Шаблон отчета по учебной практике

ФИО студента: Клюев Тимофей Александрович

Группа 38

**Содержание**

[Введение 3](#_Toc165973515)

[1. Разработка и анализ требований к программной системе 4](#_Toc165973516)

[2. Спецификация технического задания 7](#_Toc165973517)

[3. Технологии разработки программного обеспечения 7](#_Toc165973518)

[4. Построение структуры программного продукта 7](#_Toc165973519)

[5. Тестирование и сопровождение программного обеспечения 7](#_Toc165973520)

[Заключение 8](#_Toc165973521)

[Список использованных источников 9](#_Toc165973522)

[Приложения 10](#_Toc165973523)

**Введение**

Учебная практика «УП.02.01 Осуществление интеграции программных модулей» является частью основной профессиональной образовательной программы подготовки специалистов среднего звена, разработанной в соответствии с ФГОС СПО 09.02.07 «Информационные системы и программирование».

Практика проходила на базе учебной лаборатории колледжа в период с 06.05.2024 г. по 24.05.2024 г.

Целью учебной практики является закрепление теоретических знаний и приобретение практических навыков в области интеграции программных модулей.

В рамках практики я выполнял работы по разработке и доработке программного продукта «HR-Department» — системы автоматизации процессов отдела кадров. Данный проект реализуется как клиент-серверное приложение: серверная часть расположена в репозитории HR-Department, клиентская — в репозитории HR-Department.DeskTop.

Задачи учебной практики:

* Разработка требований к программным модулям на основе анализа документации;
* Интеграция новых модулей в существующую архитектуру проекта;
* Проведение отладки программных компонентов;
* Разработка тестовых наборов и проведение тестирования;
* Поддержка и сопровождение проекта с использованием системы контроля версий.

1. **Разработка и анализ требований к программной системе**

Описание предметной области «HR-Department»:

Автоматизация управления персоналом (Human Resources Management, HRM) является одной из ключевых задач цифровизации в современной организации. Проект «HR-Department» ориентирован на решение широкого спектра задач отдела кадров, связанных с ведением внутренней документации, взаимодействием с персоналом и соблюдением трудового законодательства.

Предметная область охватывает процессы жизненного цикла сотрудника в организации: от поступления на работу до увольнения. Это включает приём новых сотрудников, назначение их на должности, перемещение внутри организации, изменение условий труда, учет стажа и, при необходимости, оформление увольнений. Кроме того, значительную роль играет формирование различных форм — приказов, заявлений, отчетов и других документов, требуемых по нормативным актам и внутренним регламентам компании.

На практике отдел кадров регулярно сталкивается с необходимостью:

* вести актуальный список всех работников предприятия;
* своевременно оформлять и отслеживать кадровые приказы;
* вести учёт по штатному расписанию и вакантным должностям;
* обрабатывать и регистрировать заявления от сотрудников;
* создавать и хранить кадровые документы в соответствии с ГОСТ и ТК РФ;
* взаимодействовать с другими отделами — бухгалтерией, юридическим, IT и др.

Ручное выполнение всех этих задач занимает значительное количество времени и увеличивает риск ошибок. Кроме того, отсутствует прозрачность и история изменений, которая необходима при проведении внутренних проверок или аудитов. Программная система «HR-Department» призвана автоматизировать эти процессы, сократить бумажный документооборот, повысить прозрачность и точность ведения данных, а также обеспечить стандартизированное оформление всех операций.

Проект ориентирован на использование в организациях малого и среднего бизнеса, а также в учебных учреждениях для имитационного моделирования кадровых процессов. Его особенностью является ориентация на удобство пользователя, адаптированность под реальную практику работы кадрового специалиста и наличие гибкой архитектуры, допускающей расширение.

Разработка требований к программной системе «HR-Department»:

Система должна обеспечивать полный цикл управления данными о сотрудниках.

