МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

по дисциплине «Тестирование и верификация программного обеспечения»

Практическая работа №5

Студенты группы *ИКБО-50-23, Иващенко А.В.* _____
(подпись)

Преподаватель *Ильичев Г. П.*

_____
(подпись)

Отчет представлен «___»_____202__г.

Москва 2025 г.

1. Цели и задачи практической работы

Цель работы: освоить основные принципы разработки документации на примере «Плана тестирования», опираясь на международные и российские стандарты и познакомиться с различными системами управления тестирования.

Для достижения поставленной цели работы студентам необходимо выполнить ряд задач:

1. Исследовать международные стандарты (например, ISO/IEC/IEEE29119) для разработки тестовой документации.
2. Проанализировать российские нормативные документы (ГОСТы), применяемые при составлении плана тестирования.
3. Изучить типовую структуру и содержание плана тестирования в соответствии с требованиями как международных, так и российских стандартов.
4. Выделить ключевые разделы документа и их назначение.
5. На основе изученных стандартов и типовых разделов составить проект тест-плана для выбранного программного продукта или модуля.
6. Обеспечить документ полным и логичным изложением всех необходимых компонентов (идентификатор, тестируемые элементы, критерии прохождения тестов, график работ и т.д.).
7. Ознакомиться с различными системами управления тестированием, как отечественными (например, Test IT, ТестОпс), так и международными (TestRail, Zephyr, JIRA с плагинами для тестирования).
8. Провести сравнительный анализ функциональности, удобства использования и возможностей интеграции, выбранных TMS.
9. Практически реализовать разработанный тест-план, используя одну из систем управления тестированием.
10. Настроить рабочее пространство, загрузить тестовые случаи и план, выполнить тестирование с фиксацией результатов.

11. Сформировать итоговый отчёт, включающий описание процесса разработки тест-плана, сравнительный анализ стандартов и TMS, а также выводы и рекомендации по улучшению документации и процессов тестирования.

Решение этих задач позволит не только освоить принципы разработки документации по тестированию, но и получить практический опыт работы с системами управления тестированием, а также обеспечить соответствие документации требованиям международных и российских стандартов.

2. Теоретический материал

2.1 Международные стандарты

Стандарты ISO/IEC/IEEE 29119 — это международные стандарты, регулирующие процесс тестирования программного обеспечения. Они разработаны совместно Международной организацией по стандартизации (ISO), Международной электротехнической комиссией (IEC) и Институтом инженеров по электротехнике и электронике (IEEE). Стандарты обеспечивают формальный подход к тестированию ПО, охватывая данный процесс, документацию и методы.

Элементы ISO/IEC/IEEE 29119:

1. ISO/IEC/IEEE 29119-1:2013 определяет основные термины, концепции и принципы тестирования ПО. Включает определения ключевых понятий, таких как уровни тестирования, стратегии и методы тестирования. Устанавливает основы, на которых строятся остальные части стандарта.

2. ISO/IEC/IEEE 29119-2:2013 описывает процессы тестирования, в частности те, которые применяются при разработке программного обеспечения. Включает жизненный цикл тестирования, начиная с планирования и заканчивая анализом результатов. Регламентирует роли и ответственность участников тестирования. Может использоваться как руководство для внедрения эффективного процесса тестирования в организации.

3. ISO/IEC/IEEE 29119-3:2013 определяет форматы и требования к документации, связанной с тестированием. Включает шаблоны для тест-планов, тестовых спецификаций, отчётов о тестировании и других документов. Помогает организациям формализовать процесс тестирования и улучшить прозрачность работ.

4. ISO/IEC/IEEE 29119-4:2015 описывает методы тестирования, применяемые на разных этапах тестирования ПО. Включает такие методы, как функциональное тестирование, нагрузочное тестирование, тестирование

безопасности и др. Определяет, как выбирать методы тестирования в зависимости от типа системы и требований.

2.2 Российские стандарты

Среди российских стандартов выделяют следующие документы:

1. ГОСТ 19.101-77 «Единая система программной документации. Правила оформления документации». Этот стандарт определяет общие требования к оформлению всей технической и эксплуатационной документации, в том числе и тест-планов. Он помогает обеспечить единообразие и структурированность документов.
2. ГОСТ Р ИСО/МЭК 9126-93 «Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению». Этот стандарт используется для оценки качества программного обеспечения. Его применение помогает формализовать критерии, по которым оценивается эффективность тестирования, и может служить основой для определения показателей успешности тест-плана.
3. ГОСТ Р 56920-2016 определяет перечень обязательных разделов для документов, связанных с тестированием (например, тест-плана), включая такие элементы, как идентификатор, описание тестируемых компонентов, критерии успешного прохождения тестов, требования к тестовой среде, распределение ответственности, график работ, анализ рисков и др.

Эти ГОСТы часто применяются в совокупности, позволяя создать комплексную документацию, отвечающую как требованиям качества, так и структурным стандартам. В зависимости от конкретных задач и области применения проекта могут использоваться дополнительные методические указания или внутренние стандарты организации.

