



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт искусственного интеллекта (ИИИ)
Кафедра промышленной информатики (ПИ)

КУРСОВАЯ РАБОТА

по дисциплине «Разработка автоматизированной системы реального времени»

Тема курсовой работы: «Разработка автоматизированной системы реального времени
мониторинга параметров производства кабельной продукции»

Студент Беликов М.Д.

Группа КВБО-03-21

Руководитель Холопов В.А.

Консультант Зорина Н.В.

Консультант Володина А.М.

Курсовая работа представлена к защите «_____» _____ 20 г.

Допущен (а) к защите «_____» _____ 20 г.

Москва, 2025



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт искусственного интеллекта (ИИИ)

Кафедра промышленной информатики (ПИ)

Утверждаю
Заведующий кафедрой ПИ
Холопов В.А.
(подпись) (Ф.И.О.)
«___» _____ 20__ г.

ЗАДАНИЕ

на выполнение курсовой работы

по дисциплине «Разработка автоматизированной системы реального времени»

Тема курсовой работы: разработка автоматизированной системы реального времени
технологического процесса стерилизации свиных консервов

Студент Беликов Михаил Дмитриевич Группа КВБО-03-21

Исходные данные: описание технологического процесса, параметры процесса, модели структурно-функционального анализа. Перечень вопросов, подлежащих разработке, и обязательного графического материала: анализ предметной области; сбор и анализ требований к АС; функциональные требования на разработку АС; проектирование БД; выбор средств ведения разработки; разработка БД; разработка кода; обработка данных в режиме реального времени; разработка интерфейсов

Срок представления к защите курсовой работы

до «05» апреля 2025г.

Задание на курсовую работу выдал

(подпись руководителя)

Холопов В.А.

(Ф.И.О. руководителя)

«15» февраля 2025г.

Задание на курсовую работу выдал

(подпись консультанта)

Зорина Н.В.

(Ф.И.О. консультанта)

«15» февраля 2025г.

Задание на курсовую работу выдал

(подпись консультанта)

Володина А.М.

(Ф.И.О. консультанта)

«15» февраля 2025г.

Задание на курсовую работу получил

(подпись обучающегося)

Беликов М. Д.
(Ф.И.О. обучающегося)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Аналитический раздел	6
1.1 Краткая характеристика объекта автоматизации.....	6
1.2 Сбор и анализ функциональных требований	10
2 Проектирование системы	15
2.1 Структурно-функциональное моделирование	15
2.2 Моделирование баз данных	17
2.3 Архитектурное моделирование	17
3 Разработка АС	20
3.1 Выбор средств ведения разработки.....	20
3.2 Разработка БД	21
3.3 Разработка системы мониторинга параметров	23
3.4 Разработка интерфейса	25
4 Тестирование системы мониторинга	28
4.1 Выбор средств ведения разработки.....	28
4.2 Модульное тестирование.....	30
Результаты тестирования	32
ЗАКЛЮЧЕНИЕ	33
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	34
ПРИЛОЖЕНИЯ.....	35
Приложение А	36
Приложение Б	37

ВВЕДЕНИЕ

Современные предприятия используют множество систем мониторинга и обработки данных, передаваемых как в пределах цеха, так и между подразделениями самого предприятия и их партнёров. Автоматический сбор информации позволяет удешевить производственный процесс, снизить влияние человеческого фактора на результаты производства и увеличить скорость самого производства. Автоматизированная система мониторинга параметров производства позволяет в режиме реального времени наблюдать за состоянием нужных узлов, вывести отклонения параметров от нормы и сформировать отчёт с подробным описанием изменений. В данной работе рассматривается проектирование автоматизированной системы мониторинга параметров производства кабельной продукции в реальном времени для участка производства СИП кабелей, на котором происходит нанесение изоляции.

1 АНАЛИТИЧЕСКИЙ РАЗДЕЛ

1.1 Краткая характеристика объекта автоматизации

Необходимо автоматизировать передачу всех необходимых данных к пользователю, чтобы пользователь мог контролировать процесс.

Процесс производства состоит из следующих этапов:

1. Приём и отбраковка сырья (алюминиевая проволока, полиэтиленовые гранулы (РЕ))
2. Скрутка проволоки в жилы
3. Нанесение изоляции
4. Скрутка жил в кабель
5. Фасовка и складирование

Используя структурно-функциональный подход и нотацию IDEF0, проведём анализ производства кабельной продукции (Рисунок 1.1).

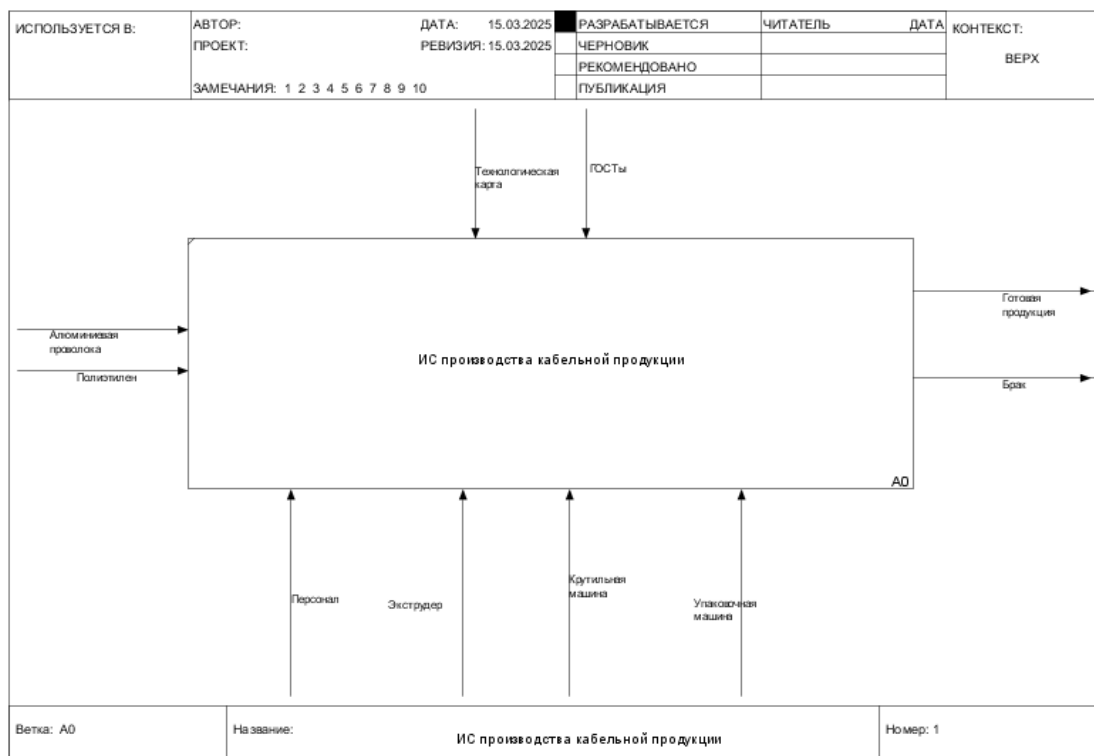


Рисунок 1.1 — IDEF0

В результате анализа выявлены необходимые для производства входные потоки:

— алюминий — материал жил кабелей СИП-4. Поступает в форме проволоки.

— полиэтилен — самый распространённый материал для изоляции электрических кабелей. Поступает в виде гранул.

В качестве механизмов, позволяющих производству выполнять проектируемые функции и задачи, были выделены следующие:

— персонал — операторы и специалисты, управляющие процессом производства, наладкой и обслуживанием оборудования;

— экструдер;

— крутильная машина для жил;

— крутильная машина для кабелей;

— упаковочная машина.

Влияние на ход выполнения процессов оказывают управляющие потоки, которые будут учтены при проектировании производства:

— технологическая карта — описывает процесс выполнения работы или производства продукта с подробными инструкциями и нормативами. Она содержит последовательность этапов, методы и инструменты, которые необходимо использовать, а также время и ресурсы, требуемые для каждого этапа;

— ГОСТы — государственные стандарты, определяющие требования к качеству и безопасности продукции.