Основной функционал включает:

* Авторизация и аутентификация пользователей с разграничением ролей и уровней доступа.
* Ведение персональных карточек сотрудников с полным набором полей: ФИО, дата рождения, ИНН, СНИЛС, адрес, образование, квалификация, стаж, текущее подразделение и должность.
* Ведение штатного расписания: список всех должностей, количество ставок, оклад.
* Автоматизация оформления документов: приказов о приёме, переводе и увольнении (по типовым формам).
* Возможность загрузки и сохранения шаблонов документов в формате DOCX, с автоматическим заполнением полей.
* Хранение истории изменений и кадровых событий для каждого сотрудника.
* Возможность импорта и экспорта данных.
* Поиск и фильтрация сотрудников по заданным параметрам (подразделение, должность, стаж и др.).
* Отчётность по кадровым показателям: текучесть кадров, список уволенных, структура штата и др.
* Работа с классификаторами (причины увольнений, основания для приёма и пр.).
* Формирование журналов учета заявлений и приказов.

Для обеспечения надежности и удобства эксплуатации, к системе предъявляются следующие нефункциональные требования:

* Архитектура клиент-сервер: централизованное хранение данных и возможность одновременной работы нескольких пользователей.
* Поддержка работы на ОС Windows 10 и выше;
* Интуитивно понятный графический интерфейс с использованием технологий WPF;
* Высокая производительность: минимальное время отклика при стандартных запросах (не более 1 секунды);
* Защита персональных данных: контроль доступа и разграничение прав пользователей;
* Возможность резервного копирования и восстановления базы данных;
* Простота в установке и сопровождении — поставка в виде готового инсталлятора;
* Расширяемость — возможность добавления новых модулей без полной переработки архитектуры.
* Средства реализации
* Проект реализуется на языке программирования C# с использованием современных технологий Microsoft:
* Клиентская часть — WPF (Windows Presentation Foundation), обеспечивающая гибкий и современный пользовательский интерфейс;
* Серверная часть — ASP.NET Core Web API, реализующая REST-интерфейсы для связи между клиентом и сервером;
* База данных — PostgreSQL, свободно распространяемая реляционная СУБД с поддержкой надежного хранения, транзакций и запросов;
* Контроль версий — Git, размещение кода осуществляется в открытых репозиториях GitHub;
* Среда разработки — Visual Studio 2022 и pgAdmin для работы с БД.

Анализ требований проводился путём изучения бизнес-процессов отдела кадров, архитектуры системы, документации и исходного кода, размещенного в репозиториях:

* https://github.com/SobakaTeam/HR-Department — серверная часть;
* https://github.com/SobakaTeam/HR-Department.DeskTop — клиентская часть (на WPF).

Полученные требования легли в основу технического задания, проектирования архитектуры и дальнейшей разработки отдельных модулей системы.

1. **Спецификация технического задания**

Разработка технического задания (ТЗ) является важнейшим этапом жизненного цикла программного обеспечения. На этом этапе определяются цели, задачи, функциональные и технические характеристики будущей системы. Техническое задание выступает основным документом, на основе которого осуществляется проектирование, реализация и последующее тестирование ПО.

В рамках проекта «HR-Department» мной была выполнена частичная формализация технического задания для клиент-серверного приложения автоматизации кадрового делопроизводства. Работа проводилась на основе анализа структуры похожих систем, изучения существующего функционала и требований к предметной области.

Постановка задачи:

Необходимо разработать систему для автоматизации работы отдела кадров, обеспечивающую:

Структура программы:

Система реализуется как распределенное клиент-серверное приложение и включает в себя следующие основные компоненты:

1. Клиентское приложение (WPF):

* Графический интерфейс пользователя.
* Механизмы авторизации и навигации.
* Формы для ввода, редактирования и отображения данных.
* Интеграция с компонентом генерации документов.

1. Серверное приложение (ASP.NET Core Web API):

* Контроллеры API для обработки запросов от клиента.
* Слой бизнес-логики для обработки данных.
* Интеграция с базой данных PostgreSQL.

1. База данных (PostgreSQL):

* Таблицы для хранения информации о сотрудниках, подразделениях, должностях, документах, приказах и др.
* Связи между таблицами для обеспечения целостности данных.

