

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА — Российский технологический университет»**

**РТУ МИРЭА**

|  |
| --- |
| Институт искусственного интеллекта |
| (наименование института, филиала) |
| Кафедра промышленной информатики |
| (наименование кафедры) |

**ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №4**

по дисциплине «Разработка автоматизированных систем реального времени»

Тема: «Разработка автоматизированной системы мониторинга параметров производства кабельной продукции в реальном времени»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент группы |  | КВБО-03-21 Беликов Михаил Дмитриевич |  |  |
|  | (Ф.И.О., учебная группа) |  | (подпись студента) |
| Преподаватель |  | Зорина Наталья Валентиновна |  |  |
|  | (Ф.И.О.) |  | (подпись преподавателя) |

Работа выполнена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Проверена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

СОДЕРЖАНИЕ

[4 Тестирование системы мониторинга 3](#_Toc194935790)

[4.1 Выбор средств ведения разработки 3](#_Toc194935791)

[4.2 Модульное тестирование 5](#_Toc194935792)

[Результаты тестирования 7](#_Toc194935793)

[ВЫВОД 8](#_Toc194935794)

# 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.
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.

## 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" |

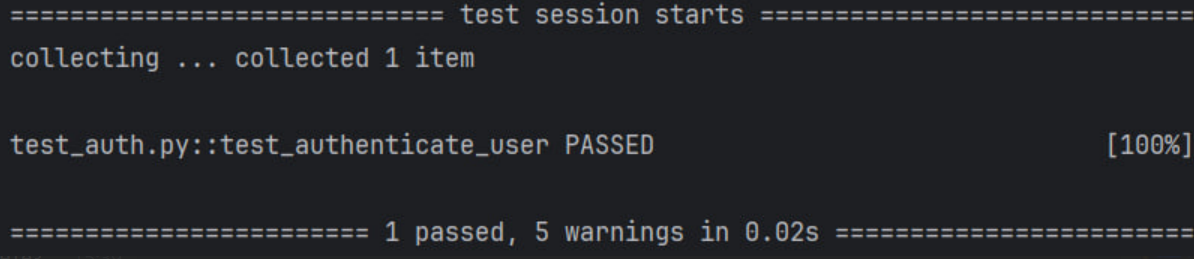


Рисунок .1 — Информация о том, что тест пройден успешно

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.

# ВЫВОД

В ходе выполнения работы было проведено программное тестирование отдельных компонентов системы. С помощью тестирования было установлено, что программа корректно обрабатывает вызовы основных функций и исправно функционирует.