Министерство образования Тульской области

ГОСУДАРСТВЕННОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ТУЛЬСКОЙ ОБЛАСТИ

«ТУЛЬСКИЙ КОЛЛЕДЖ СТРОИТЕЛЬСТВА И ОТРАСЛЕВЫХ ТЕХНОЛОГИЙ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

по МДК.05.02 «РАЗРАБОТКА КОДА ИС»

ПМ.05 «Проектирование и разработка ИС»

на тему:

Разработка и тестирование информационной системы «Отдел кадров»

Выполнил:

студент группы И 3-6

Короткова К.В.

Проверил:

преподаватель

Соцкова С.Н.

Тула 2024

**СОДЕРЖАНИЕ**

[Введение 3](#_Toc184856501)

[1 Исследование предметной области 5](#_Toc184856502)

[1.1 Анализ задачи 5](#_Toc184856503)

[1.2 Требования к проекту 5](#_Toc184856504)

[2 Проектная часть 7](#_Toc184856505)

[2.1 Средства проектирования 7](#_Toc184856506)

[2.3 Модификация диаграмм вариантов использования 8](#_Toc184856507)

[2.4 Моделирование иерархии классов 14](#_Toc184856508)

[3 Практическая часть 19](#_Toc184856509)

[3.1 Структура программного проекта 19](#_Toc184856510)

[3.2 Разработка базы данных в MS SQL Server 20](#_Toc184856511)

[3.3 Разработка пользовательского интерфейса 30](#_Toc184856512)

[3.4 Описание реализации программного продукта 34](#_Toc184856513)

[3.4.1 Разработка классов сущностей 34](#_Toc184856514)

[3.4.2 Разработка управляющих классов 37](#_Toc184856515)

[3.5 Тестирование ИС 52](#_Toc184856516)

[3.5.1. Создание тест-кейсов 52](#_Toc184856517)

[3.5.2 Модульный тест 60](#_Toc184856518)

[Заключение 69](#_Toc184856519)

[Список использованных источников 70](#_Toc184856520)

[Приложение А. Руководство пользователя 72](#_Toc184856521)

[Приложение Б. Листинг кода приложения 101](#_Toc184856522)

**ВВЕДЕНИЕ**

В настоящее время многие организации всё чаще прибегают к использованию информационных систем для оптимизации своих процессов. Однако в некоторых компаниях отделы кадров по-прежнему работают с бумажными документами, что снижает эффективность работы, увеличивает временные затраты и повышает риск возникновения ошибок.

Такое положение вещей приводит к потере ценного рабочего времени сотрудников, снижению производительности труда и может иметь негативные последствия для всей организации.

Для разработки данной системы использовались программы: MS Visio, Visual Studio, MS SQL Server.

Объект исследования – информационная система «Отдел кадров».

Предмет исследования – процесс создания ИС «Отдел кадров».

Основной целью исследования является разработка и тестирование приложения для отдела кадров, которая обеспечит выполнение следующих задач:

* Провести анализ предметной области и требований к информационной системе.
* Произвести модификацию диаграмм вариантов использования и диаграммы классов для системы.
* Определить структуру программного проекта и его архитектуру.
* Разработать базу данных, обеспечивающую хранение данных о сотрудниках, должностях и других кадровых показателях.
* Создать удобный пользовательский интерфейс, обеспечивающий интуитивное взаимодействие с системой.
* Разработать программный модуль, реализовав классы сущностей и управляющие классы информационной системы.
* Провести тестирование системы для выявления возможных ошибок и недостатков.

Структура работы соответствует логике исследования и включает в себя:

1 Введение – постановка цели и задач исследования, обоснование актуальности темы.

2 Исследование предметной области – анализ существующих решений. Выявление проблем и потребностей.

3 Проектная часть – разработка архитектуры системы. Выбор технологий и создание модели данных.

4 Практическая часть – программирование функциональности, создание интерфейса, базы данных и тестирование.

5 Заключение – подведение итогов работы, формулирование основных результатов и выводов.

6 Приложение А. Руководство пользователя – описание интерфейса системы, порядка её использования и эксплуатационной документации.

7 Приложение Б. Листинг кода приложения – содержит фрагменты исходного кода для иллюстрации ключевых моментов реализации проекта.

**1 Исследование предметной области**

**1.1 Анализ задачи**

Проектируемая информационная система «Отдел кадров» предназначена для автоматизации процессов учета сотрудников и управления кадровыми данными на предприятии. Анализ задачи заключается в исследовании текущих процессов, выполняемых вручную в отделе кадров, и выявлении тех аспектов, которые можно улучшить с помощью автоматизации. В настоящее время отдел кадров занимается следующими процессами:

* Ведение личных карточек сотрудников, включая данные о должности, заработной плате и стаже работы.
* Регистрация приема на работу, перевода на другую должность, изменения условий труда и увольнений.
* Назначение заработных плат, премий и других выплат, связанных с командировками и дополнительными показателями.
* Ведение трудовых книжек сотрудников в соответствии с законодательными требованиями.
* Формирование стандартных отчетов, таких как список сотрудников, их распределение по отделам, наличие детей, сведения о военнообязанных.

Анализ задачи показывает необходимость автоматизации этих процессов, чтобы снизить вероятность ошибок, ускорить выполнение рутинных операций, обеспечить хранение и доступ к данным в удобной форме.