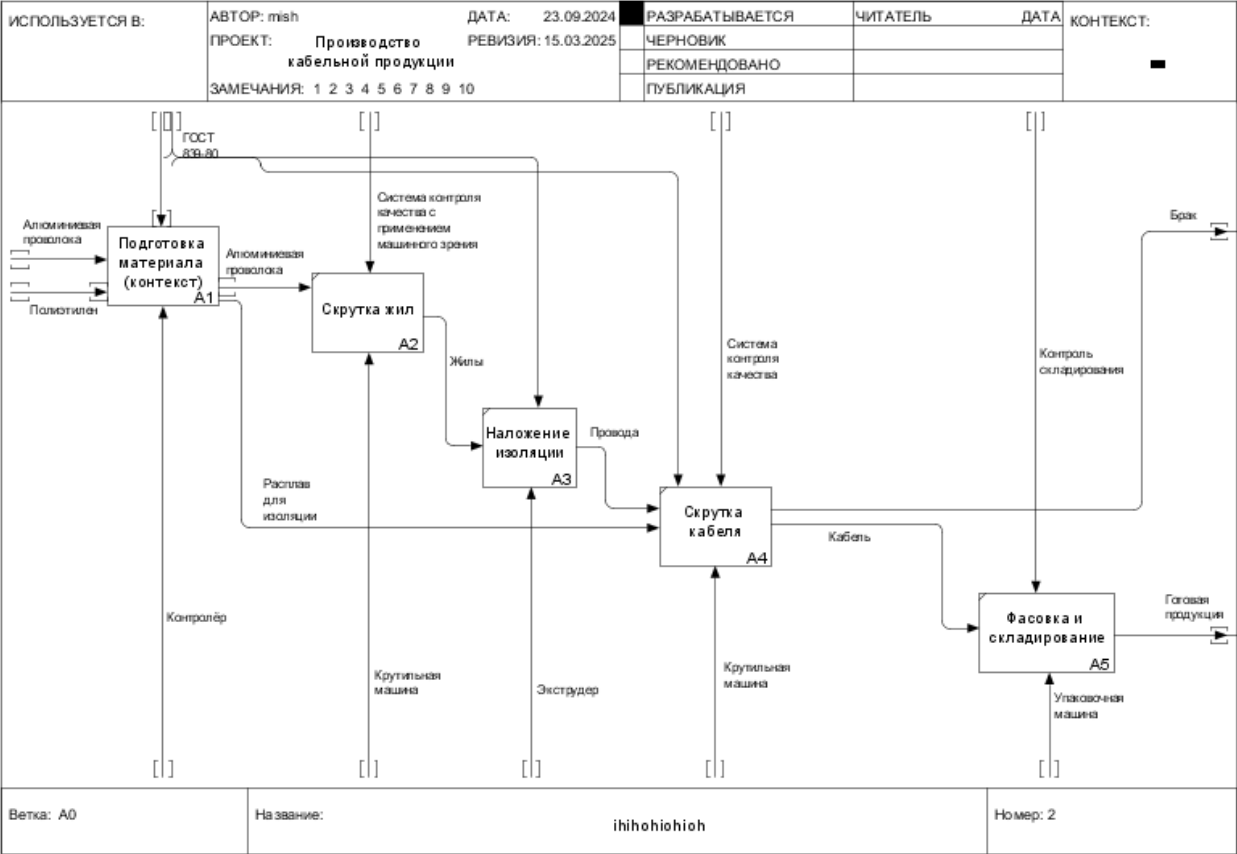


Рисунок 1.2 — IDEF0. Декомпозиция первого уровня

На Рисунке 1.2 представлена декомпозиция, отображающая основные процессы, из которых будет состоять проектируемое производство.

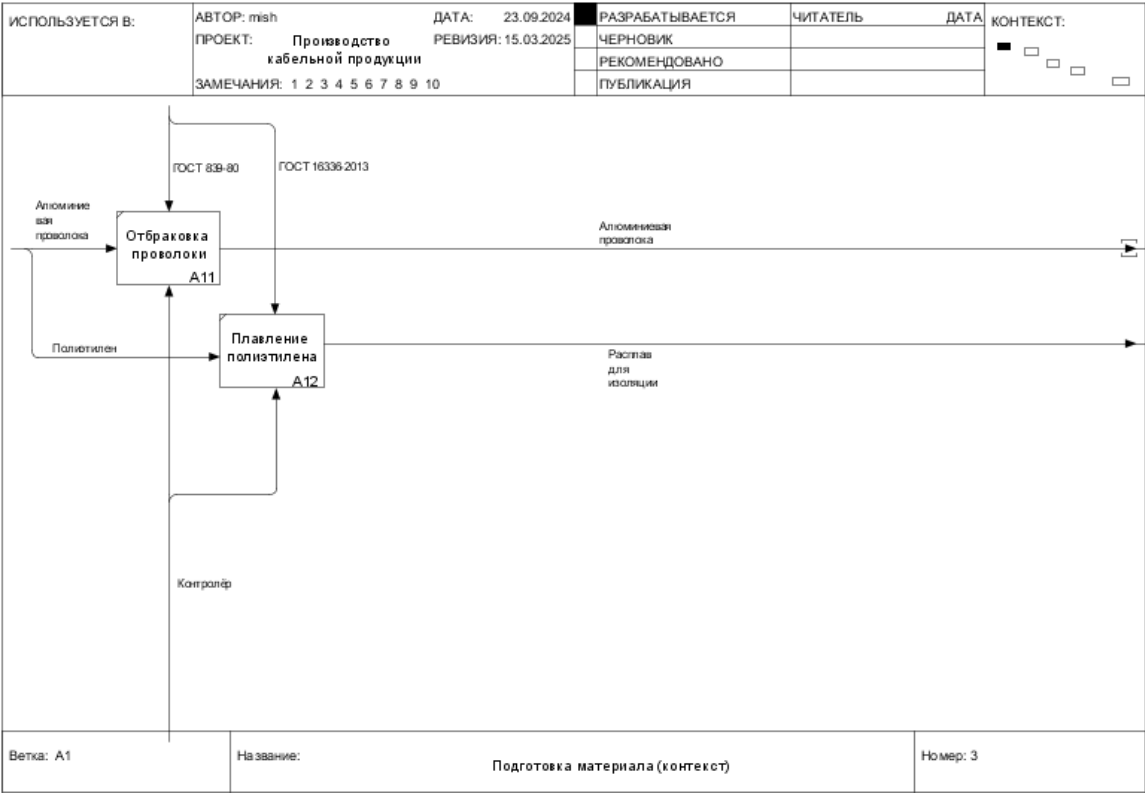


Рисунок 1.3 — Декомпозиция третьего уровня

Процесс «Подготовка материала» представляет собой набор следующих функций и задач (Рисунок 1.3):

— отбраковка алюминия — партия алюминиевой проволоки проверяется на соответствие ГОСТу. Прошедшая проволока направляется на крутильную машину;

— плавление полиэтилена — шарики полиэтилена переплавляются для последующего нанесения на скрученные жилы в качестве изоляции.

Процесс «Скрутка»: алюминиевая проволока скручивается в жилу на крутильной машине из нескольких прутков, что обеспечивает гибкость кабеля, устойчивость к физическим нагрузкам и распределяет нагрузку при работе на несколько жил, уменьшая нагрев кабеля.

Процесс «Наложение изоляции: алюминиевая жила проходит через экструдер, который наносит горячую изоляцию на поверхность жилы, обволакивая скрутку со всех сторон, что обеспечивает защиту провода от внешней среды, физических воздействий, позволяет производить работы с кабелем.

Процесс «Скрутка»: полученные провода с изоляцией скручиваются в готовый кабель.

Процесс «Фасовка и складирование: готовый кабель наматывается на катушки и отправляется на склад. В таком виде осуществляется поставка кабеля к конечному потребителю.

Считывается: температура экструдера, толщина получаемой изоляции, диаметр кабеля, скорость протяжки жилы.

Рассматривается процесс нанесения изоляции.

Таблица 1.1 — Контролируемые величины

№	Датчик	Величина	Эталонное значение	Единицы измерения
1	Термометр	Температура экструдера	160-180	°C
2	Энкодер	Скорость протяжки	30-40	м/мин
3	Ультразвуковой толщиномер	Толщина изоляции	1,8-2	мм
4	Оптический датчик	Сечение жилы	50	мм ²

1.2 Сбор и анализ функциональных требований

Разрабатываемая АС должна обеспечивать следующий функционал:

- передача команд о пуске и остановке производства;
- внесение информации о поступающем сырье и эталонных параметрах;
- получение отчётов о ходе выполнения технологического процесса и состояния оборудования;
- разграничение прав доступа и защита от несанкционированного входа.

На основе перечисленных требований была построена диаграмма прецедентов, которая поможет описать сценарии взаимодействия пользователей с системой (Рисунок 1.4).