2.3 Системы управления тестированием (TMS)

Системы управления тестированием (Test Management Systems, TMS) представляют собой специализированные программные решения, предназначенные для планирования, организации, исполнения и анализа тестирования программного обеспечения. Они обеспечивают централизованное хранение тестовой документации, автоматизацию создания тест-кейсов, контроль выполнения тестов и формирование отчетности. Применение TMS способствует повышению прозрачности тестовых процессов, ускоряет выявление дефектов и улучшает коммуникацию между участниками проекта.

Основные функциональные возможности современных систем управления тестированием включают:

1. Создание, хранение и редактирование тест-планов, тест-кейсов, сценариев и отчетов. Это позволяет стандартизировать подход к тестированию и обеспечить согласованность всей документации.
2. Формирование тестовых наборов (Test Suites), циклов тестирования и календарного плана, что помогает оптимально распределить ресурсы и контролировать сроки выполнения работ.
3. Регистрация прохождения тест-кейсов, автоматическое обновление статусов, создание задач по обнаруженным дефектам и синхронизация с системами отслеживания ошибок.
4. Генерацию подробных отчетов, метрик и диаграмм, которые позволяют оценить качество тестирования и выявить узкие места в процессе разработки.
5. Связь с системами контроля версий, инструментами CI/CD и системами управления проектами, что обеспечивает единый информационный поток между всеми участниками жизненного цикла ПО.

3. Ход работы

3.1 Часть 1 – Разработка плана тестирования

3.1.1 Идентификатор тестового плана

TP-EMPLOYEE-MANAGEMENT-19.101-TEST-PLAN-2025-01-v1.0

3.1.2 Ссылки на используемые документы

- Код приложения «EmployeeManagment» (Практическая работа №3).
- ISO/IEC/IEEE 29119-1:201 – «Software and systems engineering — Software testing — Part 1: Concepts and definitions»
- ISO/IEC/IEEE 29119-2:2013 – «Software and systems engineering — Software testing — Part 2: Test processes»
- ISO/IEC/IEEE 29119-3:2013 – «Software and systems engineering — Software testing — Part 3: Test documentation»
- ISO/IEC/IEEE 29119-4:2015 – «Software and systems engineering — Software testing — Part 4: Test techniques»
- ГОСТ 19.101-77 «Единая система программной документации. Правила оформления документации»
- ГОСТ Р ИСО/МЭК 9126-93 – «Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению»
- ГОСТ Р 56920-2016 – «Информационные технологии. Тестирование программного обеспечения. Часть 1. Основные понятия и определения»

3.1.3 Введение

Цель тестирования — проверить корректность работы модуля, разработанного в Практической работе №3. Используется модуль на языке Python с реализацией операций добавления, поиска, обновления и удаления сотрудников, а также чтение и запись данных в JSON-файл.

Тестирование функционала проводится вручную. Тестирование

проводится с целью проверки функциональной устойчивости к ошибкам и целостности модуля.

3.1.4 Тестируемые элементы

1. Класс DataEmployee — корректность хранения и инициализации данных.
2. Класс EmployeeManagment:
   - add() — добавление сотрудника
   - delete() — удаление сотрудника
   - update() — обновление данных
   - find_by_last_name() — поиск по фамилии
   - get() — получение сотрудника

3.1.5 Проблемы риска тестирования ПП

1. Возможность появления дублирующихся записей при отсутствии проверки уникальности ID.
2. Потеря данных при некорректной работе save().
3. Некорректное чтение JSON-файла после повреждения структуры.
4. Ошибки при отсутствии файла employees.json.
5. Сбои при попытке обновить/удалить/получить несуществующего сотрудника.

3.1.6 Особенности или свойства, подлежащие тестированию

1. Функциональность CRUD: создание, чтение, обновление и удаление сотрудника;
2. Корректность поиска по фамилии;
3. Корректность сериализации/десериализации данных;
4. Устойчивость к ошибочным данным (например, пустые строки);
5. Реакция на несуществующие ID;
6. Формат выходного JSON-файла.

3.1.7 Особенности (свойства), не подлежащие тестированию

1. Производительность;
2. save() и load() — работа с JSON-файлом (запись в файл и чтение из файла данных).

3.1.8 Подход

Используются методы:

– Модульное тестирование функций.
– Функциональное тестирование CRUD-операций.

Тестирования соответствует стандарту ISO/IEC/IEEE 29119-3.

3.1.9 Критерии смоук-тестирования

– Создание файла employees.json.
– Добавление сотрудника работает без ошибок.
– Корректное чтение файла после сохранения.

3.1.10 Критерии прохождения тестов

– Фактический результат соответствует ожидаемому.
– В логах отсутствуют необработанные исключения.
– JSON-файл содержит корректные данные.

3.1.11 Критерии приостановки и возобновления работ

– Файл не создаётся или не читается.
– Метод add() вызывает ошибку.
– Данные теряются после сохранения.

3.1.12 Тестовая документация

– План тестирование.
– Тестирование.
– Отчёт о тестировании.

3.1.13 Основные задачи тестирования

1. Разработать тестирование и среду для ее проведения.

2. Выполнить смоук-тесты.
3. Выполнить функциональные тесты.
4. Зафиксировать результаты.
5. Составить отчет об ошибках.

3.1.14 Необходимый персонал и обучение

– Тестировщик
– Разработчик

Необходимы знания в области тестирования с использованием системы управления тестирования (TMS).