**1.2 Требования к проекту**

На основе анализа задачи были определены требования к разрабатываемой информационной системе. Система должна:

* Автоматизировать процесс ввода, обновления и хранения данных о сотрудниках, включая информацию о должностях, заработной плате, премиях и других показателях.
* Обеспечить функциональность для регистрации приема на работу, перевода на другую должность, изменения заработной платы и других кадровых процедур.
* Поддерживать ведение электронных трудовых книжек, включая запись о приеме, переводе, увольнении и изменениях заработной платы.
* Формировать стандартные отчеты, такие как: список сотрудников с указанием отдела и занимаемой должности, список сотрудников с детьми, список военнообязанных сотрудников.
* Использовать реляционную базу данных для хранения информации, обеспечивая целостность данных и их быструю обработку.
* Иметь удобный пользовательский интерфейс, который позволит кадровому инспектору эффективно взаимодействовать с системой без необходимости длительного обучения.
* Поддерживать многоуровневую систему доступа для защиты конфиденциальной информации о сотрудниках.
* Тестироваться на предмет функциональности, удобства использования и производительности для обеспечения стабильной и корректной работы системы.

Эти требования должны быть реализованы с использованием современных средств разработки баз данных, таких как MS SQL Server, и ориентированы на обеспечение быстрого и точного доступа к данным, необходимых для эффективного функционирования отдела кадров.

**2 Проектная часть**

**2.1 Средства проектирования**

Для проектирования информационной системы «Отдел кадров» использовался объектно-ориентированный подход, который обеспечивает возможность создания модульной и гибкой архитектуры, легко поддающейся изменениям и масштабированию. Для моделирования структуры и поведения системы были применены средства графического проектирования, такие как Microsoft Visio и UML (Unified Modeling Language) — это унифицированный язык моделирования, широко используемый для визуализации, спецификации, проектирования и документирования систем программного обеспечения. Предоставляет набор диаграмм для описания структуры и поведения системы. Он используется на разных этапах разработки, начиная от анализа требований и заканчивая проектированием и внедрением программных решений. Эти инструменты позволили разработать диаграммы вариантов использования, диаграммы классов, которые четко отражают взаимодействие между элементами системы и их функциональность.

Средства проектирования позволяют наглядно представить систему, определить ее функциональные элементы и взаимосвязи между ними, что является важным этапом при создании информационных систем. Применение UML-диаграмм помогает описывать структуру системы, взаимодействие пользователей с ней и внутренние процессы, обеспечивая более качественное и понятное проектирование.

**2.2 Абстрагирование и выделение объектов**

Абстрагирование в объектно-ориентированном проектировании заключается в выделении ключевых сущностей предметной области и их представлении в виде объектов с атрибутами и методами. В контексте системы «Отдел кадров» такими объектами являются сотрудники, их трудовые книжки, приказы о приеме на работу, данные о зарплате и другие кадровые документы. Основные объекты:

Сотрудник – ключевая сущность, которая содержит атрибуты, такие как имя, фамилия, паспортные данные, адрес и др.

Трудовая книжка – хранит записи о приеме на работу, переводе, увольнении и других событиях, связанных с трудовой деятельностью.

Приказ – объект, связанный с созданием приказов о приеме, увольнении, переводе сотрудников, начислении зарплат.

Таким образом, выделение объектов и их атрибутов позволяет структурировать информацию и обеспечить корректную работу системы с кадровыми данными.

**2.3 Модификация диаграмм вариантов использования**

Диаграмма вариантов использования представляет функциональные требования и взаимодействия между участниками системы в виде визуальных элементов. Чтобы выяснить, каким образом ИС «Отдел кадров» может максимально эффективно выполнять свою работу, нужно изучить различные сценарии его использования.

Актер в диаграмме вариантов использования – это внешняя по отношению к системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения своих задач.

Вариант использования – это описание последовательности действий, которые выполняет система для достижения конкретного результата для актера.

Связь между вариантами использования — это отношения между различными вариантами использования, которые определяют зависимости и взаимодействия между ними. Существует несколько типов связей, включая ассоциации, обобщения, расширения и включения. Используется два типа связей: ассоциация и включение. Ассоциации — это базовые отношения между двумя вариантами использования, которые указывают на взаимодействие между ними. Включение — это более специализированная связь, которая используется для описания дополнительных функциональных возможностей, связанных с основным вариантом использования.

Для визуального представления требований к информационной системе «Отдел кадров» была разработана диаграмма вариантов использования, представленная на рисунке 1.