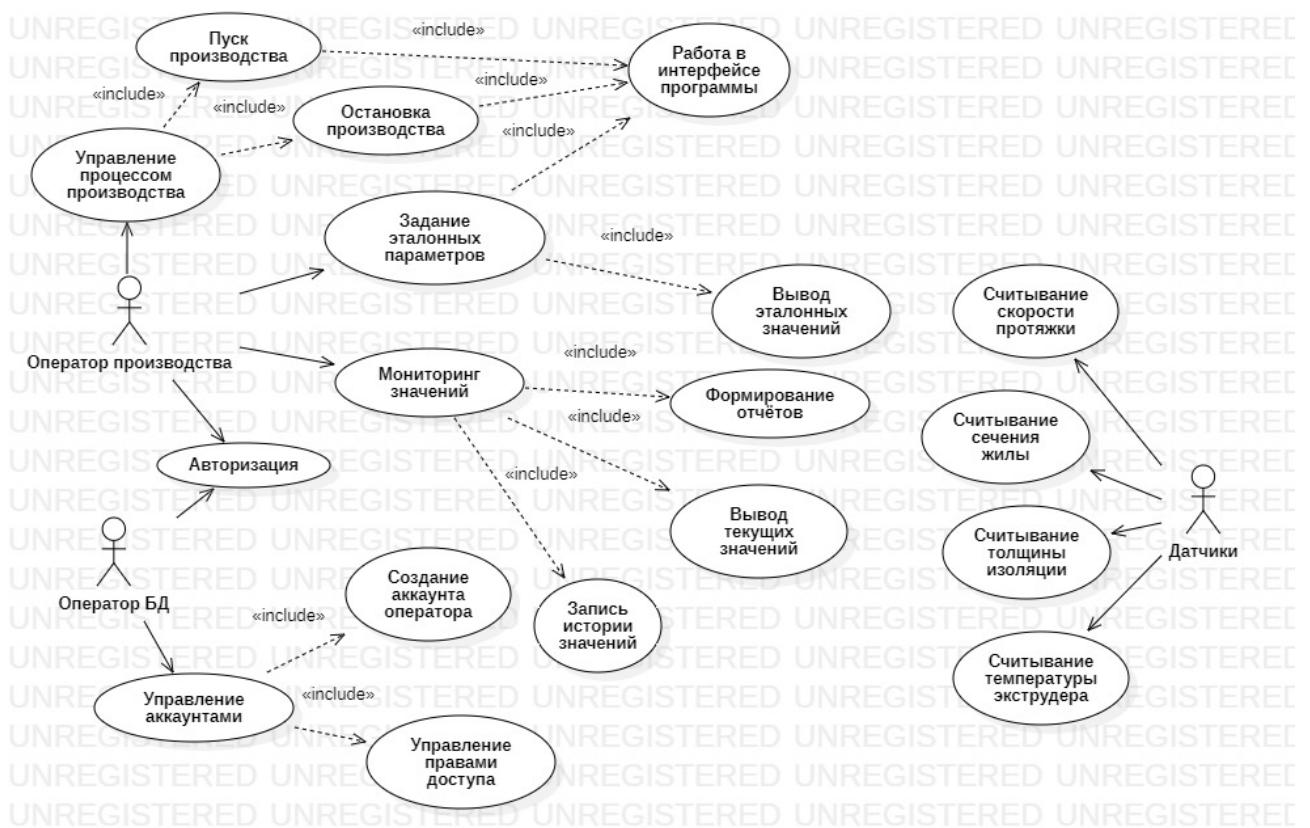


Рисунок 1.4 — Диаграмма прецедентов

История 1. Управление процессом производства

Оператору необходима возможность запускать и останавливать процесс производства через интерфейс системы для контроля для контроля процесса и возможности своевременно реагировать на изменения.

Прецедент 1. Управление процессом производства

Актёры: Оператор производства.

Предусловия: Оператор авторизован и имеет доступ к системе.

Постусловия: Процесс производства запущен, либо остановлен

Сценарий:

1. Оператор открывает интерфейс системы.
2. Проходит авторизацию в системе.
3. Выбирает опцию «Запустить процесс производства».
4. Система запускает процесс производства и оповещает оператора об успешном изменении состояния.
5. При выборе опции «Остановить производство» система останавливает производственный процесс и оповещает оператора об остановке производства.

Альтернативные сценарии:

Если процесс производства не может быть запущен, система отображает информацию об ошибке.

История 2. Задание данных о поступающем сырье

Оператору необходимо передавать данные о сырье, поступающем на производство, чтобы указать конкретные данные для каждой партии сырья и готовых изделий.

Прецедент 2. Задание данных о поступающем сырье

Актёры: Оператор производства.

Предусловия: Оператор авторизован и имеет доступ к системе.

Постусловия: Эталонные данные и данные о сырье внесены в систему.

Сценарий:

1. Оператор открывает интерфейс системы.
2. Выбирает опцию «Задать сырьё и эталонные данные».
3. Вводит необходимые данные.
4. Система вносит внесённые изменения и уведомляет об успешно сохранённых данных.

Альтернативные сценарии:

Введённые данные некорректны — система отображает сообщение об ошибке и запрашивает повторный ввод данных.

История 3. Мониторинг значений

Оператору необходимо иметь возможность наблюдать за текущими значениями датчиков на производственной линии, наблюдать за трендом изменений и формировать отчёт на основе показаний.

Прецедент 3. Мониторинг значений

Актёры: Оператор производства.

Предусловия: Оператор авторизован и имеет доступ к системе.

Постусловия: Текущие данные, отклонение от эталонных значений отображаются в интерфейсе системы.

Сценарий:

1. Оператор открывает интерфейс системы.
2. Выбирает опцию «Мониторинг».
3. Система отображает текущее значение, эталонное значение, дельту.

Альтернативные сценарии:

1. Производство не запущено — выводится соответствующее сообщение.

История 4. Авторизация

Всем пользователям системы необходимо иметь возможность авторизоваться, чтобы получить доступ к соответствующим функциям. Оператор производства не должен иметь доступа к системе аккаунтов, в то время как оператору базы данных системы не нужно взаимодействовать с самим производственным процессом напрямую.

Прецедент 4. Авторизация

Актёры: Оператор производства, оператор БД.

Предусловия: Оператор имеет доступ к системе.

Постусловия: Оператор выполнил авторизацию и имеет полный доступ к соответствующим функциям.

Сценарий:

1. Оператор открывает интерфейс системы.
2. Оператор вводит логин и пароль.
3. Система выводит сообщение об успешном входе, отображается категория авторизованного пользователя.

Альтернативные сценарии:

1. Некорректные данные для авторизации — система выводит сообщение об ошибке.
2. В системе отсутствуют аккаунты — выполняется вход в системный аккаунт с доступом к системе управления аккаунтами.

История 5. Управление аккаунтами

Оператор БД имеет возможность создавать, удалять и редактировать аккаунты пользователей системы, настраивать им доступ к функциям системы.

Прецедент 5. Управление аккаунтами

Актёры: Оператор БД.

Предусловия: Оператор авторизован и имеет доступ к системе.

Постусловия: Внесены изменения в набор аккаунтов.

Сценарий:

1. Оператор открывает интерфейс системы.
2. Выбирает опцию «Управление аккаунтами».
3. Вносятся необходимые изменения.
4. Система сохраняет внесённые изменения.

Альтернативные сценарии:

1. Попытка удалить аккаунт, который вносит изменения — система выводит сообщение об ошибке.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Структурно-функциональное моделирование

Для более детального анализа передаваемой и получаемой информации, использована нотация DFD структурно-функционального подхода (Рисунок 2.1).

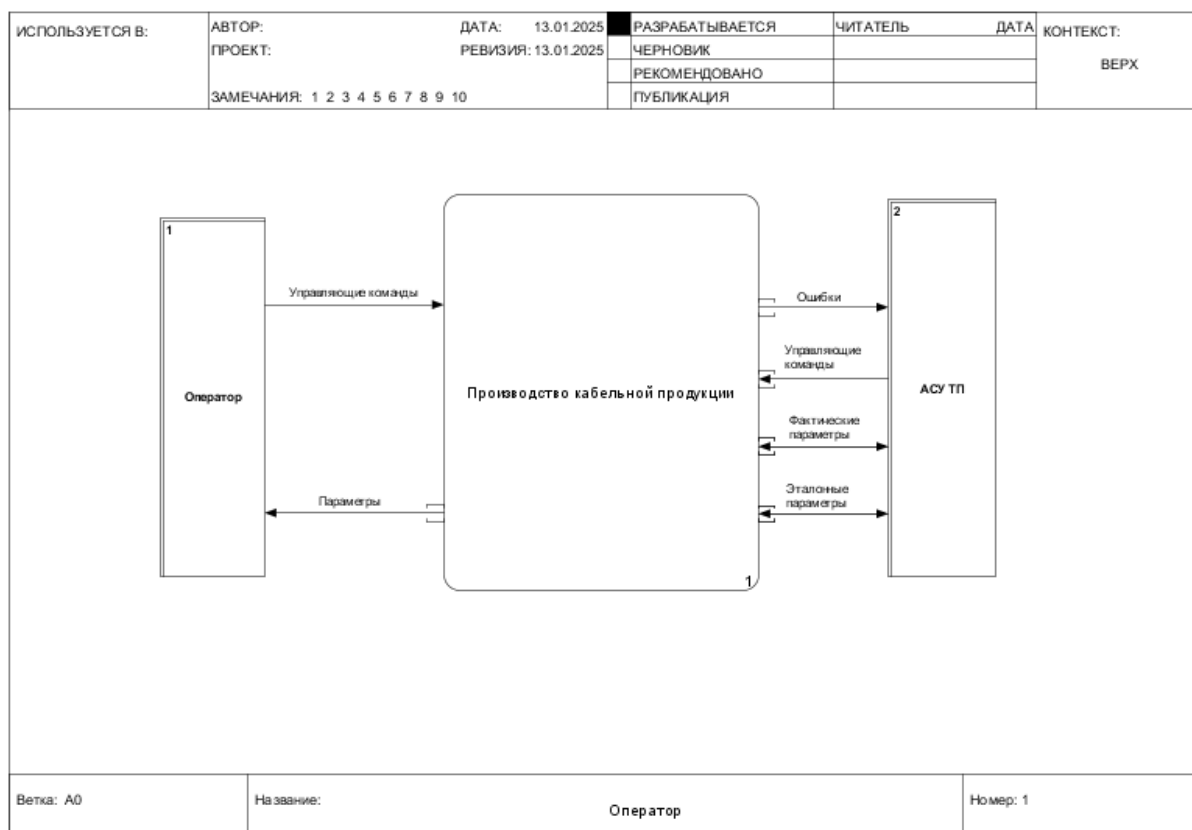


Рисунок 2.1 — DFD диаграмма первого уровня

«Оператор» получает информацию от системы для мониторинга технологических процессов, что позволяет ему следить за состоянием оборудования, контролировать параметры и оперативно реагировать на изменения или неисправности. В свою очередь, сам «Оператор» имеет возможность остановить или запустить производство посредством команд, либо осуществлять действия с базой данных системы, такие как управление аккаунтами и формирование отчётов на основе данных из БД.