3.1.15 Требования среды

Операционная система: Windows 10/11.

Язык программирования: Python 3.10+.

Файловая система с правами записи.

TMS: Test IT.

3.1.16 Распределение ответственности

Таблица 1 – Роль и ответственность участника проекта

Роль	Ответственность
Тестировщик	Разработка тестирования и составление отчетов по результатам тестирования
Разработчик	Написание программы и ее исправления на основе отчетов тестировщика

3.1.17 График работ

Таблица 2 – Этапы работ с их сроками на выполнение

Этапы работы	Срок
Планировка тестирования	24 часа
Составление тестирования	18 часа
Настройка TMS	1 час
Проведение тестирования	1 час
Составление отчета по тестированию	2 часа

3.1.18 Риски и непредвиденные обстоятельства

- Потеря данных при тестировании.
- Некорректный JSON.
- Ошибки логики в программе.

3.1.19 Утверждение плана тестирования

Таблица 3 – Подписи участников проекта

ФИО	Должность	Дата	Подпись
Иващенко А. В.	Тестировщик, разработчик	12.12.2025	

3.1.20 Глоссарий

CRUD — операции создания, чтения, обновления и удаления.

TMS — система управления тестированием.

JSON — текстовый формат хранения данных.

3.2 Часть 2 – Изучение концепции TMS

3.2.1 Анализ систем управления тестированием

Test IT – отечественная система управления тестированием, разработанная для поддержки создания и ведения тестовой документации, планирования тестовых активностей и интеграции с системами отслеживания ошибок. Продукт ориентирован на повышение эффективности тестирования в российских компаниях.

ТестОпс – российская TMS, позволяющая организовать полный цикл тестирования от планирования и создания тест-кейсов до их исполнения и формирования аналитических отчетов. Система поддерживает интеграцию с другими инструментами разработки и управления проектами, что упрощает совместную работу команды.

3.2.2 Выбор приложения для тестирования

Для выполнения практической работы была выбрана TMS Test IT за счет удобного интерфейса создания тестирований.

3.2.3 Разработка тестирований

1. Тестирование add():

   Предусловия:

   – Файл должен существовать;

   – Объект класса, с которым работаем, должен существовать.

   Шаги:

   1. Создание объекта с данными сотрудника;
   2. Добавление данных сотрудника в файл;
   3. Получение данных сотрудника.

2. Тестирование find_by_last_name():

   Предусловия:

   – Файл должен существовать;

   – Объект класса, с которым работаем, должен существовать.

Шаги:

1. Создание объекта с данными сотрудника;
2. Добавление данных сотрудника в файл;
3. Вызов функции поиска сотрудника по фамилии.

3. Тестирование update():

Предусловия:

– Файл должен существовать;
– Объект класса, с которым работаем, должен существовать.

Шаги:

1. Создание объекта с данными сотрудника;
2. Добавление данных сотрудника в файл;
3. Обновление данных сотрудника;
4. Получение данных сотрудника.

4. Тестирование delete():

Предусловия:

– Файл должен существовать;
– Объект класса, с которым работаем, должен существовать.

Шаги:

1. Создание объекта с данными сотрудника;
2. Добавление данных сотрудника в файл;
3. Удаление данных сотрудника;
4. Получить данные из файла.

5. Тестирование get():

Предусловия:

– Файл должен существовать;
– Объект класса, с которым работаем, должен существовать.

Шаги:

1. Создание объекта с данными сотрудника
2. Добавление данных сотрудника в файл
3. Получение данных сотрудника

3.2.4 Шаги выполнения, ожидаемые результаты и приоритеты

Высокий приоритет у функций add() и get(), так как добавление и получение значений важны для других тестирований. Остальные тестирования имеют средний приоритет.

Соответственно первыми пройдут тестирования add() и get(). Порядок оставшихся трех не имеет принципиального расположения.

Мы ожидаем, что весь функционал пройдет тестирование без вывода исключений.