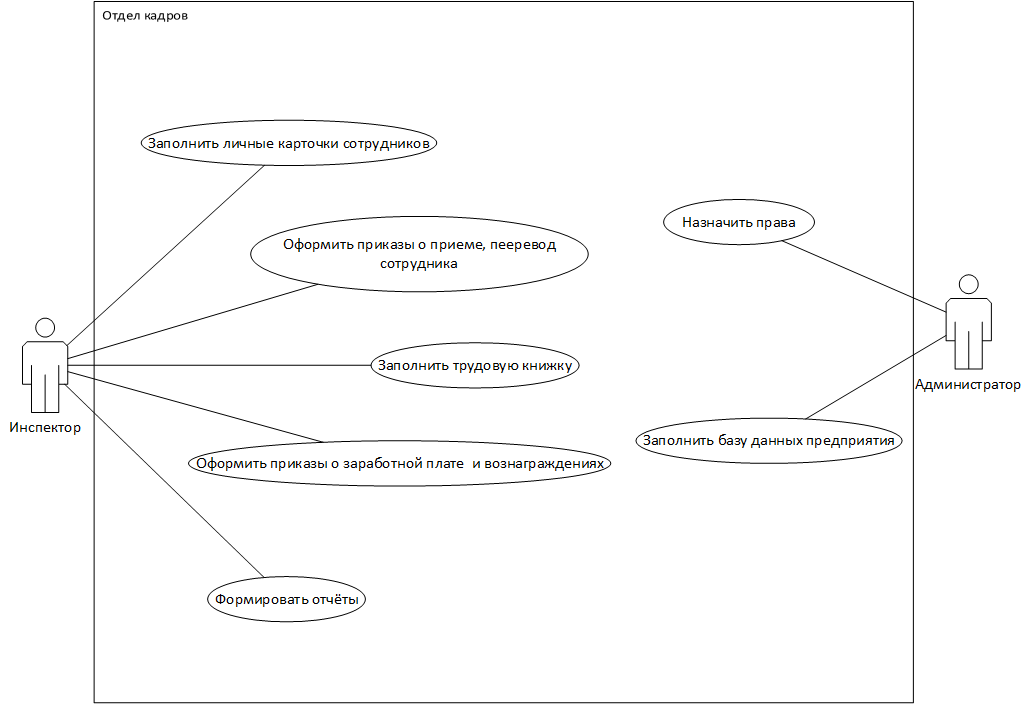


Рисунок 1 – Диаграмма вариантов использования

Для проектируемой системы в диаграмме вариантов использования определены два актера – Инспектор и Администратор, краткое описание которых представлено в таблице 1.

Таблица 1 – Актёры.

|  |  |
| --- | --- |
| **Актёры** | **Краткое описание** |
| Инспектор | Сотрудник, который занимается ведением документации, связанной с персоналом организации. |
| Администратор | Специалист, который отвечает за настройку и управление работой компьютерной системы предприятия. |

Для каждого актера в диаграмме определены варианты использования. Перечень прецедентов и их описание приведены в таблице 2.

Таблица 2 – Варианты использования.

|  |  |
| --- | --- |
| **Прецедент** | **Краткое описание** |
| Заполнять личные карточки сотрудника. | Запускается Инспектором. Позволяет вносить информацию о его имени, должности, стаже работы и других данных. |
| Оформлять приказы о приёме, перевод сотрудника. | Запускается Инспектором. Позволяет создать необходимые документы на основе данных о сотруднике и его должности. |

Продолжение таблицы 2.

|  |  |
| --- | --- |
| Заполнять трудовую книжку. | Запускается Инспектором. Позволяет выполнять различные действия, связанные с внесением информации о работе и стаже сотрудника. |
| Оформлять приказы о заработной плате и вознаграждениях. | Запускается Инспектором. Позволяет рассчитать и оформить документы на основе информации о сотруднике и его стаже работы. |
| Формировать отчёты . | Запускается Инспектором. Позволяет создавать отчеты о работе сотрудников, текучести кадров и других показателях. |
| Назначать права. | Запускается Администратором. Позволяет настроить права для каждого пользователя системы. |
| Заполнять базу данных предприятия. | Запускается Администратором. Позволяет добавлять и удалять информацию о предприятии и его сотрудниках. |

Для более глубокого анализа сценария варианта использования «Заполнить личные карточки сотрудников» была создана детализирующая диаграмма вариантов использования, которую можно увидеть на рисунке 2.

Описание прецедентов, связанных с заполнением личной карточки сотрудника, представлено в таблице 3.

Таблица 3 – Вариант использования «Заполнить личные карточки сотрудников».

|  |  |
| --- | --- |
| **Прецеденты** | **Краткое описание** |
| Заполнить данные сотрудника. | Запускается Инспектором. Позволяет вводить информацию о сотрудниках, такую как имя, дата рождения, гражданство, адрес и другие данные, необходимые для работы в организации. |
| Изменить данные. | Запускается Инспектором. Позволяет изменять информацию в системе. |
| Сохранить данные. | Запускается Инспектором. Позволяет сохранять данные, которые были внесены или изменены. |

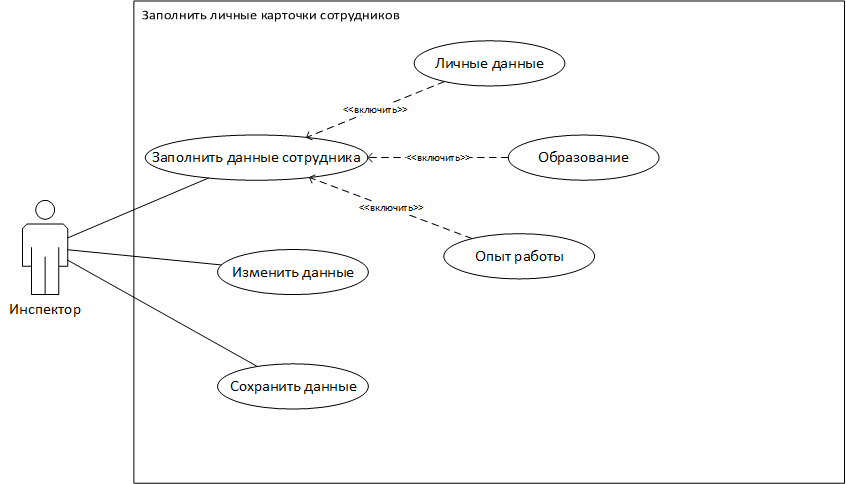


Рисунок 2 – Диаграмма вариантов использования «Заполнить личные карточки сотрудников»

Текстовое описание пошагового выполнения вариантов использования представлена в потоке событий. Спецификация потоков события варианта использования «Заполнить личные карточки сотрудников» представлена в таблице 4.