Основные спецификации системы:

1. Функциональные возможности:

* Вход в систему с учётом ролей пользователей.
* Просмотр, добавление, редактирование и удаление данных о сотрудниках.
* Формирование приказов: приём, перевод, увольнение.
* Отображение информации о структуре организации и штатных единицах.
* Ведение архивов кадровых документов.
* Поиск сотрудников по параметрам (ФИО, должность, подразделение и т. д.).
* Просмотр и редактирование классификаторов.

1. Выбор технологий:

* Язык программирования: C#.
* База данных: PostgreSQL.
* Серверная часть: ASP.NET Core.
* Клиентская часть: WPF.
* Система контроля версий: Git (GitHub).

1. Интерфейс:

* Русскоязычный.
* Современный дизайн на основе компонентов WPF.
* Упрощённая навигация и доступ к основным функциям.

1. Требования к качеству:

* Надежность хранения данных.
* Минимальное время отклика интерфейса.
* Возможность расширения функционала без изменения базовой архитектуры.
* Простота установки и настройки системы.

1. **Технологии разработки программного обеспечения**

Проектирование и реализация системы «HR-Department» основаны на современных принципах и технологиях разработки программного обеспечения. Программа реализуется как клиент-серверное приложение с использованием стеков технологий Microsoft и открытых решений. В процессе разработки применяются инструменты, обеспечивающие модульность, читаемость кода, повторное использование компонентов и устойчивость системы к изменениям.

В процессе проектирования были использованы следующие принципы:

* Модульность — система разделена на отдельные компоненты, каждый из которых выполняет строго определённую функцию. Это упрощает сопровождение и тестирование.
* Слой бизнес-логики — чёткое разделение пользовательского интерфейса и логики обработки данных.
* Client-Server архитектура — клиент отправляет запросы, сервер их обрабатывает и возвращает результат. Это повышает масштабируемость и безопасность системы.
* Использование шаблонов проектирования — например, паттерн Repository используется для доступа к данным, и паттерн MVC.
* Валидация данных — реализована на стороне сервера и клиента для повышения надёжности.

На этапе проектирования использовались диаграммы UML, такие как:

* Use Case Diagram — для описания взаимодействия пользователей с системой.
* Class Diagram — для отображения структуры классов и связей между ними.
* Sequence Diagram — для визуализации сценариев обмена данными между клиентом и сервером.

Создание диаграмм помогло определить состав классов, понять структуру взаимодействия между слоями и заранее спроектировать наиболее важные компоненты.

В процессе разработки происходила работа со следующими инструментами:

* Visual Studio 2022 — основная среда разработки для серверной и клиентской части. Использовалась для написания, компиляции и отладки C# кода.
* pgAdmin — для работы с базой данных PostgreSQL. С помощью этого инструмента создавались таблицы, ключи и выполнялись SQL-запросы.
* Postman — применялся для тестирования REST API запросов, создания коллекций и отладки маршрутов серверной части.
* Git + GitHub — система контроля версий и платформа для размещения исходного кода. Работал с ветками, коммитами, слиянием изменений и разрешением конфликтов.

Вся система реализована на языке C#, который является надёжным, универсальным и широко используемым языком для создания корпоративных приложений.

Архитектура системы:

1. Клиентская часть (WPF):

* Используется Windows Presentation Foundation — платформа для создания интерфейсов с гибкой привязкой данных (Data Binding).
* Применяются шаблоны MVVM и возможности XAML для разметки UI.
* Поддержка стилей, шаблонов управления и команд для упрощения логики обработки.

1. Серверная часть (ASP.NET Core Web API):

* RESTful API для обмена данными между клиентом и сервером.
* Контроллеры обрабатывают HTTP-запросы (GET, POST, PUT, DELETE).
* Использование Entity Framework Core для доступа к данным через ORM-модель.

1. База данных (PostgreSQL):

* Создание схемы базы с таблицами.
* Настройка внешних ключей, индексов, ограничений целостности.
* Хранение данных в UTF-8 с поддержкой русского языка.

1. **Построение структуры программного продукта**

Проект имеет трёхуровневую архитектуру, включающую следующие уровни:

1. Уровень представления (Presentation Layer) — клиентское приложение на WPF:

* Предоставляет графический интерфейс для взаимодействия с пользователем.
* Выполняет запросы к серверу, отображает данные, полученные в ответе.
* Реализует шаблон MVVM (Model-View-ViewModel) для разделения логики и интерфейса.