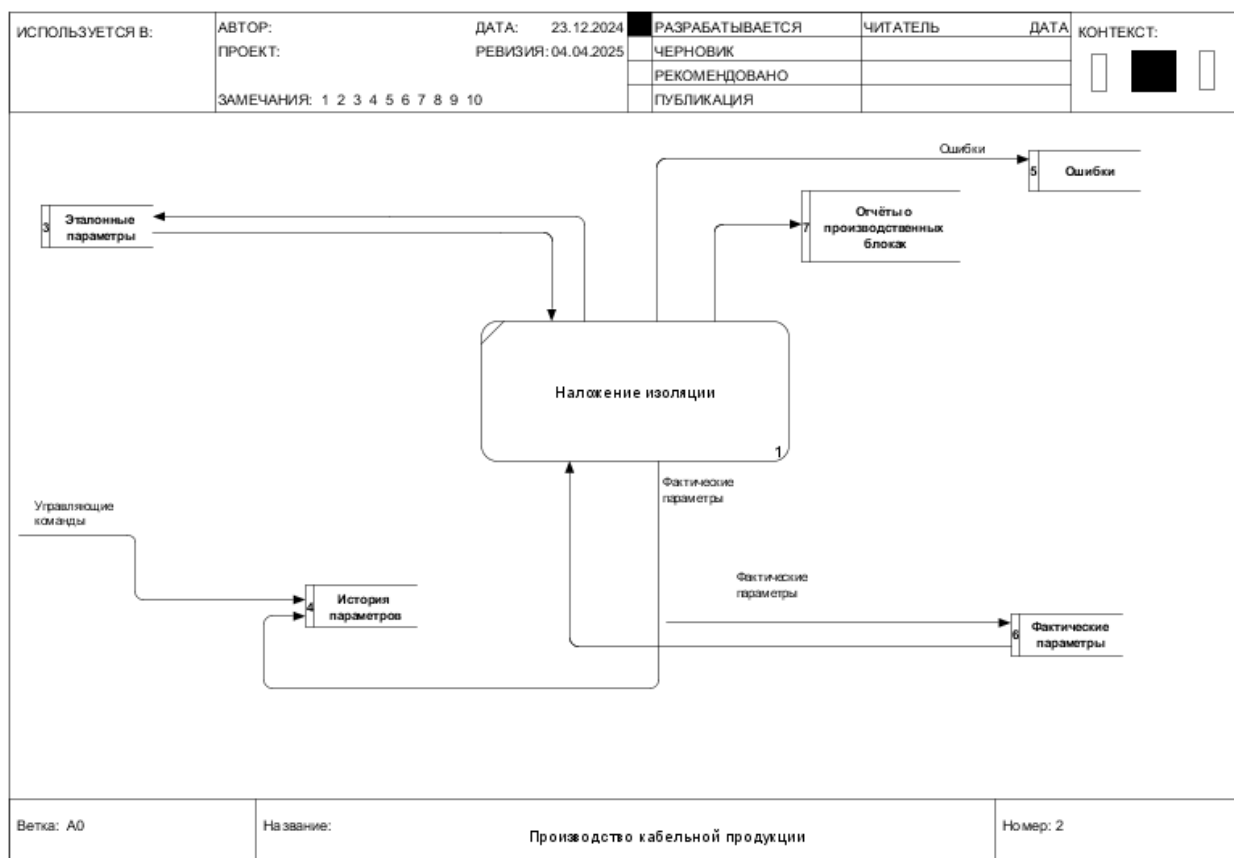


Рисунок 2.2 — DFD диаграмма второго уровня (декомпозиция первого уровня)

В диаграмме второго уровня рассматривается только процесс нанесения изоляции, мониторинг которого будет осуществляться.

Присутствуют следующие хранилища:

- эталонные параметры — хранение заданных вручную идеальных параметров, возвращаемых датчиками в ходе процесса, и допустимых отклонений от эталона;
- фактические параметры — входные данные с датчиков;
- история параметров — входные данные с датчиков, используемые для вычисления трендов изменения параметров;
- ошибки — записи о выходе фактических параметров за установленные пределы;
- отчёты о производственных блоках — информация о сформированных отчётах.

2.2 Моделирование баз данных

Была спроектирована логическая схема БД, которая позволит обеспечить эффективное хранение, обработку и анализ информации, поступающей с каждого этапа технологического процесса (Рисунок 2.3).

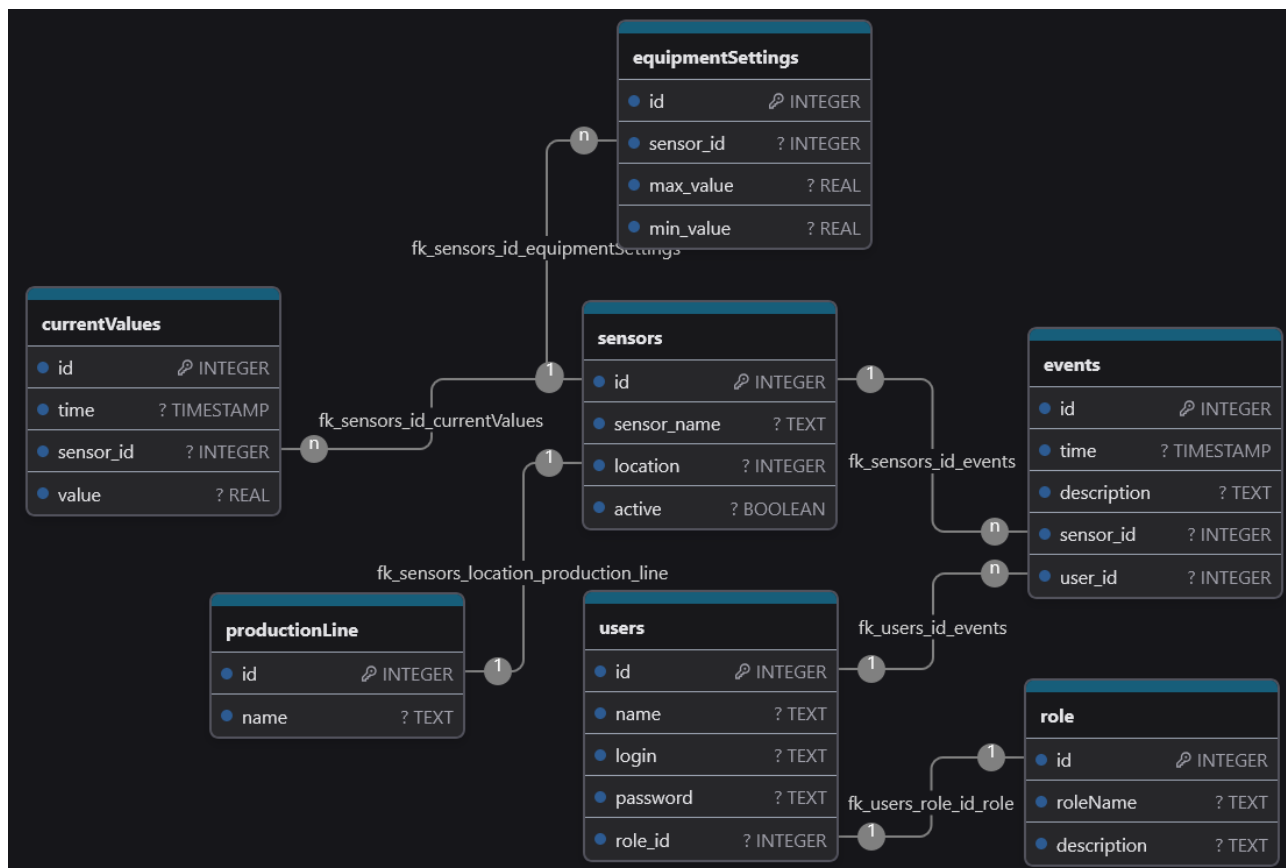


Рисунок 2.3 — Логическая схема БД

2.3 Архитектурное моделирование

Archimate представляет информацию на разных уровнях её воплощения в физическом мире. Базовые уровни:

1. Бизнес-слой — деятельность людей, сотрудников, представленных в виде их должностных позиций.
2. Слой приложений — работа программного обеспечения и взаимодействие ПО.
3. Технологический слой — работа физических устройств, компонентов.