3.2.5 Подготовка тестирований

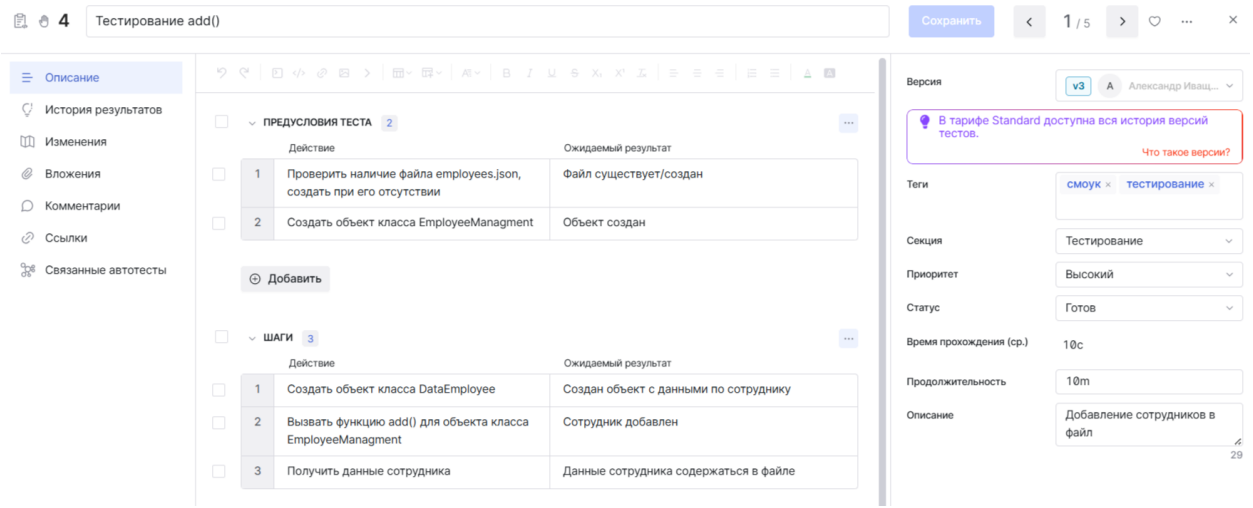

Рисунок 1 – Тестирование функции add()

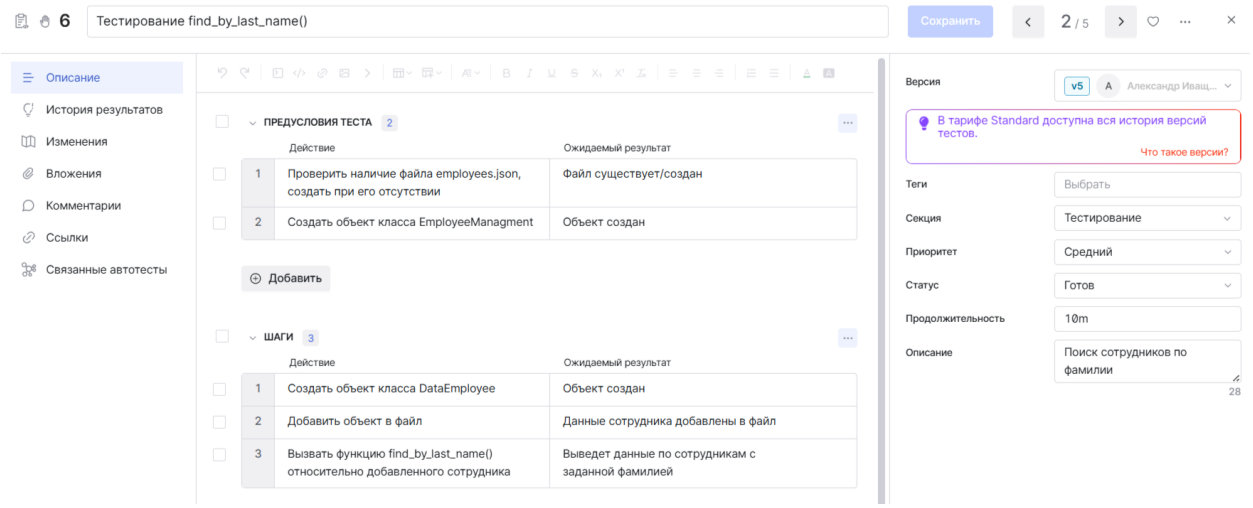

Рисунок 2 – Тестирование функции find_by_last_name()

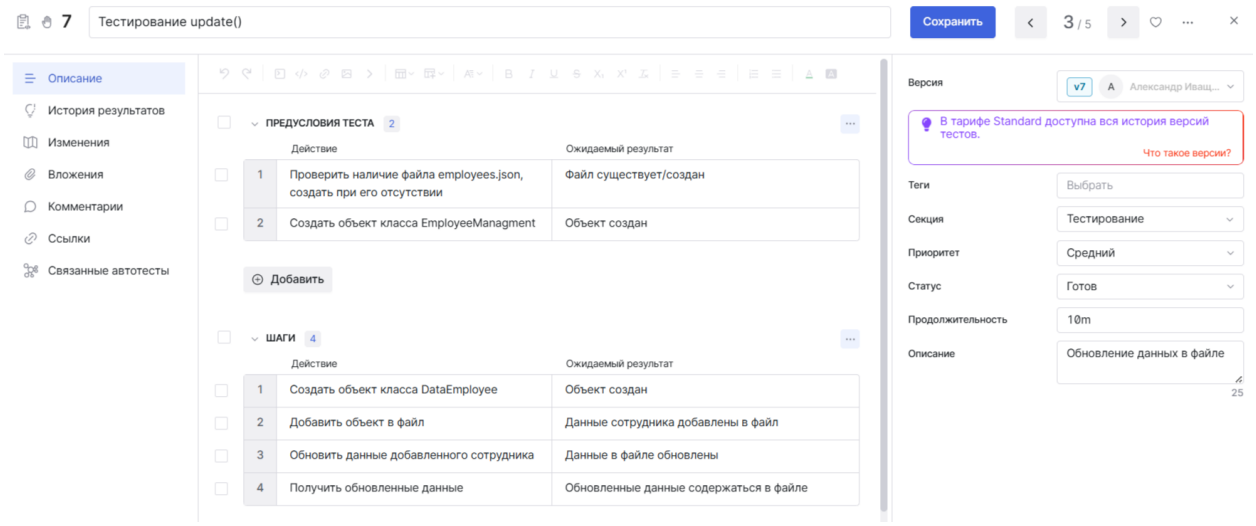

Рисунок 3 – Тестирование функции update()