1. Уровень бизнес-логики (Business Logic Layer) — серверная часть на ASP.NET Core:

* Обрабатывает полученные от клиента данные.
* Реализует правила предметной области, например, проверку условий перевода сотрудника.
* Обеспечивает взаимодействие с базой данных через ORM (Entity Framework Core).

1. Уровень доступа к данным (Data Access Layer) — PostgreSQL:

* Хранит всю информацию о сотрудниках, подразделениях, должностях, приказах, заявлениях и прочих объектах.
* Обеспечивает целостность данных с помощью ключей и ограничений.

Компоненты системы:

Клиентское приложение (WPF):

Основное окно приложения (Main Window) — основной, начальный элемент приложения содержит имеющиеся в БД отделы и краткую информацию о них. Так же с этого окна можно перейти к редактированию уже существующего или созданию нового отдела.

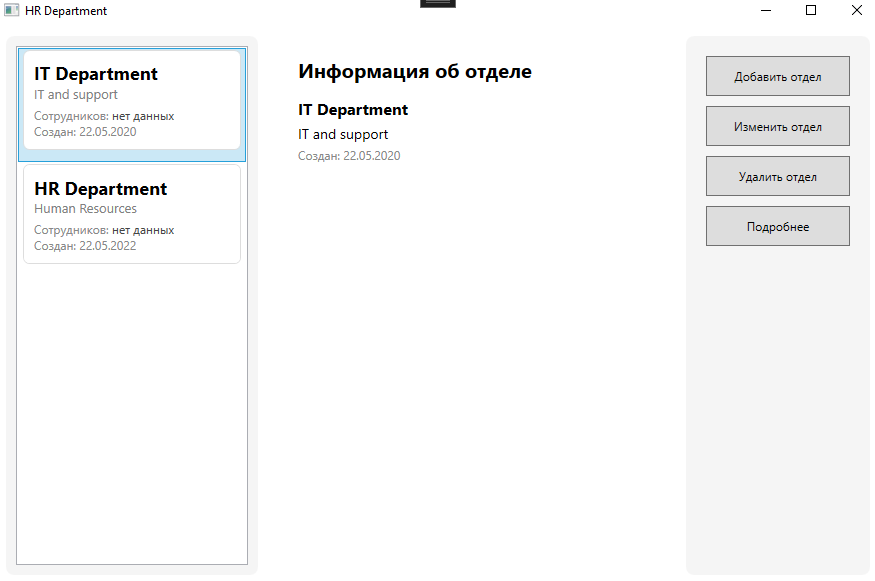


Рисунок 4.1 — Основное окно приложения (Main Window)

Окно отдела (Department Window) — Страница для работы с сотрудниками, и просмотра их сокращённых данных. Содержит информацию непосредственно об самом отделе.

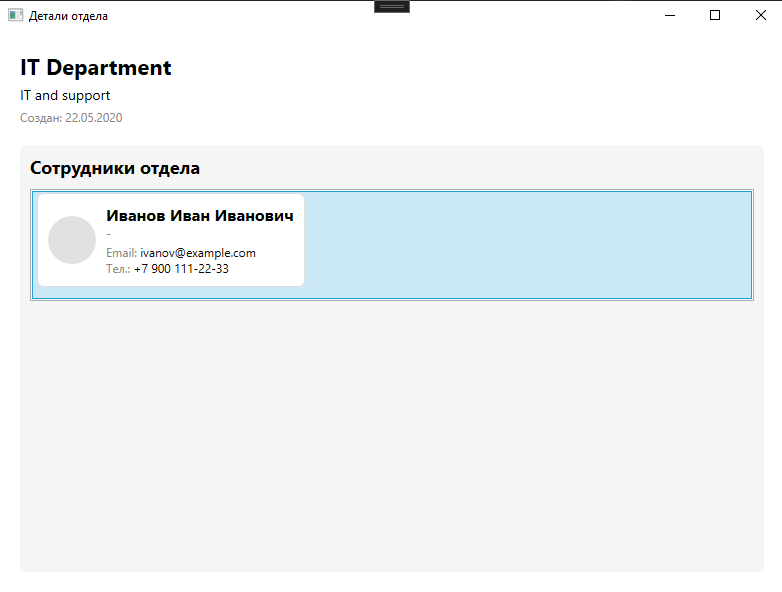


Рисунок 4.2 — Окно отдела (Department Window)

Окно сотрудника (Person Window) — Всплывающее окно просмотра данных сотрудника выбранного на окне отдела.