На технологическом слое представлены исполняющие устройства, которые подключены к ПЛК. В свою очередь, ПЛК находится в одной локальной

сети с рабочей станцией и сервером приложений (который включает в себя ОРС сервер), а также имеет выход в Интернет через брандмауэр и маршрутизатор для удалённого контроля работы. С сервером приложений взаимодействует сервер БД, который хранит всю обрабатываемую информацию и обеспечивает к ней доступ.

На слою приложений представлены модули АСУ ТП, необходимые для обеспечения функционирования производства.

На бизнес-слою представлены действия сотрудников производства и их связь с компонентами слоя приложений.

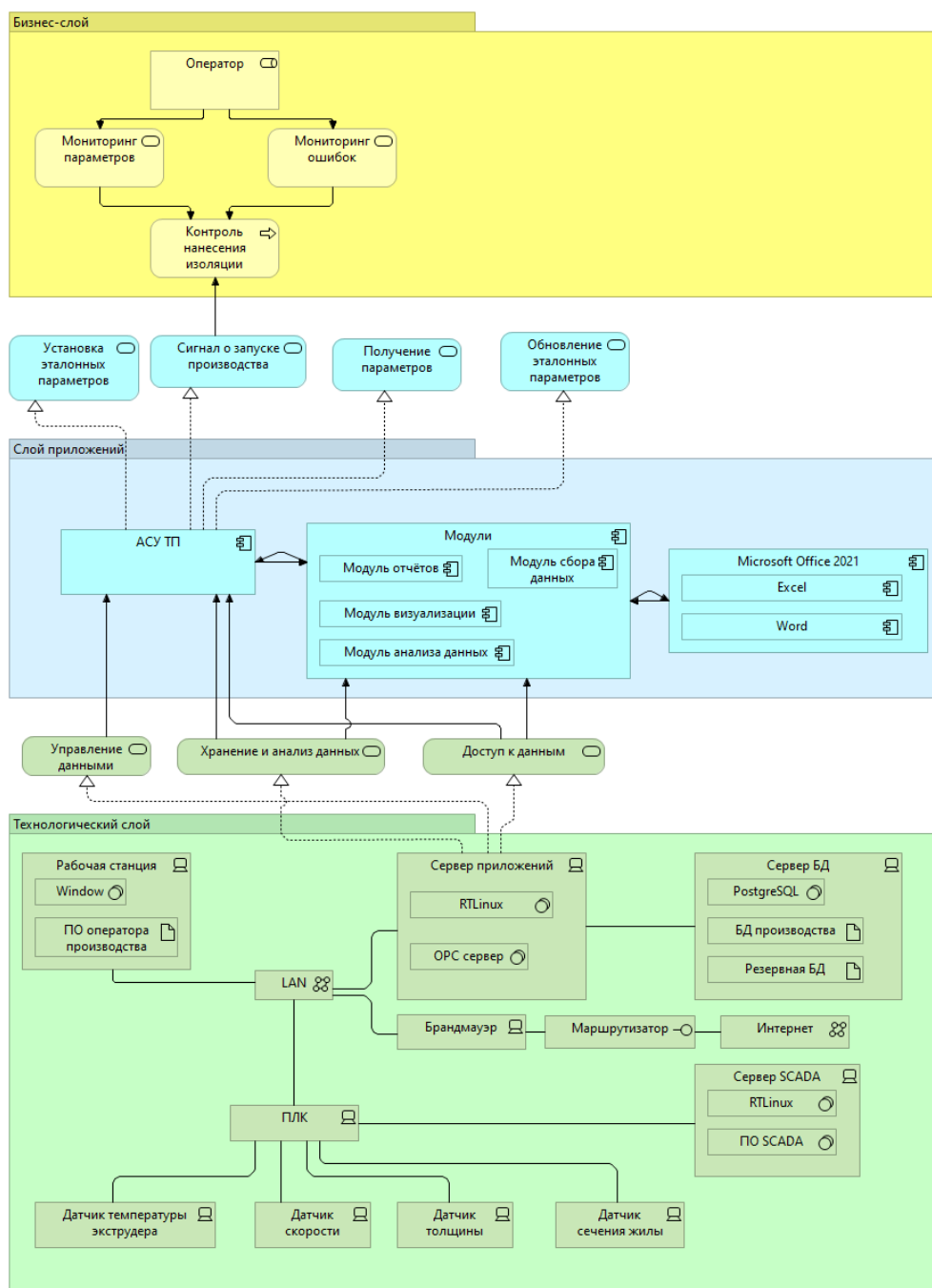


Рисунок 2.4 — Информационная структура

3 РАЗРАБОТКА АС

3.1 Выбор средств ведения разработки

Основным языком разработки для системы мониторинга является Python 3, он обладает обильной базой различных библиотек для работы с данными и веб-разработки.

Для хранения данных используется PostgreSQL — популярная и надёжная реляционная СУБД с открытым исходным кодом. PostgreSQL обеспечивает транзакционную целостность данных, поддержку JSON для хранения сложных структур данных датчиков и высокую производительность при обработке больших объёмов информации. Взаимодействие с базой данных осуществляется через асинхронный драйвер `asyncpg` и ORM `SQLAlchemy`.

В качестве фреймворка для создания веб-интерфейса и API выбран FastAPI — современный, высокопроизводительный веб-фреймворк с поддержкой асинхронных операций. FastAPI обеспечивает автоматическую генерацию документации API, валидацию данных и типизацию, что значительно снижает количество ошибок на этапе разработки. Выбор FastAPI позволил реализовать как API для взаимодействия с системой, так и веб-интерфейс для мониторинга производства и управления параметрами.

Архитектура системы основана на принципах событийно-ориентированного программирования и использует брокеры сообщений для обеспечения надёжного обмена данными между компонентами. Для получения данных с датчиков применяется протокол MQTT — легковесный протокол обмена сообщениями, который идеально подходит для устройств Интернета вещей и промышленных датчиков. В качестве MQTT брокера используется Mosquitto. Обработка потоков данных осуществляется с помощью Apache Kafka, обеспечивающей надёжную доставку сообщений и масштабируемость при растущих объёмах данных.

Интерфейс пользователя построен с использованием современных веб-технологий HTML5, CSS3 и JavaScript. Для улучшения внешнего вида и создания

адаптивного интерфейса применяется Bootstrap, обеспечивающий единообразие элементов управления и правильное отображение на различных устройствах.

В системе реализована аутентификация на основе JWT (JSON Web Tokens). Это позволяет обеспечить безопасный доступ к функциям системы в соответствии с ролями пользователей (администратор, оператор, технический специалист, менеджер).

Для связи между Backend и Fronted частями используется протокол WebSocket, который устанавливает двунаправленное соединение и позволяет не опрашивать по HTTP сервер каждую секунду.

Программная архитектура системы построена по модульному принципу и состоит из следующих основных компонентов:

1. Модуль сбора данных датчиков — отвечает за получение информации с физических датчиков или симулятора через протокол MQTT.
2. Модуль обработки и анализа данных — выполняет первичную обработку данных, проверяет соответствие показаний допустимым значениям и генерирует оповещения при отклонениях от нормы.
3. Модуль хранения данных — обеспечивает сохранение информации в базе данных PostgreSQL и предоставляет интерфейс для выполнения запросов.
4. Модуль оповещений — формирует уведомления о критических ситуациях и отклонениях в работе оборудования.
5. Модуль аутентификации и авторизации — управляет процессами идентификации пользователей и контролирует доступ к функциям системы.
6. Веб-модуль — предоставляет интерфейс пользователя для мониторинга и управления производственным процессом.
7. API модуль — обеспечивает программный интерфейс для интеграции с другими системами.

3.2 Разработка БД

Центральным элементом модели является таблица датчиков (sensors), которая содержит информацию о всех установленных на производстве

измерительных устройствах. Датчики расположены в определенных местах производственной линии, что отражено в связи с таблицей местоположений (productionLine).

Таблица показаний датчиков (currentValues) представляет собой основное хранилище данных, поступающих от датчиков в реальном времени. При отклонении показаний от нормы система генерирует события, которые сохраняются в таблице событий (events).

Доступ к системе контролируется через таблицы пользователей (users) и ролей (roles), обеспечивая разграничение прав в соответствии с должностными обязанностями пользователей.

На Листинге 3.1 показан SQL-скрипт для создания таблиц users, role, sensors, currentValues, productionLine и events.