Таблица 4 – Спецификация потока события для варианта использования «Заполнить личные карточки сотрудников».

|  |  |
| --- | --- |
| **Действия актёров** | **Отклик системы** |
| Основной поток событий | |
| 1. Вариант использования начинается, когда Инспектор входит в систему. Для этого он вводит логин и пароль.   **Исключение № 1** Данные введены неверно. | 1. Открывает главную форму. |
| 1. Инспектор выбирает опцию меню «Сотрудники». | 1. Открывает форму «Сотрудники». |
| 1. Выбор опции «Добавить сотрудника». | 1. Открывает форму «Добавить сотрудника в компанию». |
| 1. Вводит личные данные.   **Исключение № 2** Данные заполнены не во всех полях. |  |
| 1. Пользователь производит сохранение.   **Исключение № 4** Пользователь не подтверждает сохранение. | 1. Система запрашивает разрешение на сохранение. |
| **Исключение № 1** Данные введены неверно. | |
|  | 1. Выводит сообщение «Неверный логин или пароль». Возврат на форму для повторного введения логина и пароля. |
| 1. Инспектор вводит логин и пароль заново. |  |

Продолжение таблицы 4.

|  |  |
| --- | --- |
| **Исключение № 2** Данные заполнены не во всех полях. | |
|  | 1. Выводит сообщение «Заполните все обязательные поля». Возврат на форму для ввода данных о сотруднике. |
| 1. Ввод данных. |  |
| **Исключение № 3** Пользователь не подтверждает сохранение. | |
|  | 1. Открывает форму для ввода данных. |

Этапы выполнения вариантов использования описаны в виде действий. Спецификация потока событий для варианта использования «Изменить данные» представлено в таблице 5.

Таблица 5 – Спецификация потока события для варианта использования «Изменить данные».

|  |  |
| --- | --- |
| **Действия актёров** | **Отклик системы** |
| Основной поток событий | |
| 1. Вариант использования начинается, когда Инспектор входит в систему. Для этого он вводит логин и пароль.   **Исключение № 1** Данные введены неверно. | 1. Открывает главную форму. |
| 1. Выбирает опцию меню «Сотрудники». | 1. Открывает форму «Сотрудники». |
| 1. Пользователь производит поиск и выбор сотрудника. | 1. Открывает форму «Данные сотрудника компании». |
| 1. Инспектор выбирает опцию «Редактировать». Вводит новые данные. Выбор опции «Завершить редактирование». | 1. Выводит сообщение «Данные сотрудника успешно отредактированы». |
| **Исключение № 1** Данные введены неверно. | |
|  | 1. Выводит сообщение «Неверный логин или пароль». Возврат на форму для повторного введения логина и пароля. |
| 1. Инспектор вводит логин и пароль заново. |  |

Также был детально разобран вариант использования «Заполнить трудовую книжку». В ходе анализа была создана соответствующая диаграмма, представленная на рисунке 3.



Рисунок 3 – Диаграмма вариантов использования «Заполнить трудовую книжку»

Вариант использования, представляющий заполнение трудовой книжки, представлен в таблице 6.

Таблица 6 – Вариант использования «Заполнить трудовую книжку».

|  |  |
| --- | --- |
| **Прецеденты** | **Краткое описание** |
| Добавить запись. | Запускается Инспектором. Позволяет добавлять записи о работнике. |

Текстовое описание потока события, в котором указаны шаги выполнения варианта использования. Спецификация потока события для варианта использования «Заполнить трудовую книжку» в таблице 7.

Таблица 7 – Спецификация потока события для варианта использования «Заполнить трудовую книжку».

|  |  |
| --- | --- |
| **Действия актёров** | **Отклик системы** |
| Основной поток событий | |
| 1. Вариант использования начинается, когда Инспектор начинает входить в систему. Для этого он вводит логин и пароль.   **Исключение № 1** Данные введены неверно. | 1. Открывает главную форму. |
| 1. Инспектор выбирает опцию меню «Трудовые книжки». | 1. Открывает форму «Трудовые книжки сотрудников». |
| 1. Инспектор производит поиск и выбирает сотрудника. | 1. Система открывает форму «Трудовая книжка сотрудника». |
| 1. Инспектор выбирает опцию «Добавить запись». |  |
| 1. Вводит данные. Производит сохранение.   **Исключение № 2** Данные заполнены не во всех полях.  **Исключение № 3** Пользователь не подтверждает сохранение. | 1. Система запрашивает разрешение на сохранение. |

Продолжение таблицы 7.

|  |  |
| --- | --- |
| **Исключение № 1** Данные введены неверно. | |
|  | 1. Выводит сообщение «Неверный логин или пароль». Возврат на форму для повторного введения логина и пароля. |
| 1. Инспектор вводит логин и пароль заново. |  |
| **Исключение № 2** Данные заполнены не во всех полях. | |
|  | 1. Выводит сообщение «Заполните все обязательные поля». Возврат на форму для ввода данных о сотруднике. |
| 1. Ввод данных. |  |
| **Исключение № 3** Пользователь не подтверждает сохранение. | |
|  | 1. Открывает форму для ввода данных. |

**2.4 Моделирование иерархии классов**

Диаграмма классов является одним из основных инструментов объектно-ориентированного анализа и проектирования. Она позволяет визуализировать структуру системы, показывая классы, их атрибуты и методы, а также взаимосвязи между ними. Одним из ключевых подразделений в любой организации является отдел кадров, поэтому анализ этого отдела с помощью диаграммы классов поможет улучшить его работу и повысить эффективность.

Диаграмма классов может содержать различные типы связей между классами, такие как ассоциация, агрегирование и композиция. Используется такие типы связей как:

* Ассоциация *–* связь между классами, которая обозначает взаимодействие между объектами этих классов.
* Зависимость *–* связь между двумя классами, при которой один класс использует другой класс для выполнения своих операций.
* Расширение *–* связь между двумя классами, которая используется для обозначения того, что один класс расширяет функциональность другого класса, добавляя новые свойства и методы.
* Агрегация *–* связь “часть-целое”, при которой один класс представляет собой часть другого класса.