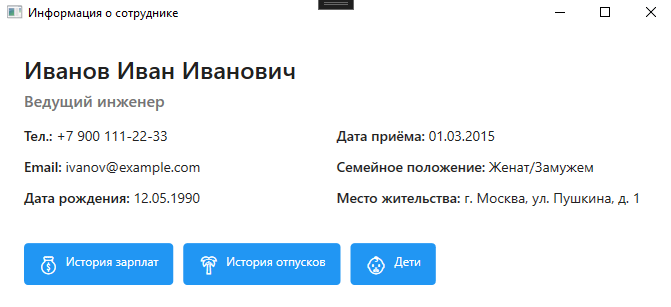


Рисунок 4.3 — Окно сотрудника (Person Window)

Окно дополнительных данных сотрудника (Sub Data View) — Можно перейти по нажатию на любую кнопку дополнительных данных снизу окна сотрудника. Содержит структурированные данные в зависимости от выбранной категории.

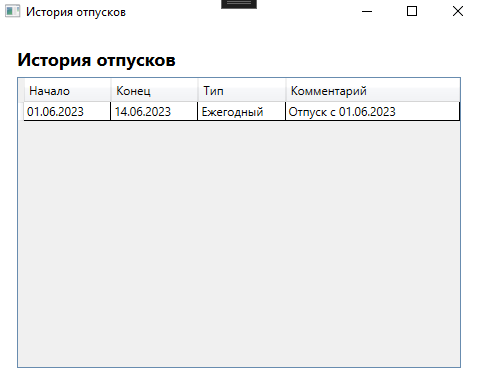


Рисунок 4.4 — Окно дополнительных данных сотрудника (Sub Data View)

Серверное приложение (Web API):

* Контроллеры: PersonController, DepartmentController, PositionsController, SalaryController и другие.
* DTO (Data Transfer Objects) для передачи данных между клиентом и сервером.
* Службы (Services) реализуют бизнес-логику и вызываются из контроллеров.
* Слой репозиториев для взаимодействия с Entity Framework Core.

База данных (PostgreSQL):

Развернутая на локальной машине (сервере) имеет следующую архитектуру:

