

|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА**  **Российский технологический университет»**  **РТУ МИРЭА** |

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

**ПРАКТИЧЕСКАЯ РАБОТА**

**по дисциплине**

**«Разработка автоматизированных систем реального времени»**

**Тема: Разработка автоматизированной системы реального времени технологического процесса стерилизации консервов**

|  |  |
| --- | --- |
| Отчет представлен к рассмотрению:  Студент группы КВБО-03-21 | Батурин М.П. |
| (подпись) |  |
| Преподаватель |  |
|  | Зорина Н.В. |
| (подпись) |  |

Содержание

[1 Аналитический раздел 4](#_Toc194678455)

[1.1 Анализ предметной области 4](#_Toc194678456)

[1.2 Краткая характеристика объекта автоматизации 5](#_Toc194678457)

[1.3 Определение требуемых параметров контроля и регулирования для выбранного объекта 7](#_Toc194678458)

[1.3.1 Контролируемые параметры 7](#_Toc194678459)

[1.4 Сбор и анализ требований к АС 8](#_Toc194678460)

[1.4.1 Функциональные требования на разработку АС 8](#_Toc194678461)

[1.4.2 Перечень модулей и их назначение 11](#_Toc194678462)

[1.4.3 Требования к способам обеспечения информационного взаимодействия компонентов системы 12](#_Toc194678463)

[1.4.4 Требования к математическому обеспечению системы 12](#_Toc194678464)

[1.4.5 Дополнительные требования 13](#_Toc194678465)

[1.4.6 Требования по применению систем управления базой данных 13](#_Toc194678466)

[1.4.7 Требования к лингвистическому обеспечению системы 13](#_Toc194678467)

[1.4.8 Требования к разрабатываемому программному обеспечению системы 13](#_Toc194678468)

[1.4.9 Требования к методическому обеспечению 14](#_Toc194678469)

[1.4.10 Требования к контролю, хранению, обновлению и восстановлению данных 14](#_Toc194678470)

[1.4.11 Организационные требования к эксплуатации автоматизированной системы (АС) 14](#_Toc194678471)

[2 Проектирование автоматизированной системы 16](#_Toc194678472)

[2.1 Структурно-функциональное моделирование 16](#_Toc194678473)

[2.2 Проектирование базы данных автоматизированной системы мониторинга процесса стерилизации 16](#_Toc194678474)

[2.3 Архитектурное моделирование 19](#_Toc194678475)

[3 Разработка АС 21](#_Toc194678476)

[3.1 Выбор средств ведения разработки 21](#_Toc194678477)

[3.2 Разработка БД 23](#_Toc194678478)

[3.3 Разработка автоматизированной системы 25](#_Toc194678479)

[3.4 Обработка данных в режиме реального времени 25](#_Toc194678480)

[3.5 Разработка интерфейса 26](#_Toc194678481)

[4 Тестирование автоматизированной системы 31](#_Toc194678482)

[4.1 Основные сценарии тестирования 31](#_Toc194678483)

[4.2 Модульное тестирование 33](#_Toc194678484)

[4.3 Результаты тестирования 35](#_Toc194678485)

[Заключение 36](#_Toc194678486)

[Приложение А 37](#_Toc194678487)

[Функция обработки сообщений с Mqqt брокера 37](#_Toc194678488)

[Приложение Б 38](#_Toc194678489)

[Код класса, отвечающего за работу по web socket 38](#_Toc194678490)

# Аналитический раздел

## Анализ предметной области

Производство свиных консервов начинается с приемки и сортировки сырья, при которой отбирают качественное свиное мясо, проверяя его на соответствие санитарным нормам. Далее следует разделка и обвалка: мясо разделывают на куски, удаляя кости, хрящи и излишки жира. После этого сырье измельчают и солят, режут на мелкие кусочки или перемалывают в фарш, добавляя соль, специи и другие ингредиенты по рецептуре. Следующий этап — приготовление, включающее бланширование или обжарку для улучшения вкуса и удаления лишней влаги. Затем готовую массу фасуют в жестяные или стеклянные банки, добавляя бульон, жир или соус при необходимости. Банки герметично укупоривают под вакуумом, чтобы предотвратить попадание воздуха. Ключевым этапом является стерилизация, при которой консервы нагревают до 110–120°C под давлением для уничтожения бактерий, спор и токсинов; продолжительность обработки зависит от объема банки и обычно составляет 60–120 минут. После этого консервы быстро охлаждают для сохранения качества, наносят маркировку с датой производства и сроком годности, а затем отправляют на хранение, предварительно проверив на герметичность и отсутствие дефектов.

Стерилизация — один из важнейших этапов производства, так как она уничтожает патогенные микроорганизмы, включая возбудителя ботулизма, предотвращая порчу продукта и пищевые отравления. Благодаря стерилизации консервы могут храниться годами без потери безопасности и питательной ценности. Без этой обработки продукт быстро испортится и станет опасным для употребления.

## Краткая характеристика объекта автоматизации

Мы будем рассматривать один из ключевых этапов производства свиной тушенки — **процесс стерилизации**.

На Рисунке 1.1 представлена схема автоклава.

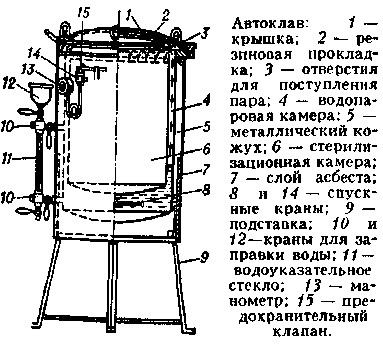


Рисунок 1.1 — Схема автоклава

Стерилизация консервных банок — это ключевой этап в производстве консервов, который обеспечивает безопасность и долговечность продукта. Процесс включает несколько этапов, каждый из которых требует использования определенных инструментов и соблюдения технологий.

Первым этапом является подготовка исходных материалов. Для стерилизации банок необходимо использовать качественные стеклянные или жестяные банки. Важно убедиться, что они не имеют повреждений и трещин. Параллельно с банками подготавливаются крышки, которые также должны быть в идеальном состоянии. Перед началом стерилизации банки тщательно моют с использованием щеток и специального моющего средства, а затем ополаскивают чистой водой.

Следующий этап включает в себя выбор метода стерилизации. Один из самых распространенных способов — это использование автоклава. В этом случае банки помещаются в автоклав, предварительно заполнив их стерилизованным содержимым, и герметично закрываются крышками. Автоклавы работают на основе повышенного давления и температуры, что позволяет эффективно убивать патогенные микроорганизмы и споры. Для этого используется термометр, который позволяет контролировать температуру внутри автоклава, а также манометр для измерения давления.

Если выбран более простой или домашний метод, то можно прибегать к кипячению. В этом случае банки заполняются содержимым и закатываются крышками, после чего помещаются в большую кастрюлю с кипящей водой. Температуру воды контролируют с помощью термометра, а для удобства погружения и извлечения банок применяются специальные щипцы. Важно, чтобы уровень воды был выше уровня содержимого в банках. Время стерилизации зависит от вида консервов и размеров банок, обычно варьируется от 20 до 60 минут.

Для обеспечения устойчивости консервов также могут использоваться водяные бани, где банки загружаются в контейнер с горячей водой. Данный метод также требует контроля температуры и времени, но может быть менее эффективным, чем автоклавирование.

После завершения стерилизации банки необходимо остудить. Если используется автоклав, он должен быть постепенно разряжен до нормального давления, после чего можно открывать крышку. Важно соблюдать осторожность, чтобы избежать ожогов паром. Если банки стерилизовались в кипящей воде, их аккуратно извлекают с помощью щипцов и помещают на деревянные или пластиковые решетки до полного остывания.

Заключительным этапом является проверка герметичности и качества продуктов. После остывания банки осматриваются: крышки не должны быть вогнутыми, а банки — поврежденными. В случае выявления каких-либо дефектов банка должна быть утилизирована. Схема процесса с использованием нотации IDEF0 представлена на Рисунке 1.2.