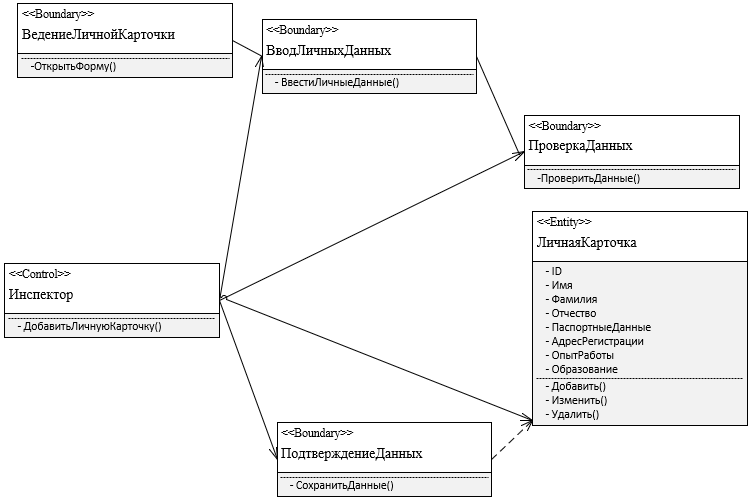
Основными классами являются:

Control *–* элемент, который управляет выполнением программы. Это может быть метод, функция или оператор.

Boundary *–* элемент, определяющий границу между системой и внешним миром. Он может включать входные и выходные данные, а также интерфейсы с другими системами.

Entity *–* основной элемент предметной области. Это может быть объект, класс или сущность, которые имеют свойства и поведение.

Диаграмма классов для варианта использования «Заполнить личные карточки сотрудников» показывает классы, которые содержат атрибуты и методы, а также отношения между ними, для работы с личной картой сотрудника. Всё это представлено на рисунке 4.

 Рисунок 4 – Диаграмма классов для варианта использования «Заполнить личные карточки сотрудников»

Подробное описание для сущности (entity) ЛичнаяКарточка представлена в таблице 8. Она показывает описание класса, его атрибуты и методы, которые могут быть выполнены над объектом.

Таблица 8 – Сущность ЛичнаяКарточка.

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой информацию, которая содержится в личной карте сотрудника. |
| Атрибуты | ID: int – код сотрудника.  Имя: varchar – имя сотрудника.  Фамилия: varchar *–* фамилия сотрудника.  Отчество: varchar – отчество сотрудника.  ПаспортныДанные: int – паспортные данные сотрудника.  АдресРегистрации: varchar – адрес регистрации сотрудника.  ОпытРаботы: varchar – опыт работы сотрудника.  Образование: varchar – образование сотрудника. |

Продолжение таблицы 8.

|  |  |
| --- | --- |
| Операции | Добавить() – добавить данные о сотруднике.  Изменить() – изменить данные о сотруднике.  Сохранить() – сохранить данные о сотруднике. |

Для работы с трудовой книжкой была составлена диаграмма классов для варианта использования «Заполнить трудовую книжку», в которой показаны классы, включающие атрибуты и методы, которые совершаются над сущностями. Это представлено на рисунке 5.

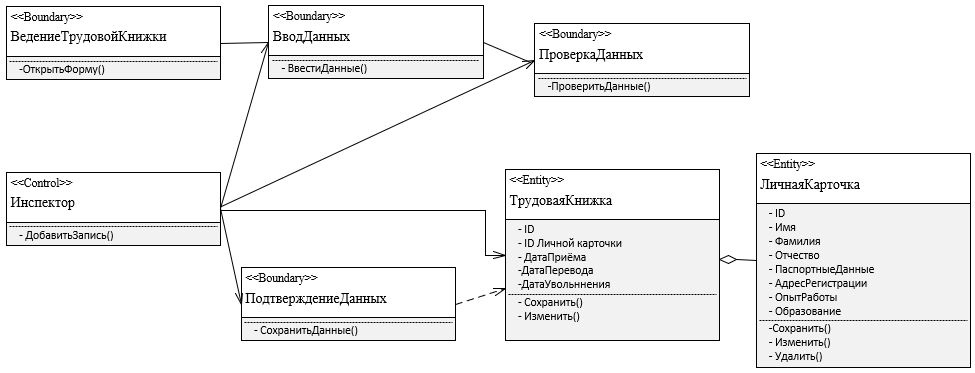


Рисунок 5 – Диаграмма классов для варианта использования «Заполнить трудовую книжку»

Сущность «ТрудоваяКнижка» детально описана в таблице 9. Это описание включает в себя сведения о сущности, а также набор её атрибутов и методов, с помощью которых можно с ней работать.

Таблица 9 – Сущность ТрудоваяКнижка.

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| Комментарий | Класс, представляющий собой информацию, которая содержится в трудовой книжке сотрудника. |
| Атрибуты | ID: int – код сотрудника.  IDЛичнойКарточки:int – номер личной карточки сотрудника.  ДатаПриёма: date – дата приёма сотрудника на работу.  ДатаПеревода: date – дата перевода на другую должность.  ДатаУвольнения:date – дата увольнения сотрудников. |
| Операции | Изменить() – изменить данные о сотруднике.  Сохранить() – сохранить данные о сотруднике. |

3 Практическая часть

3.1 Структура программного проекта

Структура программного проекта информационной системы «Отдел кадров» была построена с использованием объектно-ориентированного подхода. Проект включает несколько ключевых модулей, которые обеспечивают выполнение всех необходимых функций системы, таких как ведение личных карточек сотрудников, трудовых книжек и формирование отчетов. Основные компоненты структуры проекта:

* Пользовательский интерфейс (UI):

Этот компонент отвечает за взаимодействие пользователя с системой. Он реализован с использованием форм для ввода данных о сотрудниках, создания приказов, ведения трудовых книжек и генерации отчетов. Пользовательский интерфейс обеспечивает удобный доступ к функционалу системы, включая проверку правильности ввода данных и отображение сообщений об ошибках.