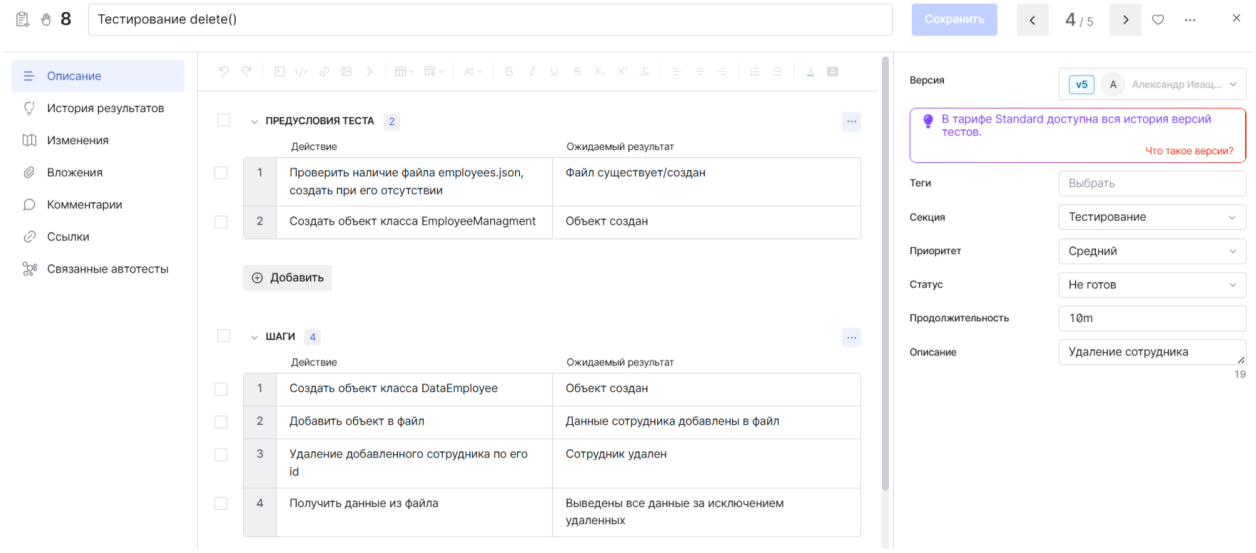

Рисунок 4 – Тестирование функции delete()

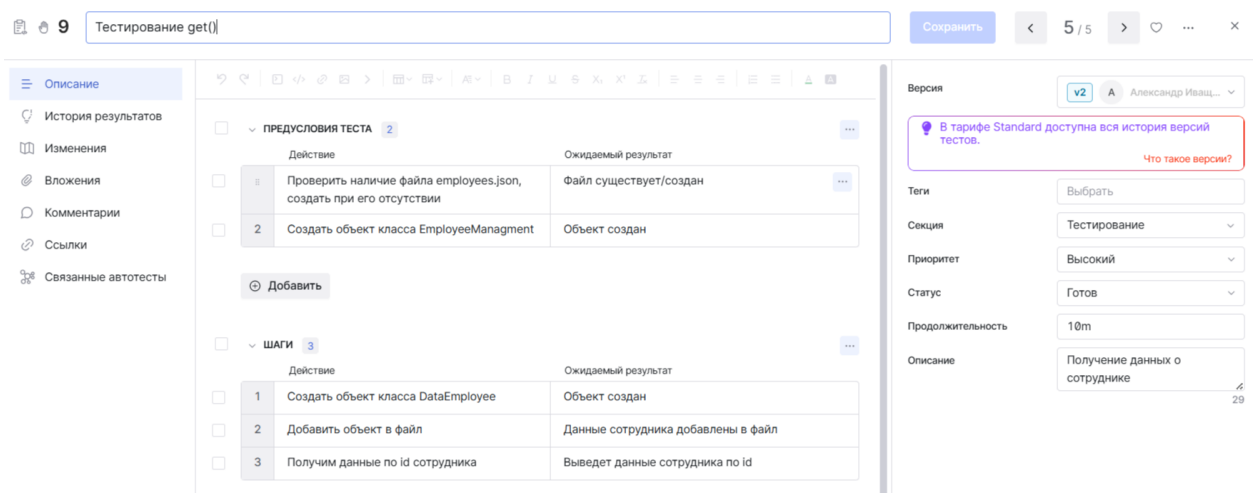

Рисунок 5 – Тестирование функции get()