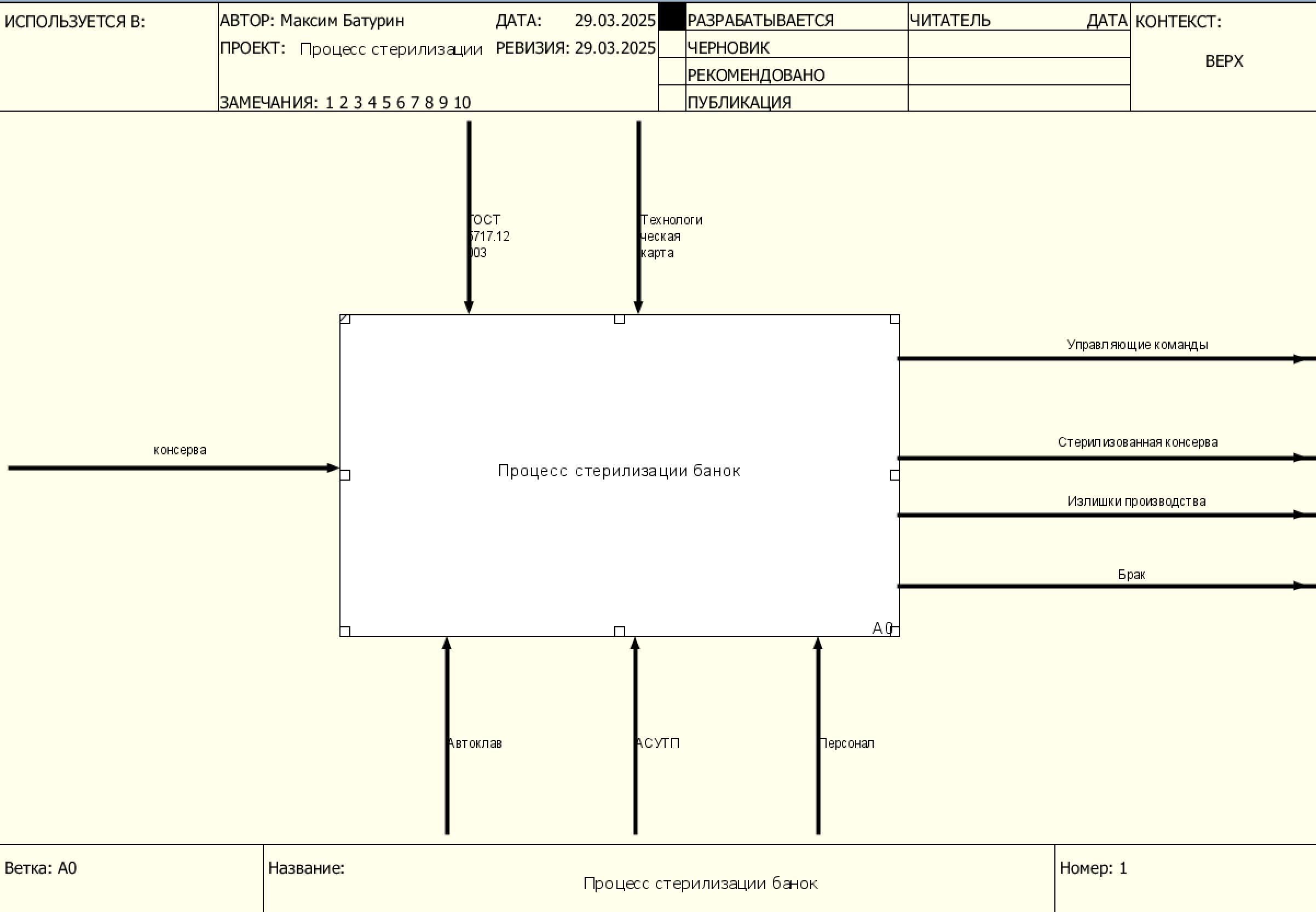


Рисунок 1.2 — Контекстная диаграмма

## Определение требуемых параметров контроля и регулирования для выбранного объекта

Выберем параметры, которые в процессе мы будем или контролировать.

### Контролируемые параметры

К контролируемым параметрам относятся:

* температура стерилизации;
* время стерилизации;
* давление пара в автоклаве;
* скорость нагрева и охлаждения;
* уровень воды в автоклаве.

На основе этого составим Таблицу 1.1.

Таблица 1.1 — Контролируемые и регулируемые параметры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Место нахождения | Измеряемый параметр | Номинальные значения | Единицы измерения |
| 1 | Автоклав | Температура стерилизации | 110-125 | °С |
| 2 | Скорость нагрева | 1-3 | °C/мин |
| 3 | Датчик уровня | Уровень воды в автоклаве | 66 | % |
| 4 | Секундомер | Время стерилизации | 15-120 | мин |
| 5 | Монометр автоклава | Давление пара в автоклаве | 0,15-0,25 | Мпа |
| 6 | Регулятор давления пара на лини подачи | Подача пара | 50-150 | кг/ч |

## Сбор и анализ требований к АС

Цель разработки данной автоматизации — обеспечить полный контроль процесса стерилизации консервных банок.

Функционал разрабатываемой АС следующий:

* мониторинг этапа стерилизации в режиме реального времени с минимальной задержкой отображения информации;
* ограничение несанкционированного доступа;
* получение показаний с датчиков и хранение подробных данных производства для последующего анализа.

### Функциональные требования на разработку АС

Для наглядного представления функциональных требований разработаны UML-диаграммы вариантов использования (Use Case Diagram) и User story, демонстрирующее взаимодействие различных категорий пользователей с системой.

Ниже представлены User story:

1. Оператор линии — «Как оператор линии, я хочу видеть в режиме реального времени показатели температуры, давления и других параметров производства, чтобы оперативно реагировать на отклонения и предотвращать сбои в работе линии.».
2. Инженер-технолог — «Как технолог, я хочу сравнивать параметры разных партий в графиках, чтобы находить причины брака.».
3. Сервисный инженер — «Как сервисный инженер, я хочу получать детальные диагностические отчёты и логи системы, чтобы оперативно выявлять и устранять неисправности, а также планировать профилактическое обслуживание оборудования.».
4. Менеджер производства — «Как менеджер производства, я хочу иметь доступ к агрегированным отчётам о производительности, эффективности и расходах, чтобы принимать обоснованные решения по оптимизации процессов и распределению ресурсов.».

**Use case — «Мониторинг технологических параметров производства».**

Актёры: Оператор линии, инженер-технолог.

**Предусловие**: Система запущена и функционирует в нормальном режиме. Все датчики (температуры, давления, влажности и т.д.) подключены и калиброваны. Оператор авторизован для доступа к системе (через HMI, веб-дашборд).

**Основной сценарий:**

1. Оператор входит в систему через выбранный интерфейс (HMI или веб-панель).
2. Система начинает сбор данных в режиме реального времени со всех датчиков.
3. На экране отображаются актуальные параметры производства, обновляемые с заданной периодичностью.
4. Если все показатели в пределах нормы, оператор продолжает мониторинг.
5. При обнаружении отклонения (например, превышение температуры) система генерирует визуальное и/или звуковое уведомление.
6. Оператор фиксирует сигнал и, согласно установленным процедурам, предпринимает корректирующие действия (например, корректирует работу оборудования или информирует ответственных специалистов).

**Расширения:**

3a. Отсутствие данных от датчика: Если один из датчиков не передаёт данные, система отображает сообщение об ошибке с указанием проблемного элемента и активирует аварийный протокол (например, уведомление сервисного инженера).

**Use case — «Диагностика и техническое обслуживание».**

Актёры: Сервисный инженер.

**Предусловие**: Система ведёт подробное логирование всех событий и аварий. Сервисный инженер имеет доступ к модулю диагностики через защищённый интерфейс.

**Основной сценарий:**

1. Сервисный инженер получает доступ к системе через защищённый веб-интерфейс или специальное диагностическое ПО.
2. Просматривает журналы событий, историю сбоев и логи работы системы.
3. Выявляет несоответствия или сбои (например, частые ошибки датчиков, сбои в работе исполнительных механизмов).
4. Инженер инициирует процедуры диагностики: тестирование датчиков, проверка соединений, калибровка оборудования.
5. По результатам диагностики формируется отчёт, и инженер планирует работы по устранению неисправностей или профилактическому обслуживанию.
6. После проведения работ инженер обновляет статус оборудования в системе, что фиксируется в журнале обслуживания.