Листинг 3.1 — Создание таблиц

```
CREATE TABLE "users" (  
    "id" BIGINT NOT NULL,  
    "name" TEXT NOT NULL,  
    "login" TEXT NOT NULL,  
    "password" TEXT NOT NULL,  
    "role_id" INTEGER NOT NULL  
);  
ALTER TABLE "users" ADD PRIMARY KEY("id");  
CREATE TABLE "role"(  
    "id" BIGINT NOT NULL,  
    "roleName" TEXT NOT NULL,  
    "description" BIGINT NOT NULL  
);  
ALTER TABLE "role" ADD PRIMARY KEY("id");  
CREATE TABLE "sensors"(  
    "id" BIGINT NOT NULL,  
    "sensor_name" TEXT NOT NULL,  
    "location" BIGINT NOT NULL,  
    "active" BOOLEAN NOT NULL  
);  
ALTER TABLE "sensors" ADD PRIMARY KEY("id");  
CREATE TABLE "currentValues"(  
    "id" BIGINT NOT NULL,  
    "time" TIMESTAMP(0) WITHOUT TIME ZONE NOT NULL,  
    "sensors_id" BIGINT NOT NULL,  
    "value" FLOAT(53) NOT NULL  
);  
ALTER TABLE "currentValues" ADD PRIMARY KEY("id");  
CREATE TABLE "productionLine"(  
    "id" BIGINT NOT NULL,  
    "name" TEXT NOT NULL  
);  
ALTER TABLE "productionLine" ADD PRIMARY KEY("id");  
CREATE TABLE "events"(  
    "id" BIGINT NOT NULL,  
    "time" TIMESTAMP(0) WITHOUT TIME ZONE NOT NULL,  
    "description" TEXT NOT NULL,  
    "sensors_id" BIGINT NOT NULL,  
    "user_id" BIGINT NOT NULL  
);  
ALTER TABLE "events" ADD PRIMARY KEY("id");
```

3.3 Разработка системы мониторинга параметров

Разработка системы мониторинга производства кабельной продукции состоит из нескольких ключевых элементов, реализующих основной функционал — от сбора данных с датчиков до их обработки и отображения в веб-интерфейсе.

Сбор данных с датчиков осуществляется через протокол MQTT (Приложение А), который обеспечивает надёжную передачу сообщений даже в

условиях нестабильного сетевого соединения. Клиент обеспечивает подключение к MQTT брокеру, подписку на необходимые топики, обработку входящих сообщений и отказоустойчивость при разрыве соединения.

API системы реализовано с использованием FastAPI и предоставляет доступ к данным о текущем состоянии производства, истории показаний датчиков и оповещениях. API поддерживает JWT-аутентификацию и разграничение доступа по ролям.

Обработка данных в режиме реального времени является ключевым аспектом разработанной системы. В этом разделе рассматривается архитектура и реализация компонентов, обеспечивающих сбор, обработку, анализ и визуализацию данных в реальном времени.

Система обработки данных в режиме реального времени построена на принципах событийно-ориентированной архитектуры (Event-Driven Architecture), где каждое изменение показаний датчика рассматривается как событие, требующее обработки.

Такая архитектура обеспечивает:

1. Низкую задержку между получением данных и их отображением;
2. Масштабируемость системы при увеличении количества датчиков;
3. Устойчивость к сбоям отдельных компонентов;
4. Возможность параллельной обработки больших объёмов данных.

MQTT (Message Queuing Telemetry Transport) был выбран как основной протокол для сбора данных с датчиков благодаря его легковесности, надёжности и поддержке модели публикации/подписки. Система использует иерархическую структуру топиков для организации потоков данных от различных участков производства.

Для обеспечения обновления данных в веб-интерфейсе без перезагрузки страницы используется технология WebSocket. Код класса, отвечающего за работу по web socket, приведен в Приложении Б.

Реализация WebSocket сервера поддерживает следующие функции:

- подключение клиентов к различным каналам данных;

- широковещательная рассылка обновлений;
- целевая отправка оповещений конкретным клиентам.

3.4 Разработка интерфейса

Разработка интерфейсов для системы была ориентирована на предоставление удобного, информативного и эффективного инструмента для работы операторов и руководителей производства. Основной целью являлось создание интуитивно понятного веб-интерфейса, обеспечивающего оперативный мониторинг и управление процессами производства в режиме реального времени.

Для разработки фронтенд-части системы были использованы следующие технологии:

1. HTML5/CSS3/JavaScript — основа веб-интерфейса.
2. Bootstrap 5 — фреймворк для создания адаптивного и современного дизайна.
3. Chart.js — библиотека для визуализации данных.
4. WebSocket API — для обеспечения обновления данных в реальном времени.
5. Jinja2 — шаблонизатор для генерации HTML на стороне сервера.
6. FontAwesome — набор иконок для улучшения визуального восприятия интерфейса.
7. Fetch API — для асинхронного взаимодействия с REST API сервера.

Структура пользовательского интерфейса

Интерфейс системы организован в виде панели управления с различными функциональными разделами (Рисунок 3.1).

Содержимое интерфейса:

1. Навигационное меню — обеспечивает быстрый доступ ко всем разделам системы

2. Информационная панель (дашборд) — отображает общее состояние производства.
3. Детальные страницы датчиков — для просмотра подробной информации по каждому датчику.
4. Раздел оповещений и событий — для отслеживания нештатных ситуаций.
5. Аналитические отчёты и графики — для анализа производственных показателей.
6. Панель настроек — для конфигурирования параметров системы.
7. Страница авторизации — для аутентификации пользователей с разным уровнем доступа.

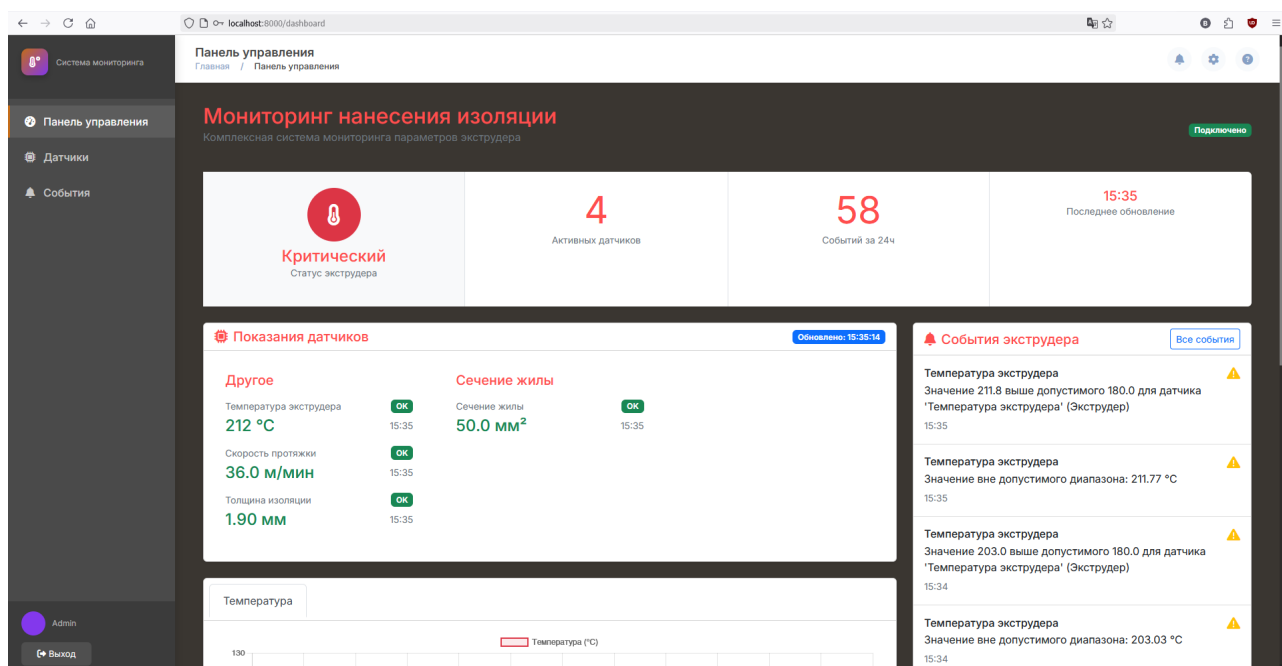


Рисунок 3.1 — Интерфейс системы мониторинга

При входе в систему появляется окно аутентификации (Рисунок 3.2).

The screenshot shows a login window titled 'Вход в систему' (System Login). It contains two input fields: 'Логин' (Login) with the text 'admin' and 'Пароль' (Password) with masked characters. Below the password field is a blue 'Войти' (Login) button.

Рисунок 3.2 — Аутентификация

На Рисунке 3.3 представлен раздел мониторинга значений. Здесь отображаются все значения с датчиков в реальном времени.