В интерфейсе предусмотрены выпадающие списки для выбора отделов и должностей, а также возможность добавления и редактирования записей о сотрудниках и приказах.

* Логика приложения:

Логика системы реализована через управляющие классы, которые обрабатывают действия пользователей и обеспечивают выполнение операций по созданию, редактированию и удалению данных в системе. Эти классы взаимодействуют с базой данных для выполнения запросов на получение или изменение данных о сотрудниках, приказах и других кадровых данных.

Управляющие классы включают методы для выполнения ключевых операций, таких как добавление новых сотрудников, изменение их данных, создание приказов и формирование отчетов.

* База данных:

Система использует реляционную базу данных на основе MS SQL Server, в которой хранится вся информация о сотрудниках, их трудовой деятельности, приказах, заработной плате и других кадровых данных. База данных включает таблицы для хранения данных о сотрудниках, их трудовых книжках, отделах, должностях и зарплатах.

Связи между таблицами обеспечивают целостность данных и позволяют эффективно выполнять запросы, такие как поиск сотрудников по отделам или формирование отчетов по заработной плате.

* Модуль отчетности:

Этот компонент отвечает за формирование отчетов по запросам пользователя. Он позволяет создавать различные типы отчетов, такие как списки сотрудников, отчеты по отделам, информация о сотрудниках с детьми или военнообязанных.

Формирование отчетов основано на данных, хранящихся в базе данных, с использованием встроенных функций для фильтрации и сортировки информации по заданным критериям.

* Классы сущностей:

Основными классами проекта являются сущности, которые моделируют объекты реального мира, такие как сотрудники, трудовые книжки и приказы. Эти классы содержат атрибуты и методы, позволяющие взаимодействовать с данными, например, добавлять или изменять информацию о сотрудниках.

Ключевые классы: сотрудник (атрибуты: ФИО, дата рождения, должность, отдел, зарплата), приказ (атрибуты: тип приказа, дата создания, описание), трудовая книжка (атрибуты: дата приема на работу, дата перевода, дата увольнения).

* Модуль безопасности:

Для защиты конфиденциальных данных сотрудников система использует механизм разграничения прав доступа, который контролирует доступ пользователей к различным функциям и данным системы. Разные роли (например, инспектор и администратор) имеют доступ только к тем функциям, которые соответствуют их полномочиям.

Таким образом, структура проекта была спроектирована с целью обеспечения гибкости, масштабируемости и безопасности, что позволяет системе эффективно справляться с автоматизацией кадровых процессов на предприятии.

3.2 Разработка базы данных в MS SQL Server

Для проектирования базы данных информационной системы «Отдел кадров» были разработаны как логическая, так и физическая модели данных с использованием Microsoft Visio.

Логическая модель была создана для абстрактного представления структуры данных, включая сущности (сотрудники, трудовые книжки, приказы и т.д.), их атрибуты и взаимосвязи. Логическая модель позволяет наглядно отобразить структуру базы данных и связи между таблицами, не привязываясь к конкретной системе управления базами данных (СУБД). Это облегчает процесс проектирования и согласования модели перед ее физической реализацией.

Методология IDEF1X (Integrated Definition for Information Modeling) — это стандартизированный подход к проектированию логической модели данных. С помощью IDEF1X моделируются сущности, их атрибуты и связи, что обеспечивает четкое и консистентное представление структуры данных. Этот подход использует нотации, упрощающие понимание и визуализацию логической модели, что облегчает ее дальнейшее использование при разработке баз данных.

С использованием IDEF1X в отделе кадров можно создать логическую модель, включающую такие сущности, как сотрудники, запись в трудовой книжке, отделы, должности и другие, представленное на рисунке 6. Каждая сущность будет иметь свои атрибуты, которые отражают характеристики и свойства персонала и процессов управления кадрами.

Связи между сущностями отражают взаимодействие и зависимости между различными элементами системы управления персоналом.

Таким образом, методология IDEF1X обеспечивает систематизированный подход к моделированию данных, что позволяет эффективно управлять информацией в ИС «Отдел кадров» и обеспечить эффективное управление персоналом в организации.