**Расширения:**

Отсутствие логов: если система не предоставляет достаточную информацию, сервисный инженер может инициировать ручное тестирование компонентов и задокументировать результаты в системе.

**Use case — «Формирование и анализ отчётов».**

Актёры: Менеджер производства.

**Предусловие**: Система собирает данные с производственной линии в течение заданного периода. Менеджер имеет доступ к аналитическим модулям системы через веб-дашборд или специальное ПО.

**Основной сценарий:**

1. Менеджер производства входит в систему через веб-дашборд.
2. Выбирает раздел формирования отчётов и задаёт параметры отчёта (период времени, ключевые показатели, тип данных).
3. Система обрабатывает собранные данные и генерирует отчёт, включающий такие показатели, как общее количество произведённых бутылок, процент брака, время простоя, энергопотребление и другие KPI.
4. Отчёт отображается на экране в виде графиков, таблиц и диаграмм.
5. Менеджер анализирует данные, принимает решения по оптимизации производственного процесса и может экспортировать отчёт в требуемом формате (PDF, Excel).

**Расширения**:

Недостаточно данных: если за выбранный период данных недостаточно, система уведомляет менеджера с рекомендацией изменить период сбора данных или проверить работу датчиков.

### Перечень модулей и их назначение

Система включает в себя следующие модули:

1. Модуль сбора и хранения данных.
2. Модуль мониторинга значений датчиков.
3. Модуль визуализации.

Модуль сбора и хранения данных выполняет следующие задачи:

1. Считывание параметров с датчиков системы.
2. Организация и хранение необходимых данных.
3. Взаимодействие с базой данных.

Модуль мониторинга выполняет следующие задачи:

1. Отслеживание изменения значений датчиков.

Модуль визуализации выполняет следующие задачи:

1. Отображение информации с помощью дашбордов, графиков и отчетов.

### Требования к способам обеспечения информационного взаимодействия компонентов системы

Информационное взаимодействие компонентов системы должно быть реализовано следующим образом:

1. Модуль сбора и хранения данных должен взаимодействовать с системой через MQTT, OPC UA.
2. Модуль мониторинга значений датчиков должен взаимодействовать с Apache Kafka. Apache Kafka — это распределённая потоковая платформа для обработки событий в реальном времени. Она нужна, чтобы эффективно работать с большими потоками данных, обеспечивая масштабируемость, отказоустойчивость и высокую пропускную способность.
3. Модуль визуализации должен взаимодействовать с программами Grafana/Power BI.

### Требования к математическому обеспечению системы

Математическое обеспечение должно выполнять те расчеты, которые представлены в методах обработки данных. В качестве алгоритмов могут использоваться:

1. Алгоритмы регрессии для прогнозирования отклонений.
2. Алгоритмы кластеризации для анализа брака.
3. Статистические методы для выявления корреляций.

### Дополнительные требования

Дополнительные требования следующие:

1. Независимость базы данных от аппаратной платформы, независимость от сетевого протокола и возможность работы в гетерогенной среде.
2. Обеспечение целостности данных.

### Требования по применению систем управления базой данных

Для хранения информационных массивов системы должна использоваться современная, реляционная, поддерживающая стандарт SQL, промышленная, транзакционная и отказоустойчивая СУБД ― PostgreSQL.

### Требования к лингвистическому обеспечению системы

Пользовательский интерфейс системы по умолчанию должен быть на русском языке. Должны быть реализована возможность переключения на английский язык. Других специальных требований не предъявляется.

### Требования к разрабатываемому программному обеспечению системы

Все программные компоненты должны быть совместимы друг с другом и с выбранной архитектурой системы. Используемое ПО должно обеспечивать необходимые уровни производительности и масштабируемости в зависимости от объёма данных.

Прикладное программное обеспечение в составе системы должно соответствовать следующим основным требованиям:

1. Программирование должно выполняться в средах «Visual Studio» или «Visual Studio Code», с использованием языков C# или Java или Python.
2. Программное обеспечение должно функционировать в операционной системе Windows 10 и выше и взаимодействовать с СУБД.
3. Поддержка русского языка и возможность работать с кириллицей и латиницей.
4. Иметь удобный пользовательский интерфейс.
5. Возможность формирования и вывода отчетных форм.
6. Наличие документации на русском языке для пользователей.

### Требования к методическому обеспечению

Список нормативно-технических документов, применяемых при разработке автоматизированной системы (АС):

* ГОСТ Р 51904-2002 Программное обеспечение встроенных систем.
* ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления.
* ГОСТ 34.602-2020 Комплекс стандартов на автоматизированные системы.

### Требования к контролю, хранению, обновлению и восстановлению данных

Система должна обеспечивать регулярное резервное копирование всех данных на резервный сервер. В процессе копирования данные должны оставаться целостными, даже в случае полной неисправности системы. Также необходимо наличие функционала для восстановления данных из резервного хранилища в случае возникновения сбоев.

### Организационные требования к эксплуатации автоматизированной системы (АС)

**Персонал**: для обеспечения бесперебойной работы системы необходимо наличие следующих категорий сотрудников:

* **операторы**: осуществляют повседневное управление процессами производства компакт-дисков, мониторинг работы оборудования и выполнение стандартных операционных процедур.
* **работники по обслуживанию**: отвечают за профилактическое обслуживание оборудования, а также диагностику и устранение неисправностей;
* **системные администраторы:** обеспечивают поддержку IT-инфраструктуры, управление доступом к системе и защиту данных;
* **разработчики и инженеры:** занимаются обновлением и улучшением функциональности системы, а также разработкой новых программных модулей для оптимизации производственных процессов.