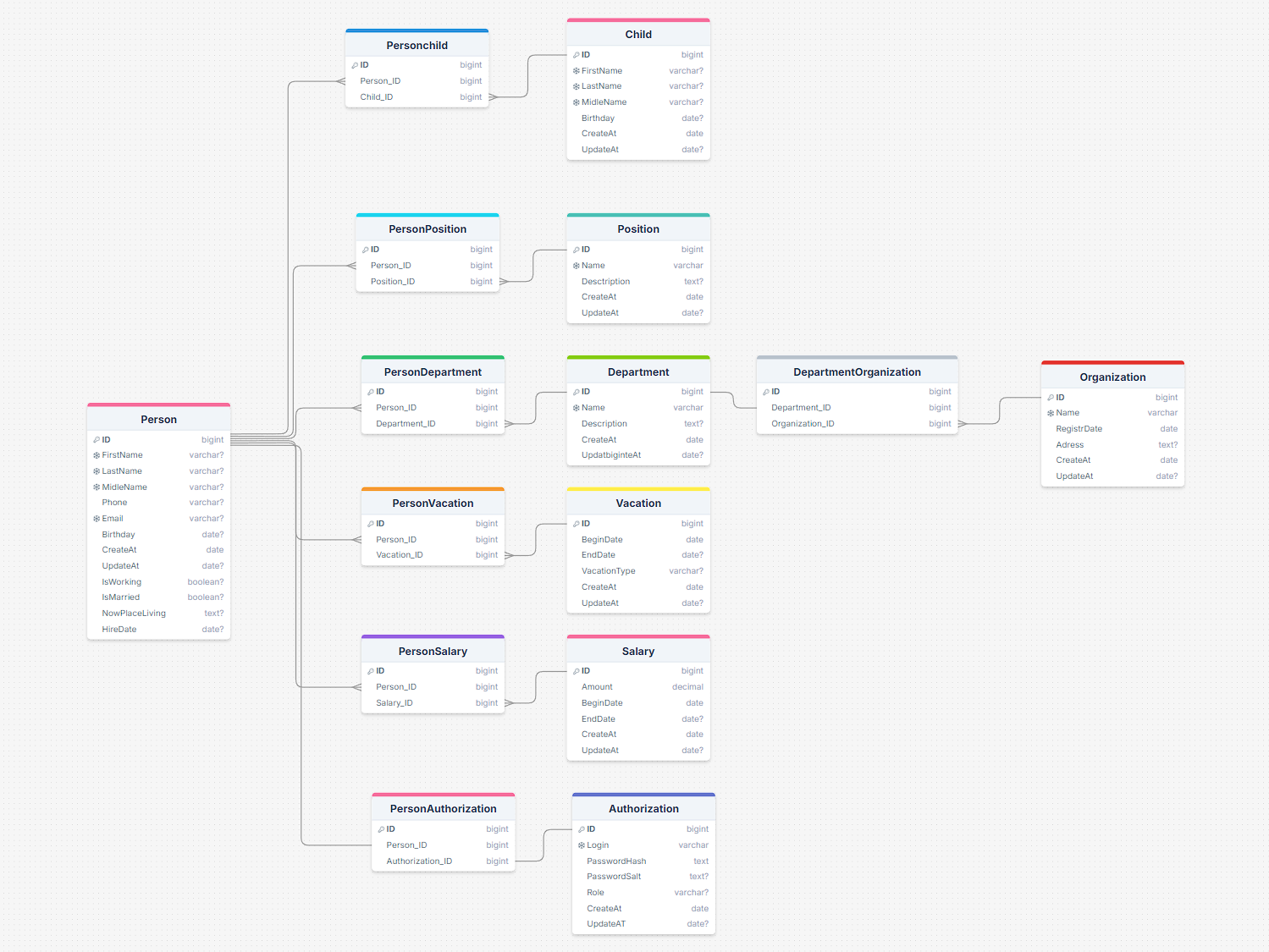


Рисунок 4.5 — Архитектура базы данных

1. **Тестирование и сопровождение программного обеспечения**

Тестирование проводилось вручную, по заранее составленным тестовым сценариям. Использовались также единичные модульные тесты для проверки функций валидации.

Виды применённых тестов:

* Функциональное тестирование интерфейсов.
* Тестирование обработки ошибок и исключений.
* Проверка связи клиента с сервером.

Подготовка тестовой среды

Среда:

* ОС: Windows 10 x64
* IDE: Visual Studio 2022
* СУБД: PostgreSQL
* Серверная часть: HR-Department.APIv2 (локально)
* Клиентская часть: HR-Department.DeskTop (WPF)

Тестовые данные:

* ФИО сотрудников
* Дата приёма/увольнения
* Отделы
* Должности

Таблицы ручного тестирования:

|  |  |
| --- | --- |
| **Название проекта** | HR-Department |
| **Версия сборки** | 1.0.0 |
| **Имя тестирующего** | Клюев Тимофей |
| **Дата теста** | 20.05.2025 |
| **Приоритет** | Высокий |
| **Название теста** | Добавление нового сотрудника |
| **Краткое описание** | Проверка корректности добавления сотрудника через клиентское приложение |
| **Этапы выполнения** | 1. Запустить клиентское приложение. 2. Перейти в раздел "Сотрудники". 3. Нажать "Добавить". 4. Заполнить все поля (ФИО, подразделение, дата). 5. Нажать "Сохранить". |
| **Тестовые данные** | ФИО: Петров Иван Сергеевич Подразделение: Отдел кадров Дата приёма: 01.01.2025 |
| **Ожидаемый результат** | Сотрудник появляется в таблице со всеми введёнными данными |
| **Фактический результат** | Сотрудник добавлен, данные отображаются корректно |
| **Предварительное условие** | Должны быть заполнены справочники "Подразделения" и "Должности" |
| **Постусловие** | Запись о сотруднике сохранена в базе данных |
| **Статус** |  |
| **Комментарии** | - |

