сообщения (если данный тип сообщения присоединенной системы уже приходил в модуль менеджера очередей сообщений (Kafka)), можно посмотреть в БД в строке в таблице схем сообщений (см. рисунок 49):

- 1) открыть `http://имя_сервера_NEST-M:5080/`;
- 2) выбрать в DB «import»;
- 3) выбрать «schemas»;
- 4) посмотреть схему в столбцах `properties` и `descriptions`, где `schema_code="event"`.

SELECT * FROM "schemas" where module_id='95802d84-31e9-4624-9078-b4e5008f85c2' LIMIT 50

id	module_id	schema_code	display_name	
43e9e100-f99c-4b7b-a993-c3eb95c11c0c	95802d84-31e9-4624-9078-b4e5008f85c2	Alarm	Тревога	{"compId": "string", "termId": "string", "ev
306660d8-5a81-44dd-825a-da8f925ee645	95802d84-31e9-4624-9078-b4e5008f85c2	Computer	Компьютер	{"ipAddress": "string", "computers_Id": "st
65d55f61-fe17-4c55-8067-a22c4f30b796	95802d84-31e9-4624-9078-b4e5008f85c2	Sensor	Сенсор	{"name": "string", "type": "string", "typeAI
e309c32b-fd37-49ab-99f5-9a53b708af50	95802d84-31e9-4624-9078-b4e5008f85c2	ArmTerminal	Поставить на охрану терминал	{"operId": "String", "termId": "String", "ev
08187f33-de45-42e3-9595-6b3ff6474299	95802d84-31e9-4624-9078-b4e5008f85c2	Terminal	Терминал	{"arm": "string", "name": "string", "ipAddr
e92c556f-59e8-4fe6-adc1-becb7523d104	95802d84-31e9-4624-9078-b4e5008f85c2	Event	Событие	{"jId": "string", "propusk": "string", "dateT
23fe797d-19de-4772-b22e-a4e5988817c6	95802d84-31e9-4624-9078-b4e5008f85c2	UnblockDoor	Разблокировать дверь	{"operId": "String", "termId": "String", "ev
b12b07b1-db32-419d-b9ce-1863b104122c	95802d84-31e9-4624-9078-b4e5008f85c2	BlockDoor	Заблокировать дверь	{"operId": "String", "termId": "String", "ev
9f55afd3-8eb5-49a3-8f5e-c63d57fdeb55	95802d84-31e9-4624-9078-b4e5008f85c2	UnalarmTerminal	Снять тревогу с терминала	{"operId": "String", "termId": "String", "ev
941da904-f05f-4f61-b9f0-d9744254ebcd	95802d84-31e9-4624-9078-b4e5008f85c2	DisarmSensor	Снять с охраны датчик	{"operId": "String", "termId": "String", "ev
026f19a0-9e80-4672-8f79-51220e242093	95802d84-31e9-4624-9078-b4e5008f85c2	ArmSensor	Поставить на охрану датчик	{"operId": "String", "termId": "String", "ev
cd8fd350-05f7-4606-8270-8e4cbbdc4b5c	95802d84-31e9-4624-9078-b4e5008f85c2	UnalarmSensor	Снять тревогу с датчика	{"operId": "String", "termId": "String", "ev
56e4ca7c-282d-452e-b8e3-fd2198754d0b	95802d84-31e9-4624-9078-b4e5008f85c2	CloseDoor	Закрыть дверь	{"operId": "String", "termId": "String", "ev
7ac66eab-d013-4b46-9d2a-d401461ef4b9	95802d84-31e9-4624-9078-b4e5008f85c2	OpenDoor	Открыть дверь	{"operId": "String", "termId": "String", "ev
3270cb18-76ea-42f4-9e4b-1d485cfa773d	95802d84-31e9-4624-9078-b4e5008f85c2	DisarmTerminal	Снять с охраны терминал	{"operId": "String", "termId": "String", "ev

Рисунок 49

Также схему входящего сообщения от сторонней системы и значения, получаемые в нем, можно посмотреть в модуле менеджера очередей сообщений (Kafka) (см. рисунок 50):

- 1) открыть `http://имя_сервера_NEST-M:9090/`;
- 2) выбрать в local «Topics» ;
- 3) выбрать сообщения `nest.messages`;
- 4) выбрать интересующую строку с Value, содержащего `"schemaCode" : "message Имя модуля присоединенной системы"`;
- 5) раскрыв строку, увидим полученное событие из присоединенной системы, какие поля с какими значениями есть в данном входящем сообщении.