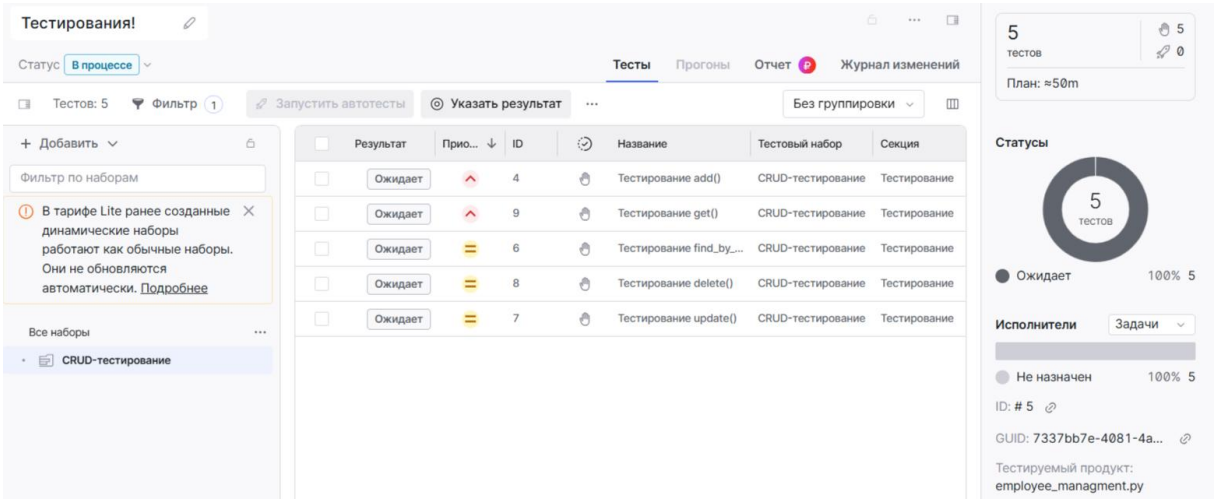

Рисунок 6 – Тест-план на основе составленных тестирований

3.2.6 Результаты тестирования

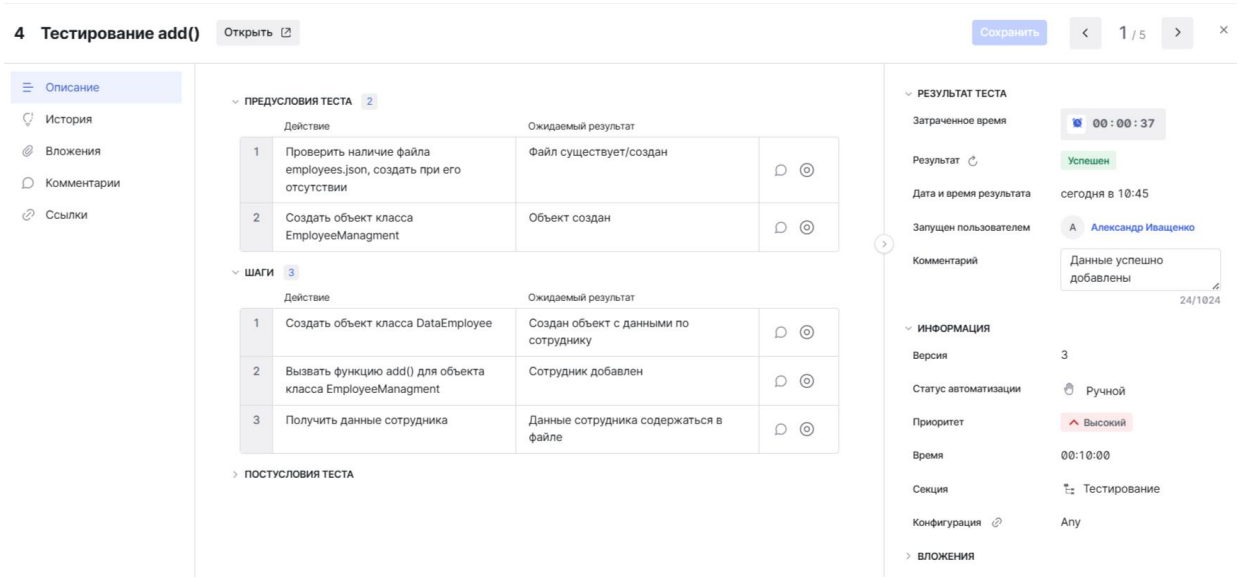

Рисунок 7 – Пройденное тестирование add()

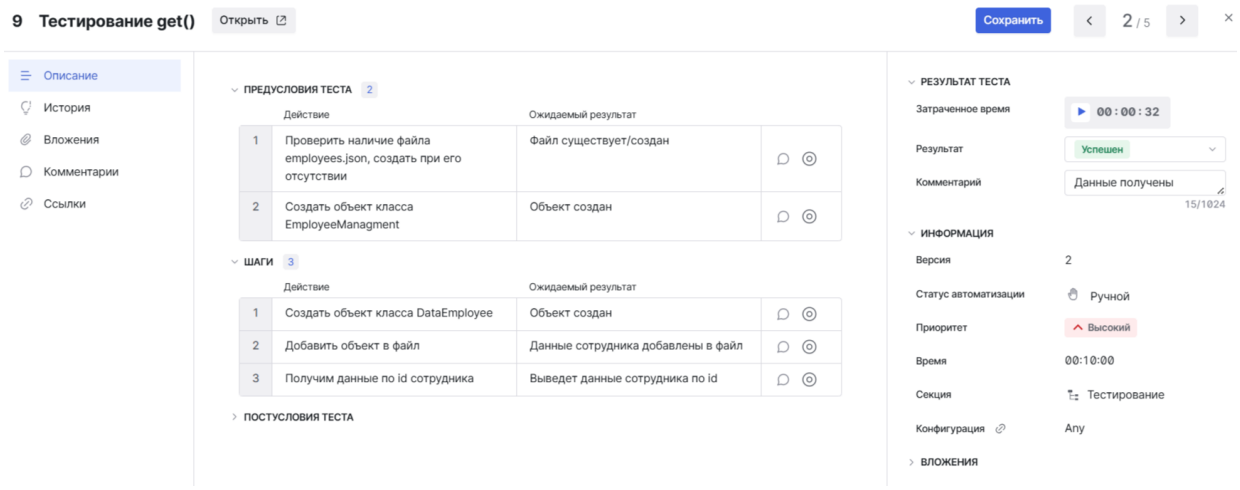

Рисунок 8 – Пройденное тестирование get()