# Проектирование автоматизированной системы

## Структурно-функциональное моделирование

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

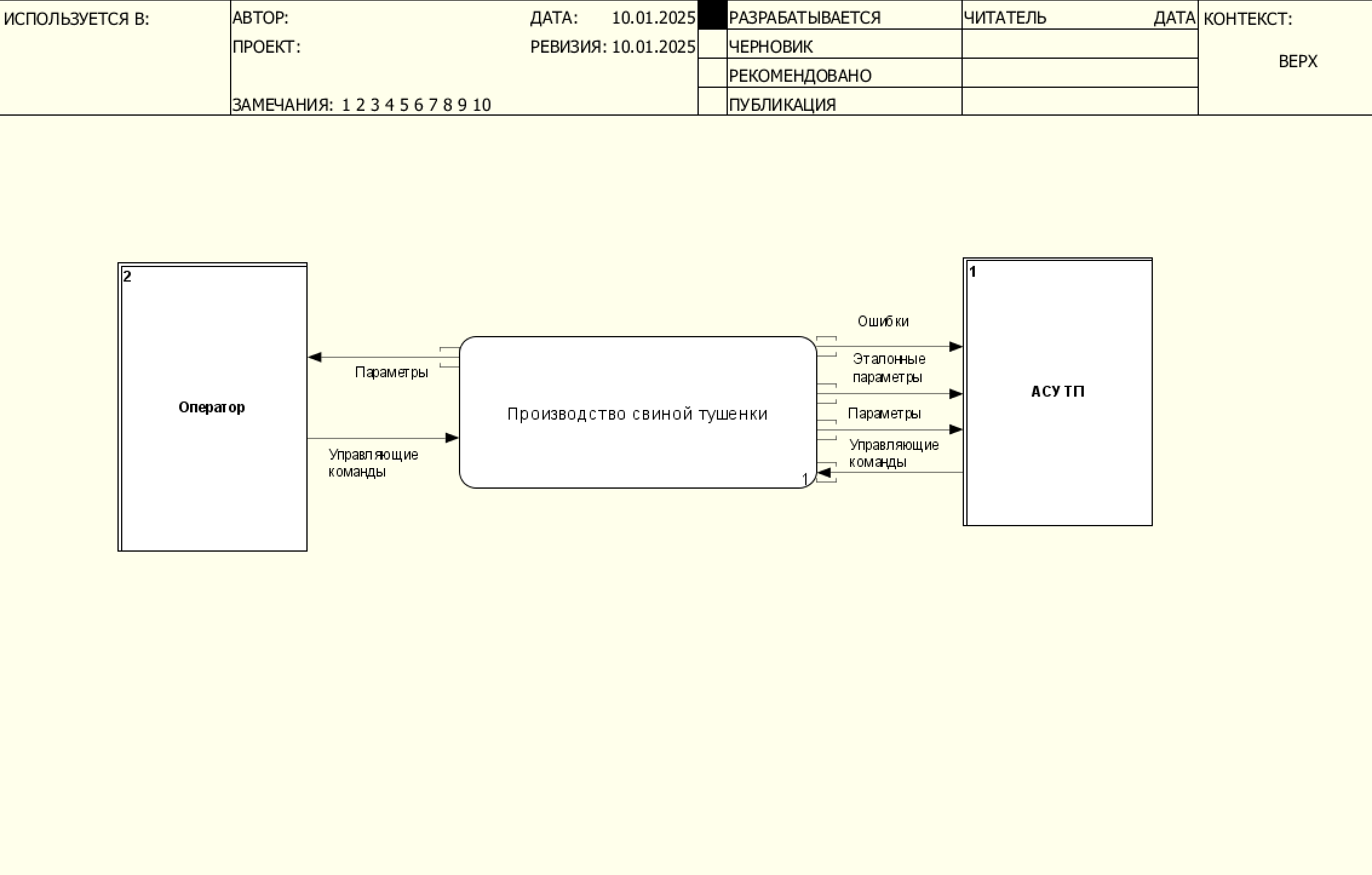


Рисунок 2.1 — DFD диаграмма первого уровня

«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.

## Проектирование базы данных автоматизированной системы мониторинга процесса стерилизации

Основными сущностями базы данных являются:

* оборудование;
* параметры процесса;
* справочные параметры;
* партии;
* пользователи.

Такая структура обеспечивает прозрачное отслеживание процесса и позволяет оперативно реагировать на отклонения.

Схема базы данных представлена на Рисунке 2.2.

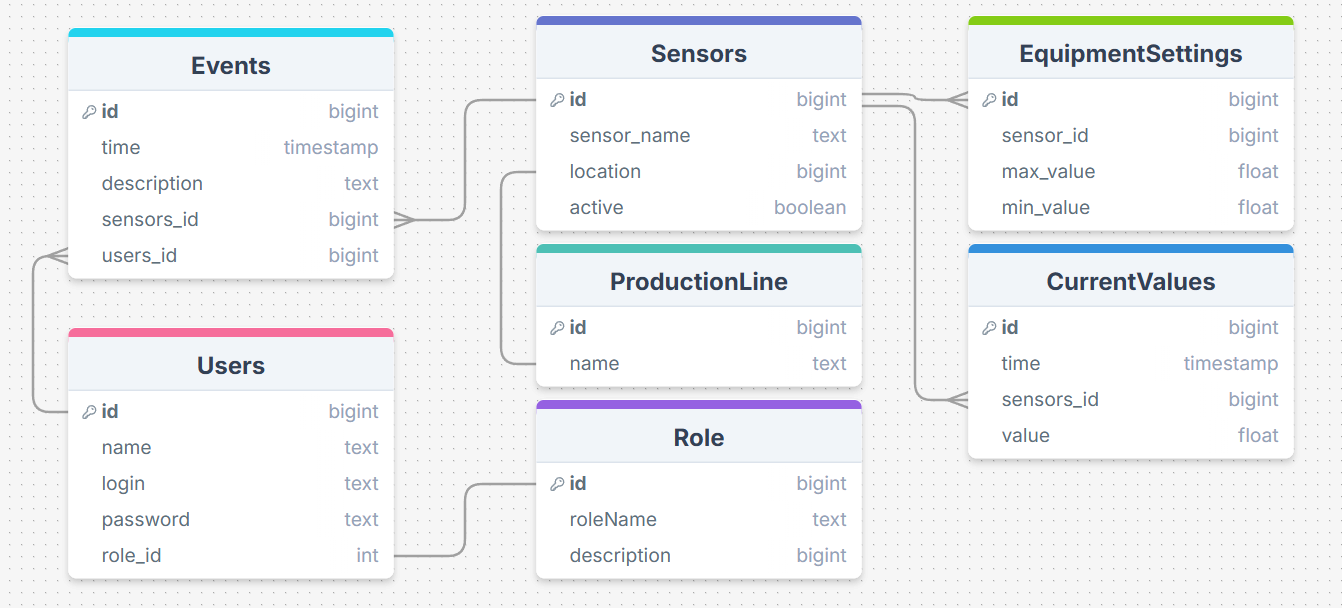


Рисунок 2.2 — Схема логической модели базы данных

1. Основные сущности.

Users (Пользователи).

Хранит информацию о пользователях системы.

Поля: id, name (имя), login, password, role\_id (ссылка на роль).

Связана с таблицей Role через role\_id.

Role (Роли).

Содержит роли пользователей системы.

Поля: id, roleName (название роли), description (описание роли).

1. Производственные сущности.

ProductionLine (Производственные линии).

Содержит информацию о производственных линиях.

Поля: id, name (название линии).

Sensors (Датчики).

Хранит информацию о датчиках оборудования.

Поля: id, sensor\_name (название датчика), location (местоположение), active (активен ли датчик).

Вероятно связана с ProductionLine через location (хотя тип bigint для location выглядит необычно).

1. Данные измерений.

CurrentValues (Текущие значения).

Содержит текущие показания датчиков.

Поля: id, time (временная метка), sensors\_id (ссылка на датчик), value (значение).

Связана с Sensors через sensors\_id.

Events (События).

Фиксирует события в системе.

Поля: id, time (временная метка), description (описание), sensors\_id (связанный датчик), users\_id (связанный пользователь).

Связана с Sensors через sensors\_id и с Users через users\_id.

1. Настройки оборудования.

EquipmentSettings (Настройки оборудования).

Хранит допустимые параметры работы датчиков.

Поля: id, sensor\_id (ссылка на датчик), max\_value (максимальное значение), min\_value (минимальное значение).

Связана с Sensors через sensor\_id.

**Основные связи:**

Users → Role (многие к одному).

Sensors → CurrentValues (один ко многим).

Sensors → Events (один ко многим).

Sensors → EquipmentSettings (один к одному или один ко многим).

Users → Events (один ко многим).

## Архитектурное моделирование

Трехслойная архитектурная модель «Производства компакт-дисков», в которой проведены бизнес, технологический и слой приложений.

Бизнес-слой описывает организацию, её бизнес-процессы, услуги и взаимодействия. Он фокусируется на том, как организация создает ценность и как различные бизнес- процессы взаимодействуют между собой (Рисунок 2.3).