|  |  |
| --- | --- |
| **Название проекта** | HR-Department |
| **Версия сборки** | 1.0.0 |
| **Имя тестирующего** | Клюев Тимофей |
| **Дата теста** | 20.05.2025 |
| **Приоритет** | Средний |
| **Название теста** | Обработка пустых полей при создании приказа |
| **Краткое описание** | Проверка поведения системы при попытке сгенерировать приказ без причины увольнения |
| **Этапы выполнения** | 1. Открыть форму "Увольнение". 2. Выбрать сотрудника. 3. Не указывать причину увольнения. 4. Нажать "Сформировать приказ". |
| **Тестовые данные** | Сотрудник: Иванова Мария Причина: — (не указана) |
| **Ожидаемый результат** | Отображается сообщение: «Не заполнены обязательные поля» |
| **Фактический результат** | Сообщение об ошибке отображается |
| **Предварительное условие** | Сотрудник должен быть добавлен ранее |
| **Постусловие** | Приказ не создается, приложение не падает |
| **Статус** |  |
| **Комментарии** | Желательно сделать обязательное поле подсвеченным |

Помимо ручного функционального тестирования клиентской части, в проекте реализованы модульные тесты, предназначенный для проверки логики работы одного из компонентов серверной части (API).

Модульное тестирование проводится с использованием фреймворка xUnit. Такой подход позволяет локально проверять работу отдельных методов без необходимости запускать всю систему.

Следующий модульный тест проверяет работоспособность серверного модуля — интерфейс взаимодействия системы с БД. Данный тест имитирует реальную базу данных что помогает использовать абсолютно любую SQL БД для тестирования системы. Тест добовляет нового сотрудника в базу, а затем проверяет десвительно ли данные прошли все этапы валидации и успешно сохранились в БД путём выборки сотрудника по его индикационному номеру.



Рисунок 5.1 ­­— Модульный тест

**Заключение**

В ходе прохождения учебной практики по модулю УП.02.01 «Осуществление интеграции программных модулей» я закрепил полученные ранее знания в области программирования, проектирования, тестирования и сопровождения программных продуктов. Практика проходила на базе учебного проекта «HR-Department», направленного на автоматизацию процессов управления персоналом в организации.

В процессе работы:

* Я изучил архитектуру клиент-серверного приложения, реализованного на основе ASP.NET Core и WPF.
* Ознакомился с особенностями построения REST API и реализацией бизнес-логики на серверной стороне.
* Разработал и протестировал функционал в клиентской части, включая формы добавления сотрудников, генерации приказов и авторизации пользователей.
* Сформировал ручные тест-кейсы и провёл их выполнение, что позволило выявить и устранить ряд ошибок в логике и интерфейсе.
* Создал модульный тест для проверки работы серверной логики.
* Работал с системой контроля версий Git и научился вести командную разработку через GitHub.

Полученные в ходе практики знания и навыки являются важным шагом в профессиональной подготовке и пригодятся мне при дальнейшем обучении и трудоустройстве. Работа с реальным кодом, участие в проекте и тестирование системы помогли лучше понять жизненный цикл программного обеспечения и требования к его качеству.

**Список использованных источников**

1. Windows Presentation Foundation documentation // Mircosoft URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/ (дата обращения: 20.05.2025).
2. Windows Presentation Foundation Overview // Mircosoft URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/ (дата обращения: 20.05.2025).
3. WPF Walkthroughs // Mircosoft URL: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/get-started/wpf-walkthroughs (дата обращения: 20.05.2025).
4. Base question of WPF // Stackoverflow URL: https://stackoverflow.com/questions/ (дата обращения: 20.05.2025).

**Приложения**