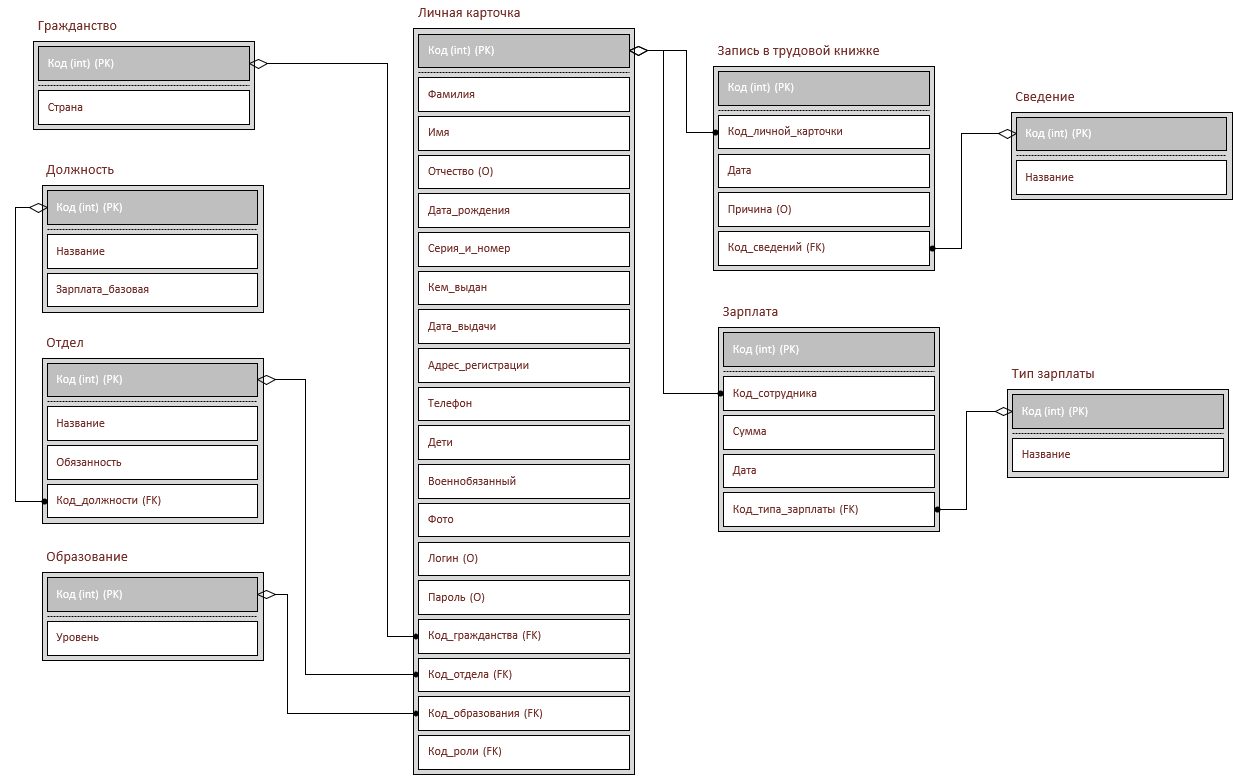
****

Рисунок 6 – Логическая модель диаграмма IDEF1X

Физическая модель данных (ФМД) – это детальное описание структуры данных, хранимых в базе данных. Она определяет, как данные будут представлены на физическом носителе, какие средства будут использоваться для их хранения, и как данные будут организованы для обеспечения оптимальной производительности.

ФМД является низкоуровневым представлением данных, ориентированным на систему управления базами данных (СУБД). Она не зависит от бизнес-логики и может быть изменена без изменения логической модели данных.

ФМД создается на основе логической модели данных с учетом ограничений СУБД, требований к производительности и других факторов.

Диаграмма, представленная на рисунке 7, является физической моделью IDEF1X для ИС «Отдел кадров». Она иллюстрирует структуру данных, хранимых в базе данных, с учетом:

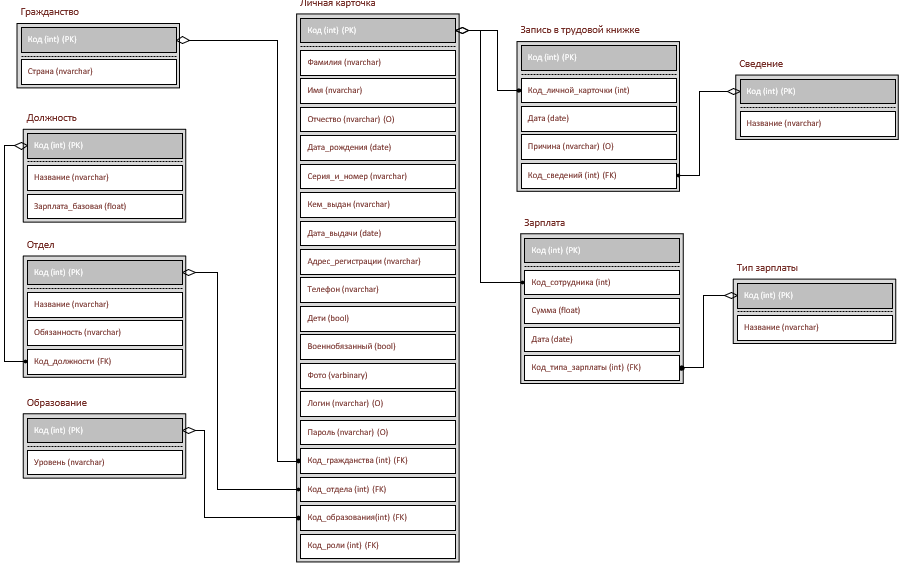


Рисунок 7 – Физическая модель диаграмма IDEF1X

Использование Microsoft Visio позволило визуализировать обе модели и обеспечить ясное представление структуры данных для всех участников проекта.

Словарь баз данных (или метаданные) — это система, которая хранит информацию о структуре, свойствах и взаимосвязях данных в базе данных. Он служит важным инструментом для управления данными и облегчает их использование. Основные аспекты словаря баз данных включают:

* Структура данных: Описание таблиц, их полей и типов данных. Каждая таблица имеет уникальное имя и определенные поля, которые определяют тип хранимой информации.
* Типы данных: Указание формата данных, таких как целые числа, строки, даты и т.д. Это помогает обеспечить целостность данных.
* Связи между таблицами: Определение отношений (например, один-ко-многим, многие-ко-многим) помогает понять, как данные взаимодействуют друг с другом.
* Ограничения: Правила, которые обеспечивают корректность данных (например, уникальность значений, обязательные поля).
* Индексы и представления: Информация о создании индексов для ускорения доступа к данным и представлениях для упрощения работы с часто используемыми запросами.