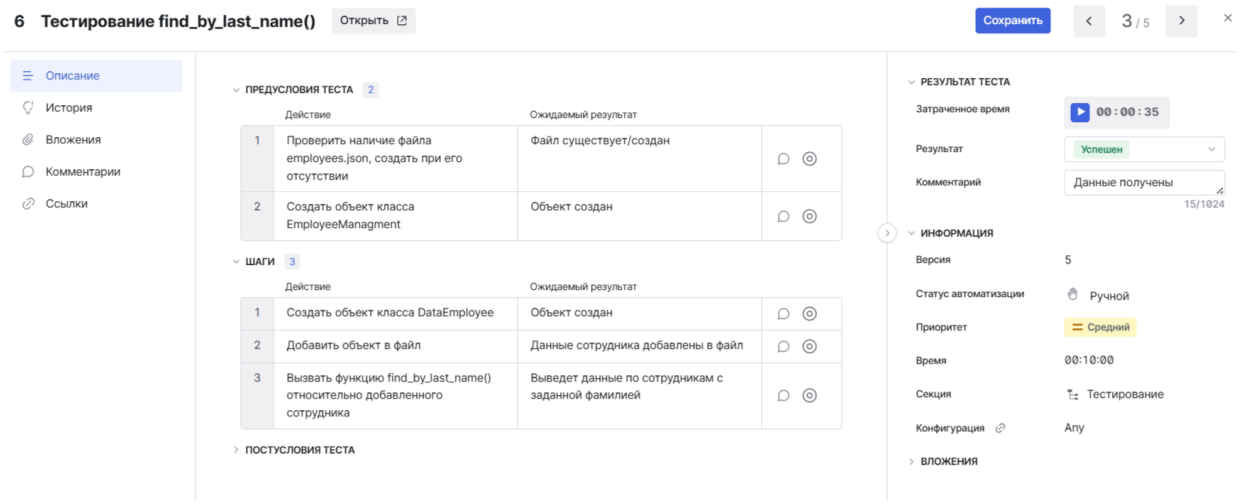

Рисунок 9 – Пройденное тестирование find_by_last_name()

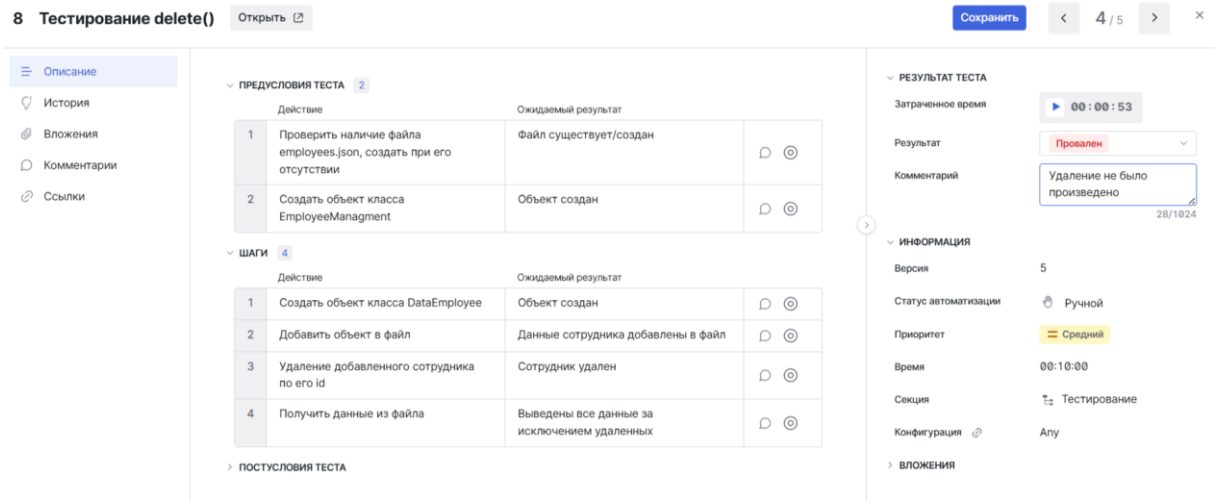

Рисунок 10 – Пройденное тестирование delete()