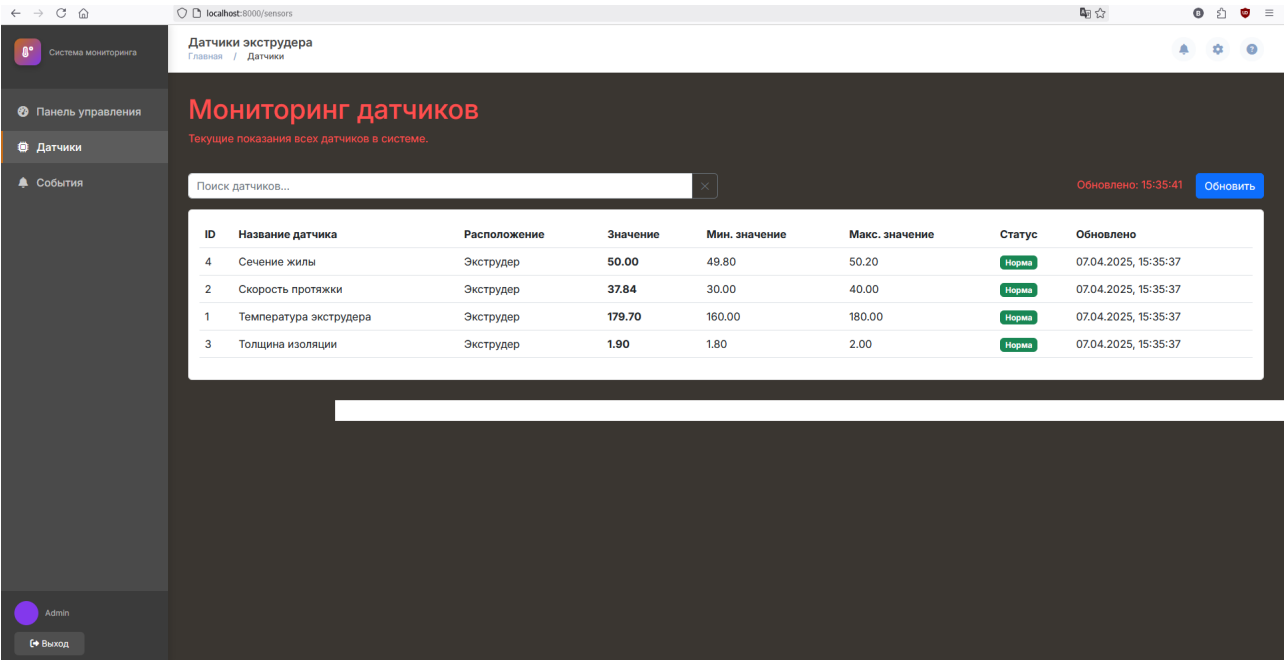


Рисунок 3.3 — Мониторинг значений датчиков

Раздел «Оповещения системы» отображает все записанные ошибки, которые были обработаны системой (Рисунок 3.4).

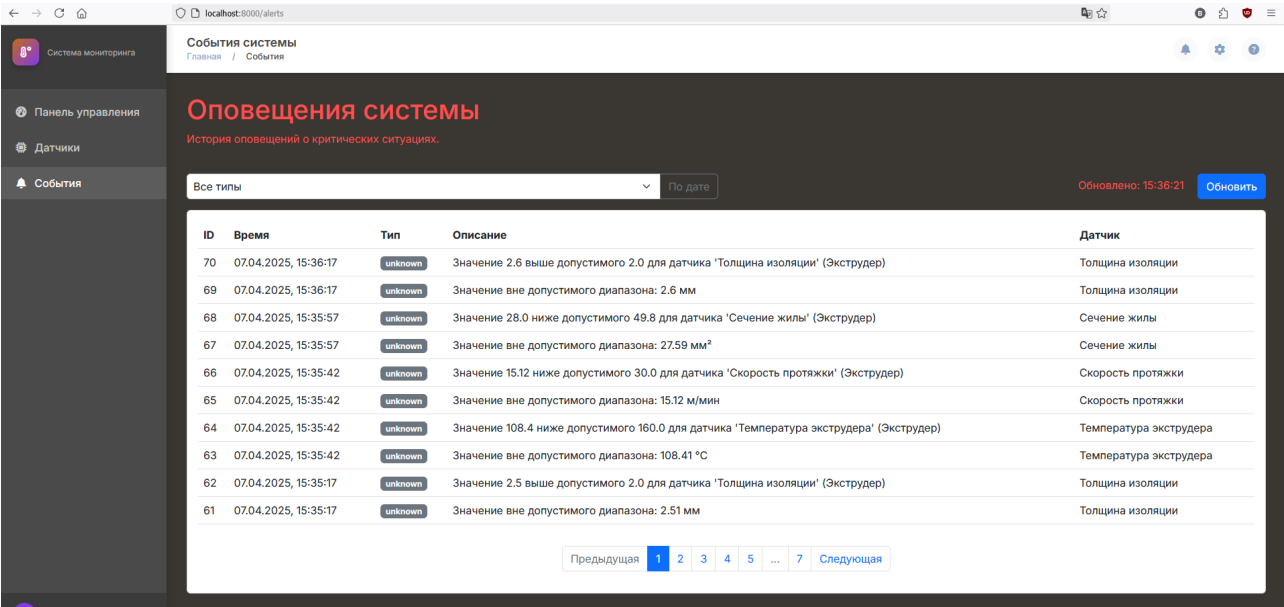


Рисунок 3.4 — Оповещения системы

4 ТЕСТИРОВАНИЕ СИСТЕМЫ МОНИТОРИНГА

Процесс тестирования автоматизированной системы основан на Use Case диаграмме, описанной в 1 главе, которая иллюстрирует основные варианты использования и взаимодействия пользователей с системой. Тестирование было направлено на проверку корректности работы всех функциональных компонентов, описанных в диаграмме.

4.1 Выбор средств ведения разработки

Тест авторизации оператора

Описание: проверка корректности авторизации оператора через различные интерфейсы.

Предусловия: Система запущена и функционирует.

Шаги:

1. Запустить интерфейс HMI.
2. Ввести корректные учетные данные оператора.
3. Нажать кнопку входа.
4. Повторить процедуру входа через веб-дашборд.

Ожидаемые результаты:

- успешная авторизация через оба интерфейса;
- предоставление доступа к функционалу мониторинга;
- отображение имени авторизованного пользователя в интерфейсе.

Тест сбора данных с датчиков

Описание: Проверка корректности сбора данных со всех датчиков линии.

Предусловия: Оператор авторизован в системе.

Шаги:

1. Перейти на экран мониторинга.

1. Проверить обновление данных от всех типов датчиков (температуры, давления, влажности).

2. Замерить частоту обновления показаний.

Ожидаемые результаты:

- отображение актуальных данных со всех датчиков;
- обновление показаний с заданной периодичностью;
- корректное отображение единиц измерения.

Тест отображения параметров в реальном времени

Описание: Проверка корректности отображения производственных параметров.

Предусловия: Оператор авторизован, датчики передают данные.

Шаги:

1. Наблюдать за отображением параметров в течение 10 минут.
2. Сверить отображаемые данные с показаниями контрольных приборов (если доступны).
3. Проверить различные режимы отображения (графики, числовые значения).

Ожидаемые результаты:

- корректное отображение всех параметров;
- соответствие отображаемых данных фактическим показаниям;
- правильная работа различных режимов отображения.

Тест генерации уведомлений при отклонениях

Описание: Проверка срабатывания системы уведомлений при выходе параметров за допустимые пределы.

Предусловия: Оператор авторизован, система мониторинга активна.

Шаги:

1. Эмулировать превышение температуры (через тестовый режим или физическое воздействие на датчик).

2. Наблюдать за реакцией системы.
3. Подтвердить получение уведомления.
4. Повторить для других типов параметров (давление, уровень воды в автоклаве).

Ожидаемые результаты:

- генерация визуального уведомления;
- отображение детальной информации о характере отклонения;
- возможность фиксации сигнала оператором;
- запись события в системный журнал.

Тест просмотра журналов событий и логов

Описание: Проверка доступности и полноты журналов событий и логов системы.

Предусловия: Сервисный инженер авторизован в системе.

Шаги:

1. Перейти к разделу журналов событий.
2. Проверить наличие фильтров и поиска.
3. Просмотреть записи о различных типах событий (штатная работа, ошибки, предупреждения).
4. Проверить доступность детальной информации о каждом событии.

Ожидаемые результаты:

- отображение полного списка событий;
- наличие всех необходимых категорий событий;
- корректная работа фильтров и поиска;
- доступность подробной информации о каждом событии.

4.2 Модульное тестирование

Также в рамках тестирования автоматизированной системы производства бутылок было проведено модульное (unit) тестирование. Для тестирования были

скачены дополнительные библиотеки — `pytest-asyncio`, `pytest-mock` — для создания заглушек.

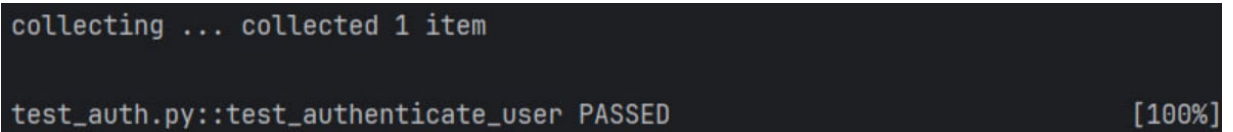
На Листинге 4.1 изображен unit тест для проверки работы аутентификации системы.