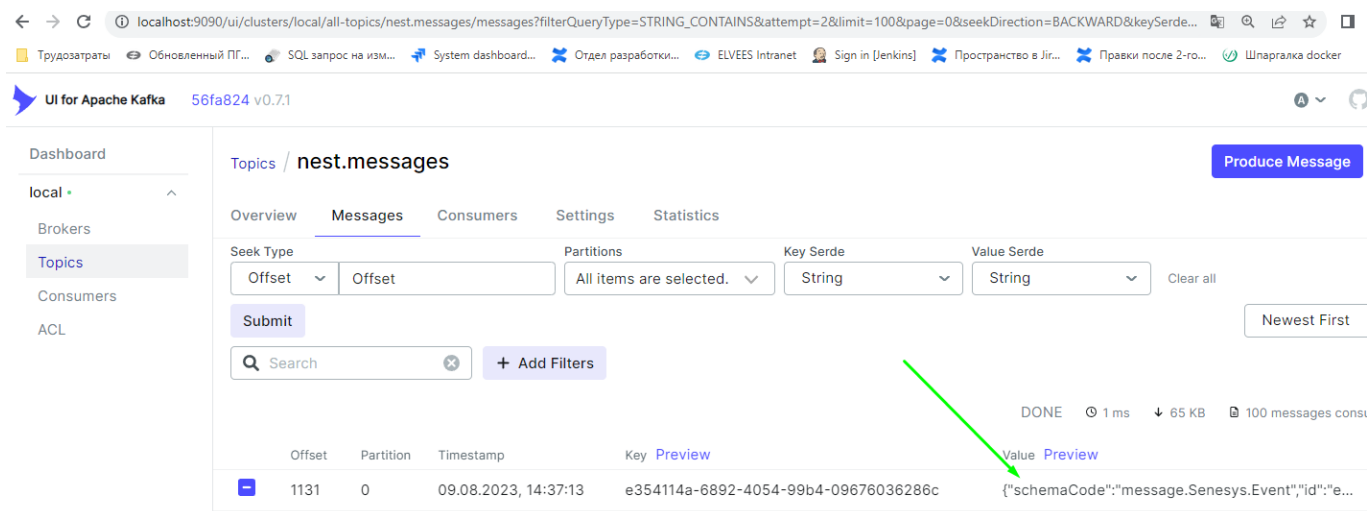


Рисунок 50

Схемы сообщений и значения, передаваемые в этих сообщениях, получаемые в результате выполнения реакций, также можно просмотреть в Kafka.

5.9 Инструкция по выявлению неисправностей

5.9.1 Возникновение нескольких сообщений в Kafka вместо одного

При написании и проверки работы автоматов бывает ситуация, что вместо одного приходит несколько событий инцидентов.

Возможная причина: такая проблема возникает тогда, когда меняют название кода автомата без удаления автомата со старым названием кода-схемы.

Решение: посмотреть в Kafka на наличие нескольких сообщений с одинаковой кодом-схемой. При наличии нескольких сообщений с одним кодом-схемы выписать названия кодов и посмотреть в БД по пути:

- 1) открыть *http://имя сервера NEST-M:5080/*;
- 2) выбрать в DB «business»;
- 3) выбрать «automations»;
- 4) отфильтровать по условию **deleted_utc is null**.

Для деактивации ненужного автомата нужно пометить его как удаленный. Для этого в строке автомата поле «deleted_utc» сделать не равным NULL. После этого сгенерировать заново сообщение с кодом-схемой и проверить, что в Kafka приходит только одно сообщение с ожидаемой кодом-схемой.

5.9.2 Отсутствует инцидент в модуле интерфейсов обработки инцидента

5.9.2.1 Отсутствие сообщения в kafka о создании инцидента

Если сообщение на создание инцидента ("accident.new.xxx") есть в Kafka, а инцидент так и не создан (отсутствует сообщение с SchemaCode "accident accident.new.xxx"), необходимо:

1) посмотреть логи Control-service на наличие в них записей об ошибках (fail/error) и наличие предупреждений (warn) (см. рисунок 51);