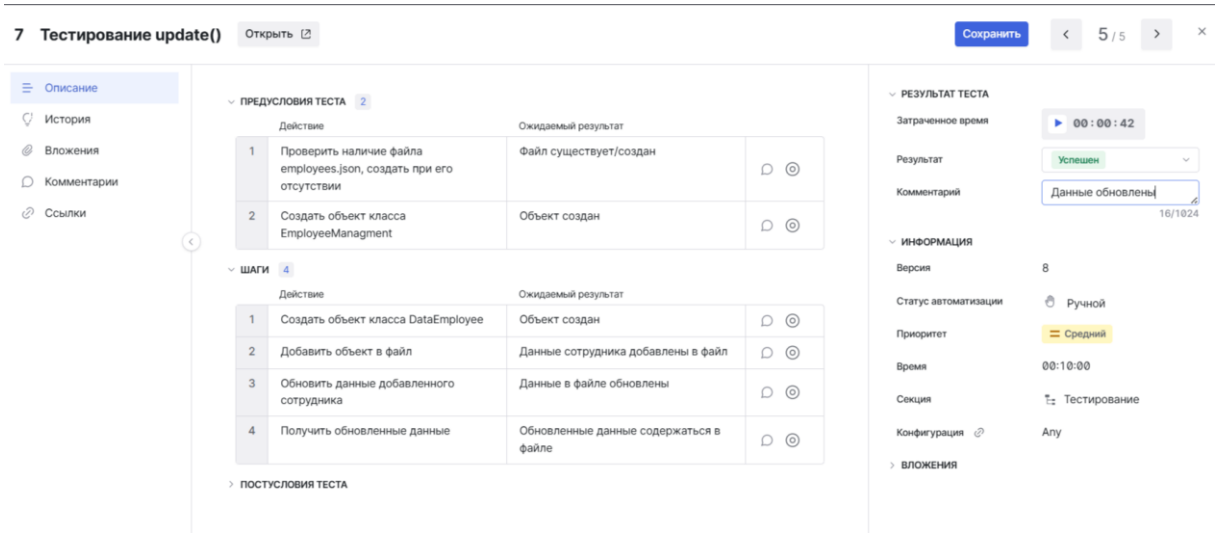

Рисунок 11 – Пройденное тестирование update()

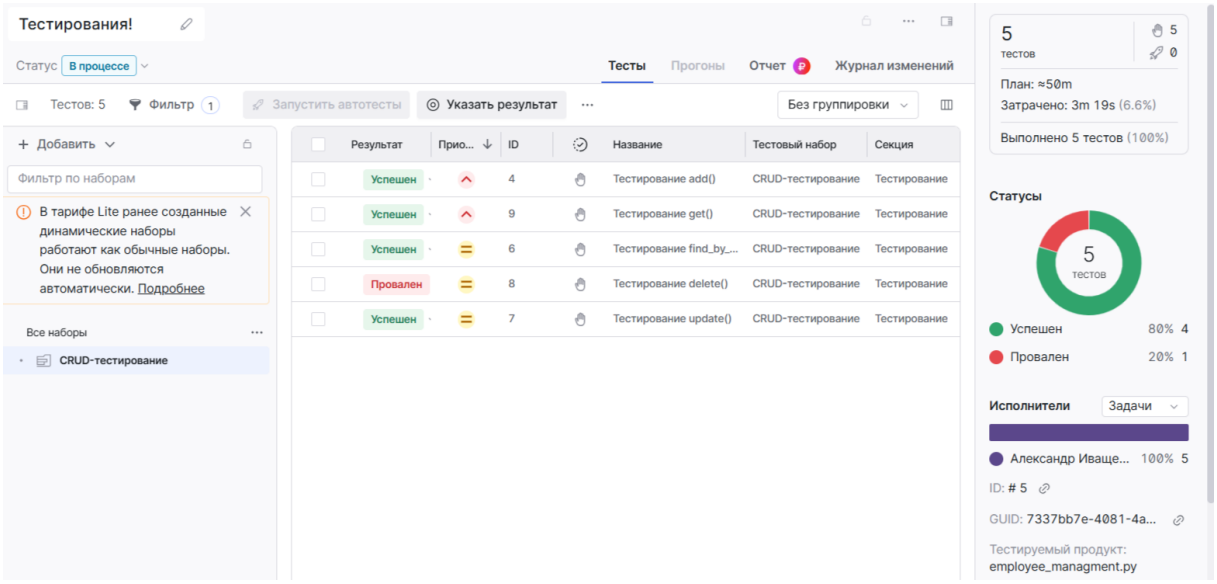

Рисунок 12 – Итоговый результат

3.2.7 Дефекты из TMS

Было провалено тестирование функции delete() – удаление сотрудника не производилось, оставались данные.

3.2.8 Отчет о проделанных тестированиях

В ходе тест-плана были произведены 5 ручных тестирований, из которых были выполнены только 4. Самый важные функции получения и добавления данных сотрудника работают исправно. Также обновление данных и поиск сотрудников по фамилии тоже работают без ошибок. Однако удаление данных по сотрудникам не было произведено из-за ошибочного написания логики функции.

3.2.9 Рекомендации по улучшению тестирования

– Исправить ошибки, выявленные предыдущим тестированием

– Добавить больше сценариев тестирования, для полного покрытия всех возможных исходов работы программы

– Изменить ручное тестирование на автоматическое

– Улучшить структуру изначального кода

4. Заключение

В ходе практической работы были изучены и освоены принципы разработки документации, опираясь на международные и российские стандарты, а также ознакомились с отечественными системами управления тестирования.