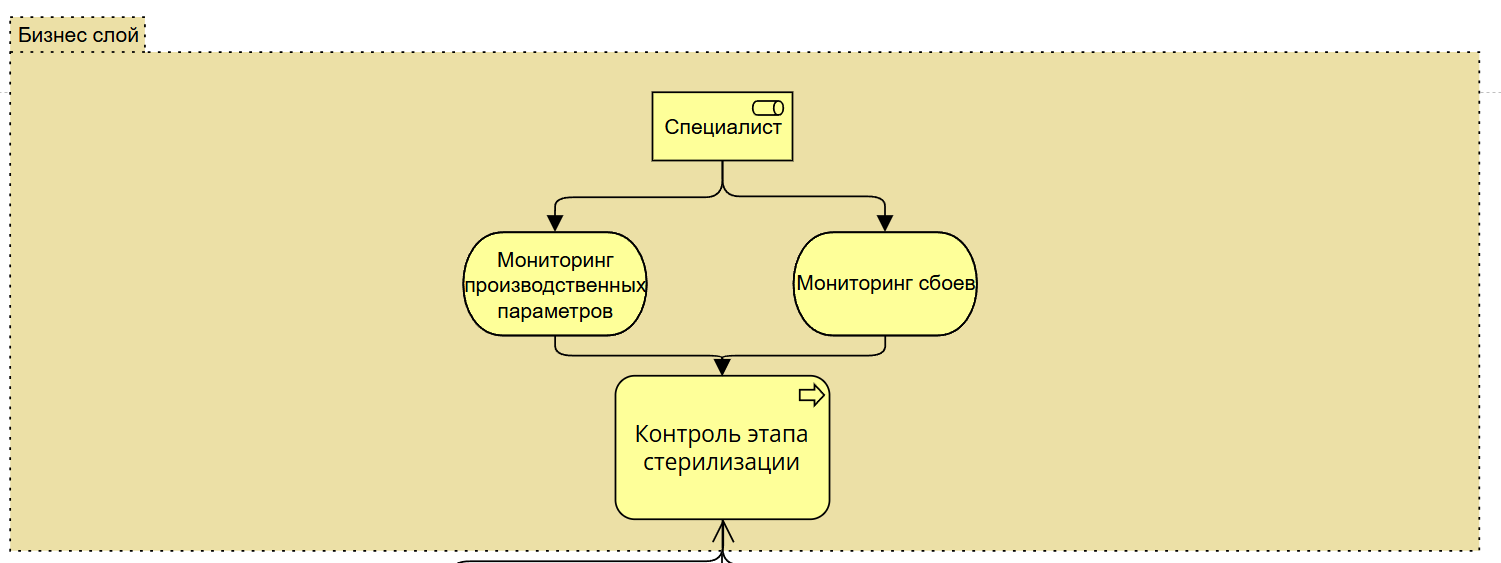


Рисунок 2.3 — Бизнес-слой

Слой приложения описывает программные приложения и их взаимодействия, которые поддерживают бизнес-процессы. Он фокусируется на том, как приложения обеспечивают бизнес-услуги и как они взаимодействуют друг с другом (Рисунок 2.4).

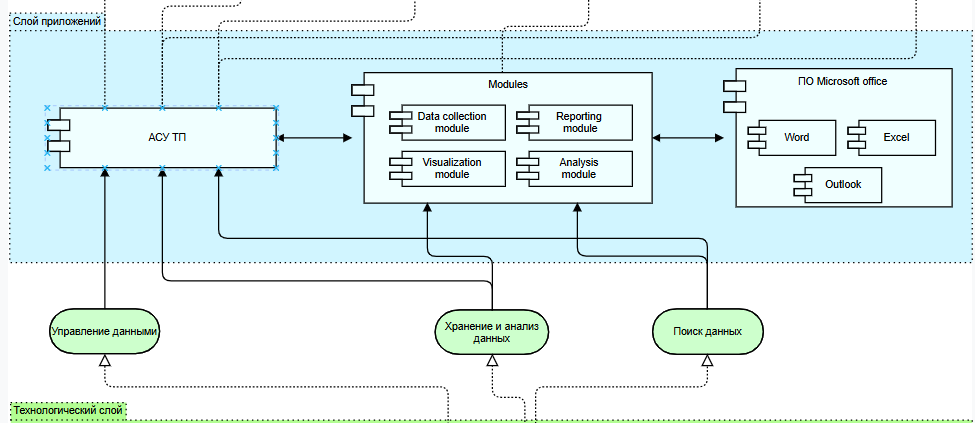


Рисунок 2.4 — Слой приложения

Технологический слой описывает физическую инфраструктуру и технологии, которые поддерживают приложения и бизнес-процессы. Он включает в себя оборудование, сети, платформы и другие компоненты, необходимые для работы приложений (Рисунок 2.5).

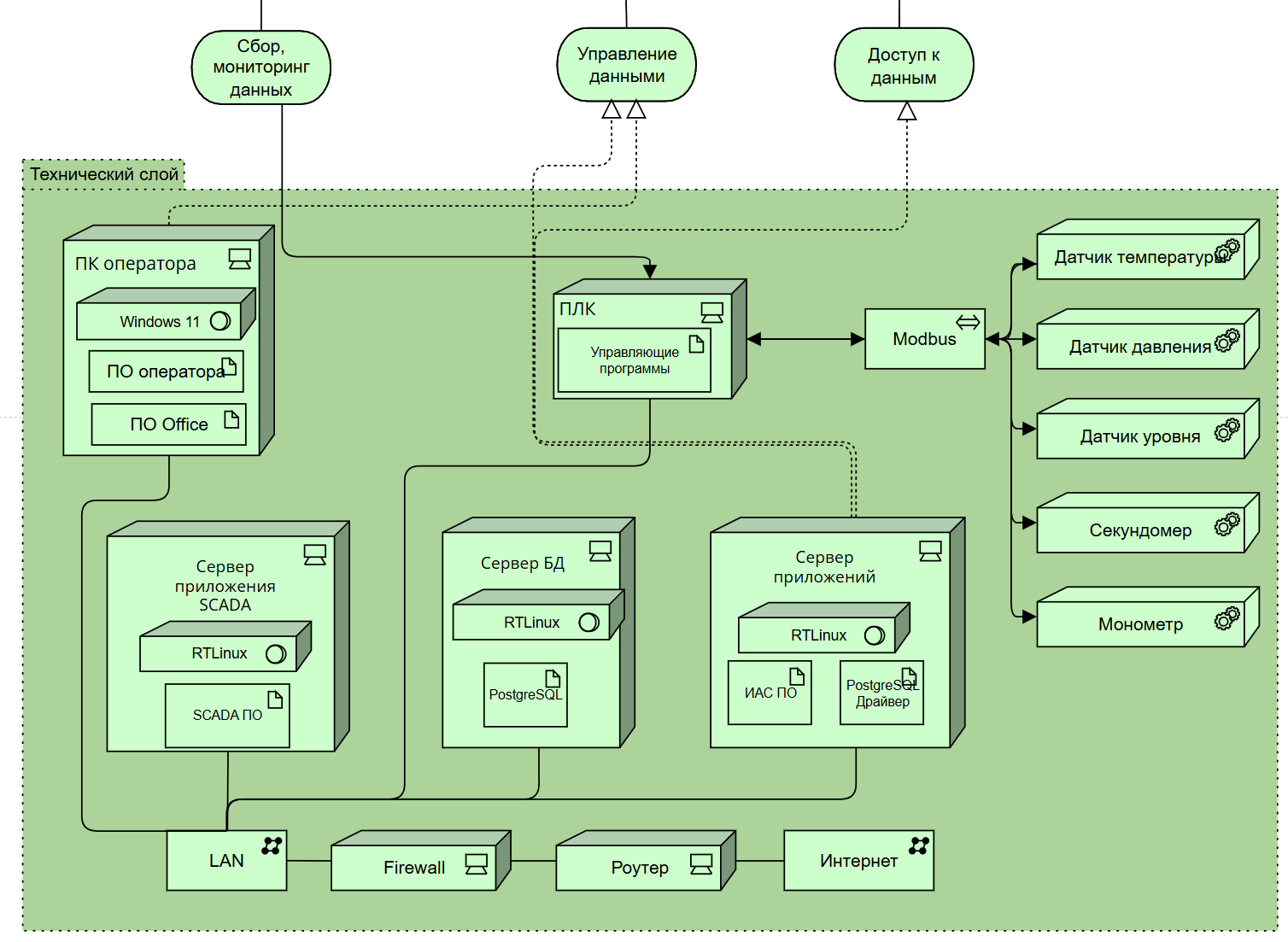


Рисунок 2.5 — Технологический слой

**Вывод**: В результате работы были определены критически важные параметры для мониторинга и управления технологическим процессом, а также сформулированы требования к архитектуре автоматизированной системы. Особое внимание уделено проектированию базы данных, которая обеспечивает прозрачный контроль за производственными партиями, оборудованием и пользователями.

# Разработка АС

## Выбор средств ведения разработки

Для создания системы мониторинга был выбран современный стек технологий, обеспечивающий надежность, масштабируемость и высокую производительность приложения. Основным языком разработки является Python, который обладает богатой экосистемой библиотек для работы с данными и построения веб-приложений.

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.

Для хранения данных используется PostgreSQL — надежная, проверенная временем 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 объемов информации. Взаимодействие с базой данных осуществляется через асинхронный драйвер asyncpg и ORM SQLAlchemy, что позволяет эффективно работать с данными даже при высоких нагрузках.

Архитектура системы основана на принципах событийно-ориентированного программирования и использует брокеры сообщений для обеспечения надежного 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 надежную доставку сообщений и масштабируемость при растущих объемах данных.

Для обеспечения безопасности 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).