На Рисунке 4.1 изображено успешное завершение теста.

Листинг 4.1 — Проверка работы аутентификации

```
@pytest.mark.asyncio
async def test_authenticate_user(mock_db):
    mock_result = MagicMock()
    mock_db.execute.return_value = mock_result
    test_user = Employee (username="test", hashed_password="test123")
    mock_result.scalar_one_or_none.return_value = test_user

    user = await authenticate_user (mock_db, username: "test", password: "test123")
    assert user is not None, "Пользователь должен быть найден"
    assert user.username == "test"
    user = await authenticate_user(mock_db, username: "test", password:
    "wrong_password")
    assert user is None, "Пользователь не должен быть аутентифицирован с неверным
    паролем"
```



The screenshot shows a terminal window with a dark background. The first line is 'collecting ... collected 1 item' in green. The second line is 'test_auth.py::test_authenticate_user PASSED' in green, followed by a progress bar indicator '[100%]' in green on the right side.

Рисунок 4.1 — Информация о том, что тест пройден успешно

Также были написаны unit тесты для проверки API и работы Websocket. Некоторые из них представлены на Рисунках 4.3 и 4.4.

Листинг 4.2 — Проверка работы Websocket

```
@pytest.mark.asyncio
async def test_get_latest_sensor_readings():
    async with AsyncClient(base_url="http://localhost:8000") as client:
        response = await client.get("/api/sensors/latest")
        assert response.status_code == 200
        data = response.json()
        assert isinstance(data, list)
        for sensor in data:
            assert "sensor_id" in sensor
            assert "sensor_name" in sensor
            assert "value" in sensor
```

Листинг 4.3 — Проверка работы Websocket

```
@pytest.mark.asyncio
async def test_websocket_dashboard (monkeypatch):
    # Заглушка
    async def mock_generate_dashboard_data(db):
        return {
            "sensor_readings": [],
            "recent_alerts": [],
            "production_status": {
                "status": "normal",
                "message": "Test message",
                "metrics": {}
            }
        }

    monkeypatch.setattr(target: "web.app.generate_dashboard_data",
                        mock_generate_dashboard_data)

    # WebSocket
    client = TestClient (app)
    with client.websocket_connect("/ws/dashboard") as websocket:
        data = websocket.receive_json()
        assert "production_status" in data
        assert data["production_status"]["status"] == "normal"
```

Результаты тестирования

Тестирование показало, что все основные компоненты системы функционируют корректно и соответствуют требованиям, представленным в Use Case диаграмме. Все сообщения выводятся согласно описанным сценариям, а данные корректно обрабатываются и сохраняются.

ЗАКЛЮЧЕНИЕ

В рамках данной работы создана комплексная автоматизированная система мониторинга за процессом стерилизации свиных консервов, обеспечивающая мониторинг, анализ и управление производственными процессами в режиме реального времени. Система реализована на основе событийно-ориентированной архитектуры с использованием современных технологий: Python, FastAPI, PostgreSQL, MQTT, Apache Kafka и WebSocket. Созданная система решает основные производственные задачи: непрерывный мониторинг всех этапов производства, раннее выявление отклонений, повышение эффективности производства и снижение влияния человеческого фактора. Практическая значимость разработки заключается в возможности снижения брака, уменьшения энергозатрат, оптимизации режимов работы оборудования, увеличения производительности и сокращения простоев.

Результат работы представляет собой готовое к промышленному внедрению решение, объединяющее современные технологии программной инженерии с глубоким пониманием технологического процесса стерилизации консервов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 31946-2012 Провода самонесущие изолированные и защищённые для воздушных линий электропередачи. Общие технические условия.
2. ГОСТ 34.602-2020 Информационные технологии (ИТ). Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы (с Поправками).
3. ГОСТ Р ИСО 15745-1-2014 Системы промышленной автоматизации и интеграция. Прикладная интеграционная среда открытых систем. Часть 1. Общее эталонное описание.
4. ГОСТ Р ИСО 13374-2-2011 Контроль состояния и диагностика машин. Обработка, передача и представление данных. Часть 2. Обработка данных.
5. PostgreSQL 17: Documentation — URL: <https://www.postgresql.org/docs/current/index.html> (Дата обращения: 01.04.2025).
6. Python 3.13.2 Documentation — URL: <https://docs.python.org/3/index.html> (Дата обращения 30.03.2025).

ПРИЛОЖЕНИЯ

Приложение А

Приложение Б

Приложение А

Функция обработки сообщений с Mqtt брокера

Листинг 3.1 — Создание таблицы

```
def mqtt_client_thread():
    global mqtt_client_instance, mqtt_connected, stop_flag

    # Создаем и настраиваем клиент
    client = mqtt.Client()
    mqtt_client_instance = client

    # Устанавливаем обработчики
    client.on_connect = on_connect
    client.on_message = on_message
    client.on_disconnect = on_disconnect

    # Устанавливаем учетные данные, если указаны
    if config.MQTT_USERNAME and config.MQTT_PASSWORD:
        client.username_pw_set(config.MQTT_USERNAME, config.MQTT_PASSWORD)

    while not stop_flag:
        try:
            if not mqtt_connected:
                logger.info(f"Попытка подключения к MQTT брокеру
{config.MQTT_BROKER}:{int(config.MQTT_PORT)}...")
                client.connect(config.MQTT_BROKER, int(config.MQTT_PORT), 60)

                # Запускаем цикл обработки сообщений
                client.loop_start()

                while not stop_flag:
                    time.sleep(1)
                    if not mqtt_connected:
                        break

                client.loop_stop()

                if stop_flag:
                    break

                logger.info("Переподключение к MQTT брокеру...")
                time.sleep(5)

            except Exception as e:
                logger.error(f"Ошибка MQTT клиента: {e}")
                time.sleep(5)

        try:
            if mqtt_connected:
                client.disconnect()
                logger.info("MQTT клиент отключен")
        except Exception as e:
            logger.error(f"Ошибка при отключении MQTT клиента: {e}")
```

Приложение Б

Код класса, отвечающего за работу по Websocket

Листинг 3.1 — Создание таблиц

```
class ConnectionManager:
    def __init__(self):
        # Словарь подключений по группам
        self.active_connections: Dict[str, List[WebSocket]] = {}
        # Задачи для отправки данных
        self.tasks: Dict[str, asyncio.Task] = {}

    async def connect(self, websocket: WebSocket, group: str):
        """Подключение нового клиента"""
        await websocket.accept()

        if group not in self.active_connections:
            self.active_connections[group] = []

        self.active_connections[group].append(websocket)
        logger.info(f"Клиент подключен к группе {group}, всего подключений: {len(self.active_connections[group])}")

    def disconnect(self, websocket: WebSocket, group: str):
        """Отключение клиента"""
        if group in self.active_connections:
            if websocket in self.active_connections[group]:
                self.active_connections[group].remove(websocket)
                logger.info(f"Клиент отключен от группы {group}, осталось подключений: {len(self.active_connections[group])}")

    async def send_personal_message(self, message: dict, websocket: WebSocket):
        """Отправка сообщения конкретному клиенту"""
        try:
            await websocket.send_json(message)
        except Exception as e:
            logger.error(f"Ошибка отправки сообщения: {e}")

    async def broadcast(self, message: dict, group: str):
        """Отправка сообщения всем подключенным клиентам группы"""
        if group not in self.active_connections:
            return

        disconnected = []
        for connection in self.active_connections[group]:
            try:
                await connection.send_json(message)
            except Exception as e:
                logger.error(f"Ошибка широковещательной отправки: {e}")
                disconnected.append(connection)

        # Удаление отключенных соединений
        for connection in disconnected:
            self.disconnect(connection, group)
```

Продолжение Листинга 3.1

```
    async def start_broadcast_task(self, group: str, interval: float,
data_generator):
        """Запуск задачи для периодической отправки данных"""
        if group in self.tasks and not self.tasks[group].done():
            self.tasks[group].cancel()

        task = asyncio.create_task(self._broadcast_task(group, interval,
data_generator))
        self.tasks[group] = task
        return task

    async def _broadcast_task(self, group: str, interval: float,
data_generator):
        """Периодическая отправка данных всем клиентам группы"""
        while True:
            try:
                # Если нет подключений, просто ждем
                if group not in self.active_connections or not
self.active_connections[group]:
                    await asyncio.sleep(interval)
                    continue

                # Получаем данные от генератора
                data = await data_generator()

                # Отправляем данные всем клиентам группы
                await self.broadcast(data, group)

                # Ждем указанный интервал
                await asyncio.sleep(interval)

            except asyncio.CancelledError:
                logger.info(f"Задача трансляции для группы {group} отменена")
                break
            except Exception as e:
                logger.error(f"Ошибка в задаче трансляции для группы {group}:
{e}")

                await asyncio.sleep(interval)

    def stop_all_tasks(self):
        """Остановка всех запущенных задач"""
        for group, task in self.tasks.items():
            if not task.done():
                task.cancel()
        self.tasks.clear()
```