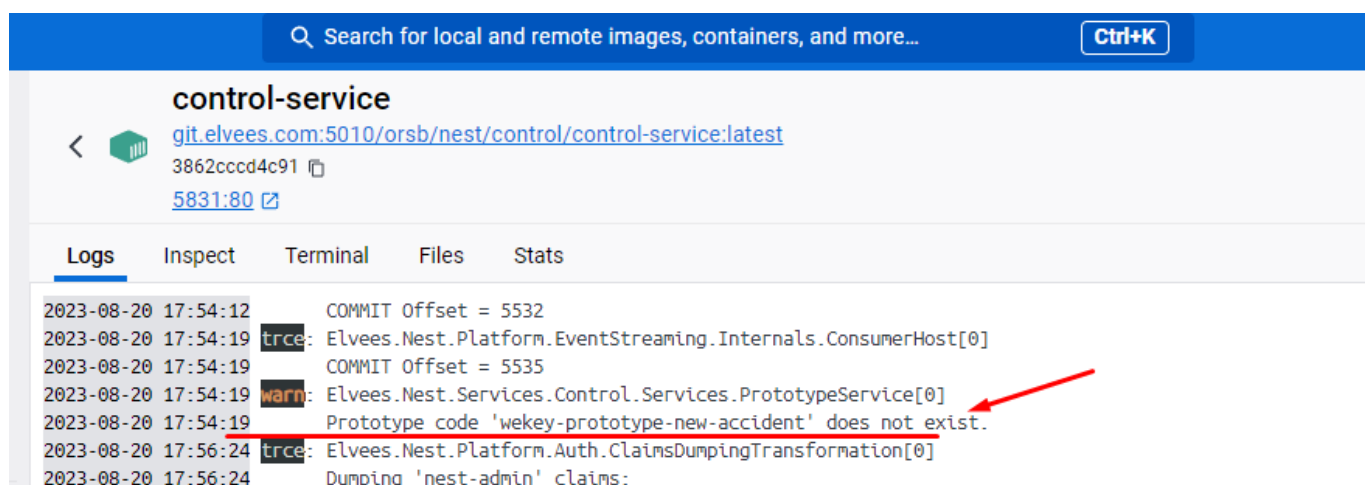


Рисунок 51

2) при наличии ошибок и предупреждений, устранить причины их возникновения.

В примере, приведенном выше, причиной того, что инцидент не создан, является отсутствие в БД прототипа с таким названием кода схемы.

5.9.2.2 Сообщение в Kafka о создании инцидента

Если в Kafka есть сообщение о создании инцидента (сообщение с кодом "accident accident.new.xxx"), а сам инцидент не отображается на странице в control, то необходимо проверить, что у сообщения есть tag. При этом tag попадает в зону ответственности, которая доступна оператору.

5.10 Запись автоматов, прототипов, зоны ответственности, файлов в БД

5.10.1 Запись автоматов, прототипов, зон ответственности с помощью утилиты VSCode

Наиболее удобным способом написания, применения и загрузки автоматов, прототипов и зон ответственности является использование утилиты VSCode. (<https://code.visualstudio.com/>):

- 1) в папку \docker-infra\имя проекта\business\automations перемещаем файлы json с автоматами;
- 2) в папку \docker-infra\stage-4-test\control\prototypes перемещаем файлы json с прототипами;
- 3) в папку \docker-infra\stage-4-test\scopes перемещаем файлы json с зонами ответственности;
- 4) в папку \docker-infra\stage-4-test\control\priorities перемещаем файлы json с приоритетами;
- 5) в папку \docker-infra\stage-4-test\filestore перемещаем аудиофайлы которые необходимо записать в БД. Поддерживаются форматы аудиофайлов mp3 и wav.

Для того чтобы применить данные автоматы, прототипы, зоны ответственности и прочее необходимо:

- 1) открыть командную строку;
- 2) перейти в «stage-4-test»;
- 3) запустить скрипт «setup.sh» с параметром «имя_сервера_NEST-M». Или открыть терминал bash и выполнить команду:

```
bash setup.sh имя_сервера_NEST-M
```

5.10.2 Запись автоматов, прототипов, зон ответственности с помощью утилиты Postman

Также есть возможность добавления автоматов, прототипов и зон ответственности через отправку POST-запросов на адрес соответствующего сервиса. Формат отправляемого тела во всех запросах в ЦП NEST-M – json формат. В теле запроса указывается схема автомата, приоритета и т.д. Успехом записи запроса есть получение ответа со статусом 200. Для отправки POST-запросов можно использовать утилиту Postman.

Автоматы отправлять на адрес модуля импорт-сервиса:

<http://адрес-сервера-NEST-M:5821/services/business/automations/{schemacode}>,

где {schemacode} – название автомата. Желательно, чтобы название соответствовало коду схемы отправляемого автомата.

Прототипы отправлять на адрес:

<http://адрес-сервера-NEST-M:5831/services/control/prototypes/{code}>,

где {code} - название кода прототипа.

Для записи зон ответственности необходимо отправить POST запрос на следующие адреса:

<http://адрес-сервера-NEST-M:5871/services/timeline/scopes/{code}> -модуль обработки событий,

<http://адрес-сервера-NEST-M:5861/services/monitor/scopes/{code}> – модуль обработки картографической информации,

<http://адрес-сервера-NEST-M:5861/services/import/scopes/{code}> – модуль универсального адаптера,

<http://адрес-сервера-NEST-M:5831/services/control/scopes/{code}> – модуль обработки инцидентов,

где {code} - название кода зоны ответственности.

Для записи приоритетов отправить POST-запрос на адрес:

<http://адрес-сервера-NEST-M:5831/services/control/scopes/{code}>,

где {code} - название кода прототипа.

Для записи аудиофайлов в БД отправить POST-запрос, прикрепив необходимый аудиофайл на адрес:

<http://адрес-сервера-NEST-M:5841/services/filestore>

В полученном ответе в поле "**fileName**" будет указано, под каким именем файл сохранился в БД и какое имя указывать в прототипе в поле "**soundNotificationSource**". Как пример:

```
"fileName": "Boney_M_-_Daddy_Cool_47966571.mp3"
```

Внимание: имя файла должно быть записано на латинском языке и без пробелов.

6 ПРОВЕРКА ПРОГРАММЫ

Описание проверок программы приведено в указаниях разделов 3 и 5 данного документа.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

АРМ	– автоматизированное рабочее место
БД	– база данных
ОС	– операционная система
ПК	– персональный компьютер
ПО	– программное обеспечение
СВН	– система видео наблюдения