Интерфейс пользователя построен с использованием современных веб-технологий HTML5, CSS3 и JavaScript. Для улучшения внешнего вида и создания адаптивного интерфейса применяется фреймворк Bootstrap, обеспечивающий единообразие элементов управления и правильное отображение на различных устройствах. Обновление данных в реальном времени реализовано через WebSockets, что позволяет моментально отображать изменения состояния датчиков и оповещения без необходимости обновления страницы.

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

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

На Рисунке 3.1 представлено взаимодействие сервисов разрабатываемой системы.

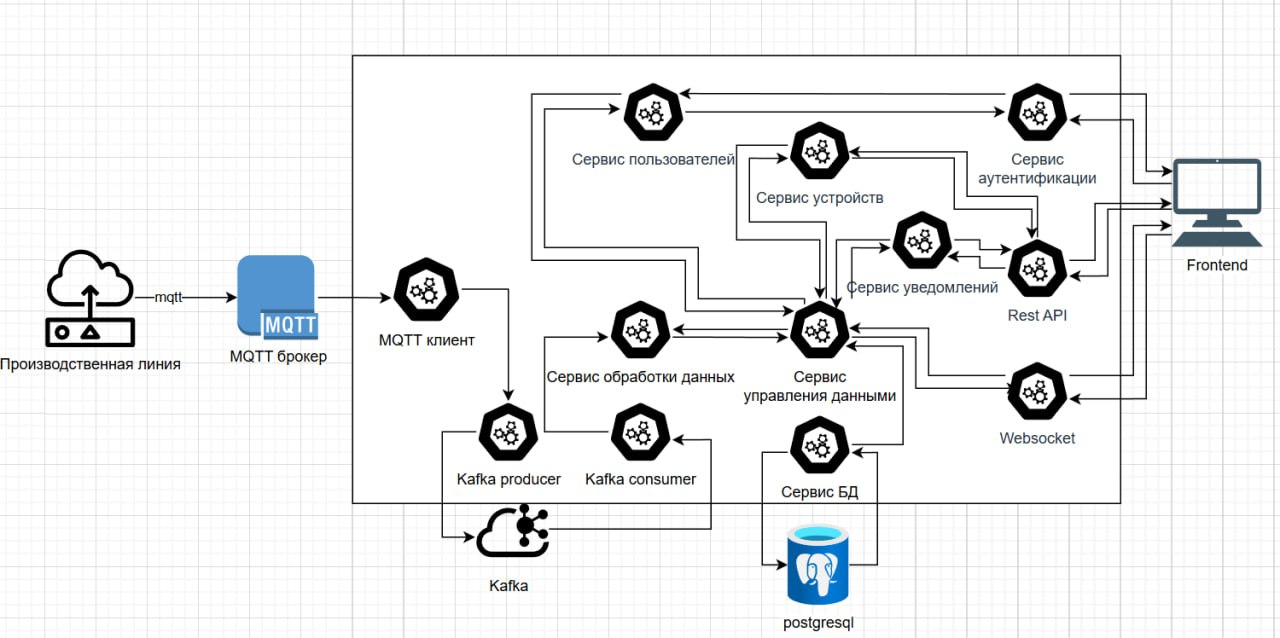


Рисунок 3.1 — Взаимодействие сервисов разрабатываемой системы

## 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 (location). Для каждого датчика определены допустимые диапазоны значений и интервалы измерений, хранящиеся в таблице настроек оборудования (equipment\_settings).

Таблица показаний датчиков (sensor\_readings) представляет собой основное хранилище данных, поступающих от датчиков в реальном времени. Показания хранятся в формате строки JSON, что обеспечивает гибкость при работе с различными типами датчиков и форматами данных. При отклонении показаний от нормы система генерирует события, которые сохраняются в таблице событий (events).

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

Sql скрипт создания таблиц приведен на Рисунке 3.2.

|  |
| --- |
| *Листинг 3.1 Код sql скрипта для создания таблиц* |
| -- Таблица пользователей системы  CREATE TABLE "Users" (  "id" BIGINT NOT NULL,  "name" TEXT NOT NULL,со  "login" TEXT NOT NULL,  "password" TEXT NOT NULL,  "role\_id" INTEGER NOT NULL,  CONSTRAINT "Users\_pkey" PRIMARY KEY ("id")  );  -- Таблица ролей пользователей  CREATE TABLE "Role" (  "id" BIGINT NOT NULL,  "roleName" TEXT NOT NULL,  "description" BIGINT NOT NULL,  CONSTRAINT "Role\_pkey" PRIMARY KEY ("id")  );  -- Таблица датчиков  CREATE TABLE "Sensors" (  "id" BIGINT NOT NULL,  "sensor\_name" TEXT NOT NULL,  "location" BIGINT NOT NULL,  "active" BOOLEAN NOT NULL,  CONSTRAINT "Sensors\_pkey" PRIMARY KEY ("id")  );  -- Таблица текущих показаний  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,  CONSTRAINT "CurrentValues\_pkey" PRIMARY KEY ("id")  );  -- Таблица производственных линий  CREATE TABLE "ProductionLine" (  "id" BIGINT NOT NULL,  "name" TEXT NOT NULL,  CONSTRAINT "ProductionLine\_pkey" PRIMARY KEY ("id")  );  -- Таблица событий системы  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,  CONSTRAINT "Events\_pkey" PRIMARY KEY ("id")  ); |

## Разработка автоматизированной системы

Разработка системы включила несколько ключевых компонентов, реализующих основную функциональность — от сбора данных с датчиков до их обработки и отображения в веб-интерфейсе. В данном разделе приведено описание процесса разработки основных модулей системы и их взаимодействия. Сбор данных с датчиков осуществляется через протокол MQTT, обеспечивающий надежную передачу сообщений даже в условиях нестабильного сетевого соединения. Для реализации клиента MQTT использовалась библиотека paho-mqtt с асинхронной обработкой сообщений.

Функция обработки сообщений с Mqqt брокера приведена в Приложении А.

Разработанный MQTT клиент обеспечивает:

* подключение к MQTT брокеру;
* подписку на необходимые топики;
* обработку входящих сообщений;
* отказоустойчивость при разрыве соединения.

Ключевым компонентом бизнес-логики является модуль обработки и анализа данных, который анализирует поступающие значения и генерирует оповещения при выходе показаний за допустимые пределы.

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

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:

* низкую задержку между получением данных и их отображением;
* масштабируемость системы при увеличении количества датчиков;
* устойчивость к сбоям отдельных компонентов;
* возможность параллельной обработки больших объемов данных.

MQTT (Message Queuing Telemetry Transport) был выбран как основной протокол для сбора данных с датчиков благодаря его легковесности, надежности 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.

## 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:

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

**Структура пользовательского интерфейса.**

Интерфейс системы организован в виде единой веб-панели управления с различными функциональными разделами (Рисунок 3.2).

Основная структура включает:

* Навигационное меню — обеспечивает быстрый доступ ко всем разделам системы.
* Информационная панель (дашборд) — отображает общее состояние производства.
* Детальные страницы датчиков — для просмотра подробной информации по каждому датчику.
* Раздел оповещений и событий — для отслеживания нештатных ситуаций.
* Аналитические отчеты и графики — для анализа производственных показателей.
* Панель настроек — для конфигурирования параметров системы.
* Страница авторизации — для аутентификации пользователей с разными уровнями доступа.

.

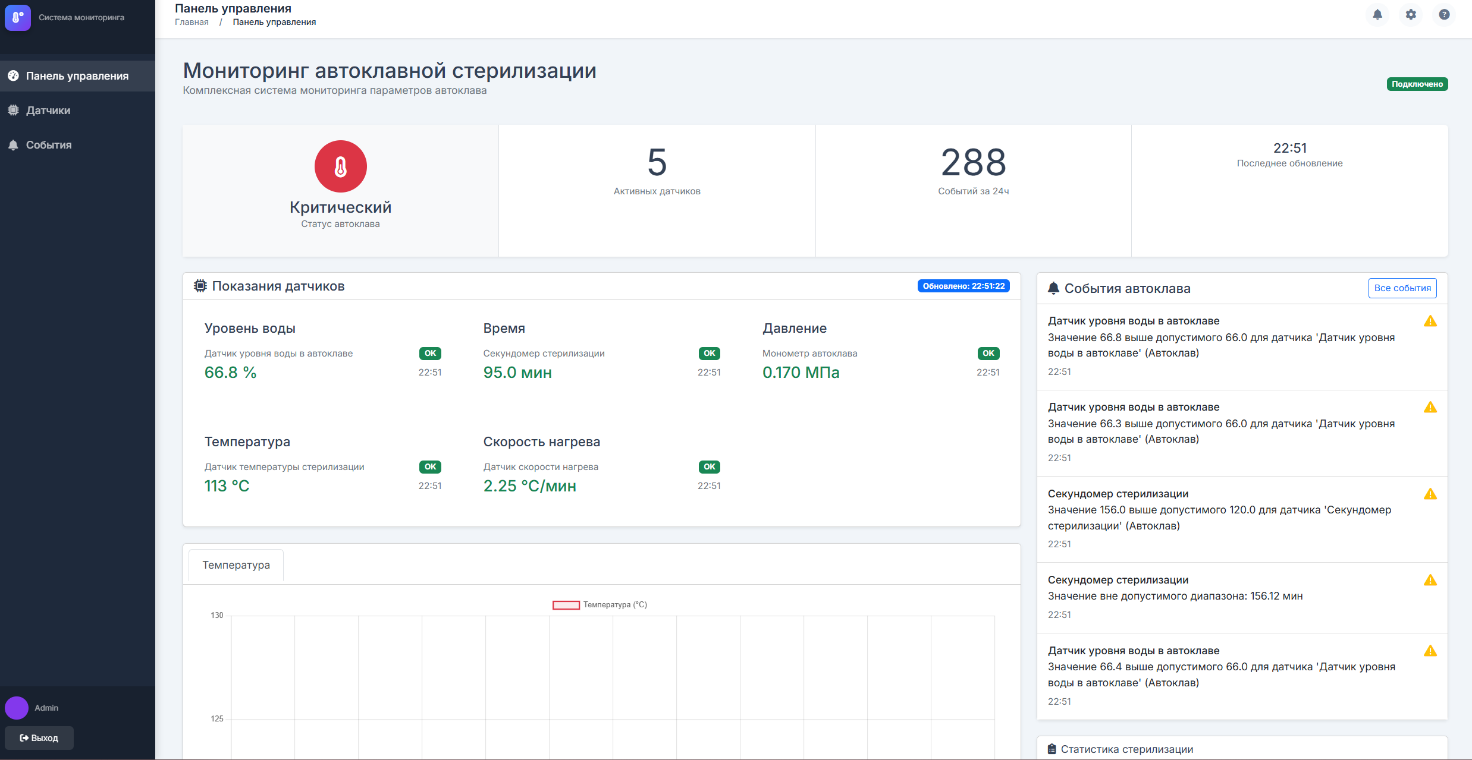


Рисунок 3.2 — Интерфейс автоматизированной системы

Форма аутентификации пользователя приведена на Рисунке 3.3.

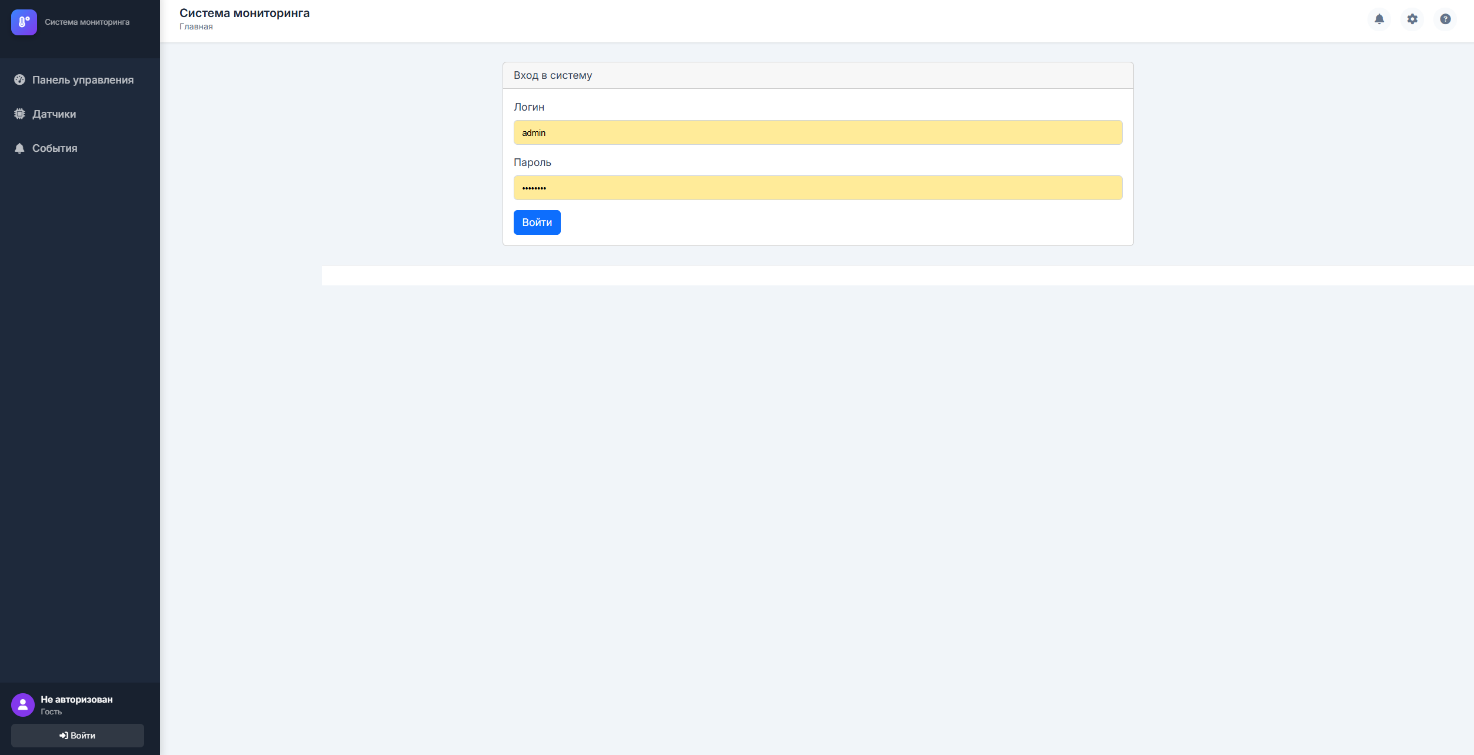


Рисунок 3.3 — Форма аутентификации пользователя

Форма мониторинга датчиков приведена на Рисунке 3.4.

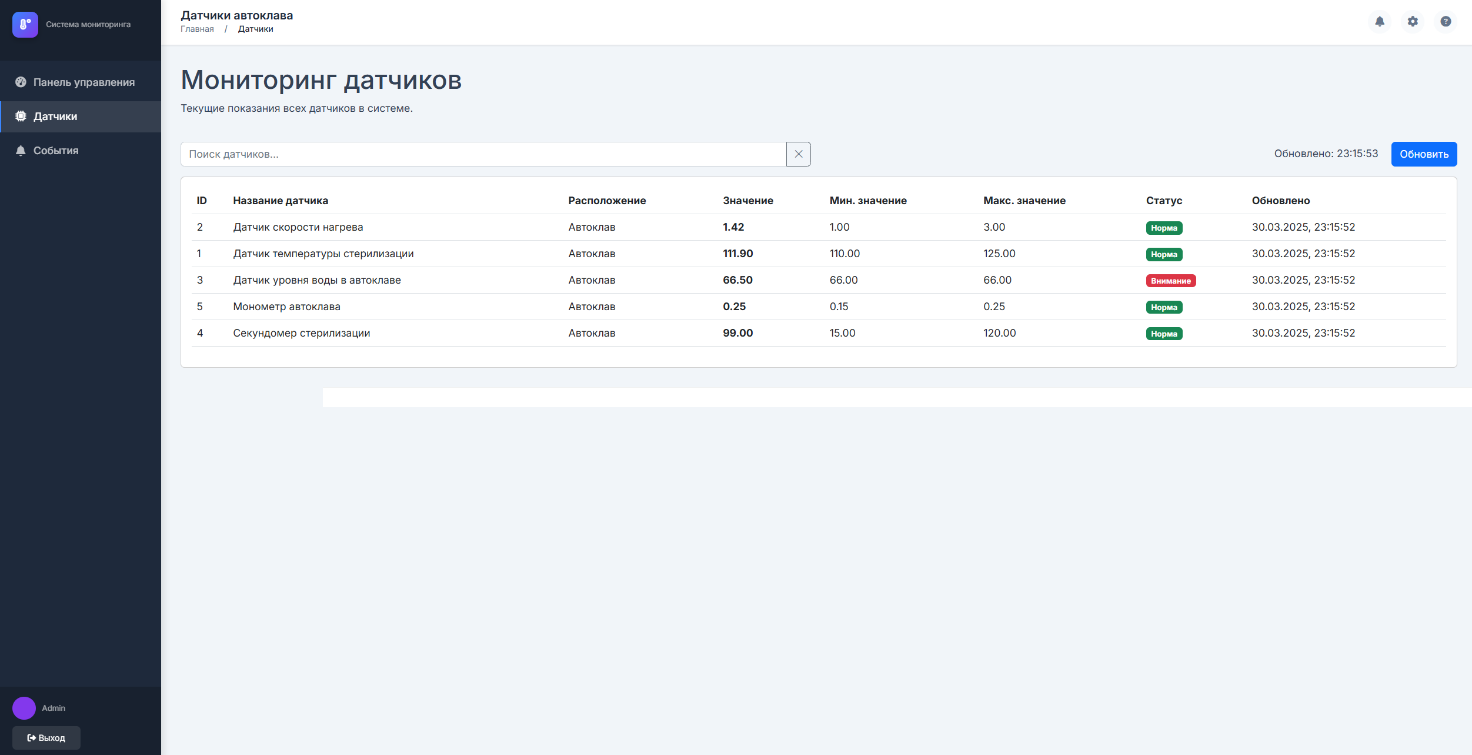


Рисунок 3.4 — Мониторинг датчиков

# Тестирование автоматизированной системы

В процессе тестирования автоматизированной системы мы опирались на Use Case диаграмму, описанную 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.

## Основные сценарии тестирования

**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.
2. Проверить обновление данных от всех типов датчиков (температуры, давления, влажности).
3. 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.

## 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 представлен unit тест для проверки работы аутентификации системы.

На Рисунке 4.1 представлено успешное завершение теста.

|  |
| --- |
| *Листинг 4.1 Код функции для теста аутентификации* |
| import pytest  from unittest.mock import MagicMock  from models import Employee  from services.auth import authenticate\_user  @pytest.mark.asyncio  async def test\_authenticate\_user(mock\_db):  """  Тестирование функции аутентификации пользователя  Проверяет два сценария:  1. Успешная аутентификация с корректными данными  2. Ошибка аутентификации при неверном пароле  """    # Настройка mock-объектов  mock\_result = MagicMock()  mock\_db.execute.return\_value = mock\_result    # Создание тестового пользователя  test\_user = Employee(  username="test",  hashed\_password="test123" # В реальной системе должен быть хеш пароля  )  mock\_result.scalar\_one\_or\_none.return\_value = test\_user  # Тест 1: Успешная аутентификация  user = await authenticate\_user(  mock\_db,  username="test",  password="test123" # В реальной системе сравнение хешей  )    assert user is not None, "Пользователь должен быть найден в системе"  assert user.username == "test", "Имя пользователя должно совпадать"  # Тест 2: Неверный пароль  user = await authenticate\_user(  mock\_db,  username="test",  password="wrong\_password"  )    assert user is None, (  "Система не должна аутентифицировать пользователя "  "с неверным паролем"  ) |

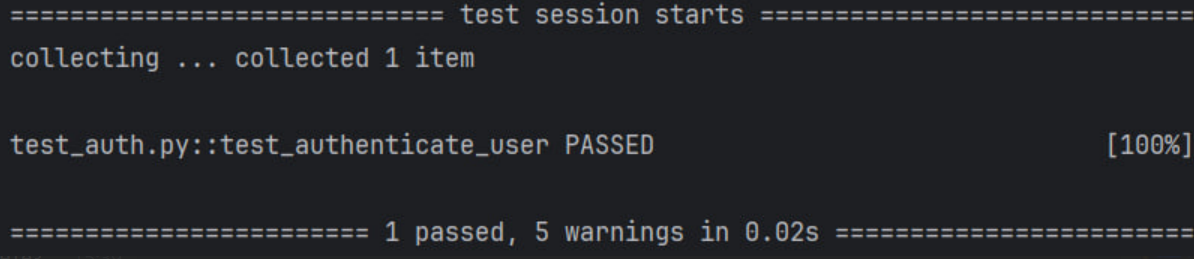


Рисунок 4.1 — Информация о том, что тест пройден успешно

|  |
| --- |
| *Листинг 4.2* Код функции для теста эндпоинта API |
| import pytest  from httpx import AsyncClient  @pytest.mark.asyncio  async def test\_get\_latest\_sensor\_readings():  """  Тестирование эндпоинта получения последних показаний датчиков.  Проверяет:  1. Успешность HTTP-запроса (код 200)  2. Структуру возвращаемых данных  3. Наличие обязательных полей для каждого датчика  """    async with AsyncClient(base\_url="http://localhost:8000") as client:  # Отправка GET-запроса к API  response = await client.get("/api/sensors/latest")    # Проверка статус-кода ответа  assert response.status\_code == 200, (  "API должен возвращать статус 200 при успешном запросе"  )    # Парсинг JSON-ответа  data = response.json()    # Проверка типа возвращаемых данных  assert isinstance(data, list), (  "API должен возвращать список показаний датчиков"  )    # Проверка структуры каждого элемента  for sensor in data:  assert "sensor\_id" in sensor, (  "Каждое показание должно содержать идентификатор датчика"  )  assert "sensor\_name" in sensor, (  "Каждое показание должно содержать название датчика"  )  assert "value" in sensor, (  "Каждое показание должно содержать значение"  ) |

|  |
| --- |
| *Листинг 4.3* Код функции для теста эндпоинта API WebSocket |
| @pytest.mark.parametrize("endpoint", ["/api/sensors/latest", "/api/sensors/history"])  @pytest.mark.asyncio  async def test\_sensor\_endpoints(endpoint):  async with AsyncClient(base\_url="http://localhost:8000") as client:  response = await client.get(endpoint)  assert response.status\_code == 200  data = response.json()  assert isinstance(data, list) |

## 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.

[**https://github.com/Flexer12/Kursach.git**](https://github.com/Flexer12/Kursach.git)-код проекта доступен по данной ссылке

Приложение А

Функция обработки сообщений с Mqqt брокера

|  |
| --- |
|  |
| def mgtt\_client\_thread():  # Создаем клиент MQTT  client = mgtt.Client()  mgtt.client\_instance = client  # Устанавливаем обработчики событий  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)  # Бесконечный цикл повторных попыток подключения  while not stop\_flag:  try:  if not mgtt\_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()  # Ожидаем, пока не будет установлен флаг остановки  while not stop\_flag:  time.sleep(1)  if not mgtt\_connected:  break  # Останавливаем цикл  client.loop\_stop()  # Если установлен флаг остановки, выходим из основного цикла  if stop\_flag:  break  # Иначе ждем немного и пробуем подключиться снова  logger.info("Переподключение к MQTT брокеру...")  time.sleep(5)  except Exception as e:  logger.error(f"Ошибка подключения: {str(e)}")  time.sleep(5) |

Приложение Б

Код класса, отвечающего за работу по web socket

|  |
| --- |
|  |
| class ConnectionManager:  """Менеджер для управления WebSocket-подключениями по группам."""    def \_\_init\_\_(self):  """Инициализация менеджера подключений."""  # Словарь активных подключений по группам  self.active\_connections: Dict[str, List[WebSocket]] = {}    # Словарь задач для отправки данных  self.tasks: Dict[str, asyncio.Task] = {}  async def connect(self, websocket: WebSocket, group: str):  """Подключение нового клиента к указанной группе.    Args:  websocket: WebSocket-подключение клиента  group: Название группы подключения  """  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 от группы.    Args:  websocket: WebSocket-подключение клиента  group: Название группы подключения  """  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): |