Словарь баз данных упрощает управление, документирование и поддержку базы данных, обеспечивая ясность для разработчиков и пользователей. Он также помогает в процессе проектирования, улучшая качество и эффективность хранения данных.

Структура базы данных для таблицы «Citizenship», содержащая информацию, связанную с гражданством, которая представлена в таблице 10.

Таблица 10 – Словарь базы данных для таблицы Citizenship.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ гражданства | int |  | NOT NULL |
| Country | Наименование страны | Nvarchar | 50 | NOT NULL |

Структура базы данных для таблицы «Department», содержащая информацию, связанную с отделом, которая представлена в таблице 11.

Таблица 11 – Словарь базы данных для таблицы Department.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ отдела | int |  | NOT NULL |
| Title | Название отдела | nvarchar | 50 | NOT NULL |

Продолжение таблицы 11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Responsibiltities | Обязанности отдела | nvarchar | 50 | NOT NULL |

Структура базы данных для таблицы «Post», содержащая информацию, связанную с должностью, которая представлена в таблице 12.

Таблица 12 – Словарь базы данных для таблицы Post.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ должности | int |  | NOT NULL |
| Title | Наименование должности | Nvarchar | 50 | NOT NULL |
| Base\_salary | Оклад | Float |  | Нецелое,> 0 |
| Id\_department | Внешний код должности | int |  | NOT NULL |

Структура базы данных для таблицы «Education», содержащая информацию, связанную с образованием, которая представлена в таблице 13.

Таблица 13 – Словарь базы данных для таблицы Education.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ образования | int |  | NOT NULL |
| Levell | Уровень образования | Nvarchar | 50 | NOT NULL |

Структура базы данных для таблицы «Mixing», содержащая информацию, связанную с сведениями, которая представлена в таблице 14.

Таблица 14 – Словарь базы данных для таблицы Mixing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ сведений | int |  | NOT NULL |
| Title | Наименование сведений | Nvarchar | 50 | NOT NULL |

Структура базы данных для таблицы «Salary\_type», содержащая информацию, связанную с типом гражданства, которая представлена в таблице 15.

Таблица 15 – Словарь базы данных для таблицы Salary\_type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ типа зарплаты | int |  | NOT NULL |
| Title | Наименование типа зарплаты | Nvarchar | 50 | NOT NULL |

Структура базы данных для таблицы «Salary», содержащая информацию, связанную с зарплатой, которая представлена в таблице 16.

Таблица 16 – Словарь базы данных для таблицы Salary.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ зарплаты | Int |  | NOT NULL |
| Id\_personal\_card | Внешний ключ личной карточки | Int |  | NOT NULL |
| Amount | Сумма зарплаты | Float |  | Нецелое,> 0 |
| Date | Дата зарплаты | Date |  | NOT NULL |
| Id\_salary\_type | Внешний ключ типа ключа | Int |  | NOT NULL |

Структура базы данных для таблицы «Entry\_in\_the\_work\_book», содержащая информацию, связанную с записью в трудовой книжке, которая представлена в таблице 17.

Таблица 17 – Словарь базы данных для таблицы Entry\_in\_the\_work\_book.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ записи в трудовую книгу | int |  | NOT NULL |
| Id\_personal\_card | Внешний ключ личной карточки | int |  | NOT NULL |
| Date | Дата записи | date |  | NOT NULL |
| Reason | Причина записи | Nvarchar | 50 |  |
| Id\_mixing | Внешний ключ сведений | int |  | NOT NULL |

Структура базы данных для таблицы «Personal\_card», содержащая информацию, связанную с личной карточкой, которая представлена в таблице 18.

Таблица 18 – Словарь базы данных для таблицы Personal\_card.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя** | **Назначение** | **Тип** | **Длина** | **Ограничение** |
| ID | Первичный ключ личной карточки | Int |  | NOT NULL |
| Surname | Фамилия | Nvarchar | 50 | NOT NULL |
| Name | Имя | Nvarchar | 50 | NOT NULL |
| Patronymic | Отчество | Nvarchar | 50 |  |
| Date\_of\_birth | Дата рождения | Date |  | NOT NULL |
| Series\_and\_number | Серия и номер | Int |  | NOT NULL |
| Issued\_by\_whom | Кем выдан | Nvarchar | 150 | NOT NULL |
| Date\_of\_issue | Дата выдачи | Date |  | NOT NULL |
| Registration\_address | Адрес регистрации | Nvarchar | 255 | NOT NULL |
| Telephone | Номер телефона | Nvarchar | 15 | NOT NULL |
| Children | Дети | bool |  | True\False |
| Military\_service | Военнообязанный | bool |  | True\False |
| Photo | Фото | Nvarchar | 200 | NOT NULL |
| Login | Логин | Nvarchar | 50 | NOT NULL |
| Password | Пароль | Nvarchar | 50 | NOT NULL |
| Id\_citizenship | Внешний ключ гражданства | Int |  | NOT NULL |

Продолжение таблицы 18.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id\_post | Внешний ключ должности | Int |  | NOT NULL |
| Id\_education | Внешний ключ образования | Int |  | NOT NULL |

Для разработки базы данных информационной системы «Отдел кадров» была использована реляционная система управления базами данных MS SQL Server, которая обеспечивает надежное хранение данных и возможность их масштабирования. Проектирование базы данных включало определение сущностей, их атрибутов и связей между ними для представления кадровых процессов предприятия, которые представлены на рисунке 8.

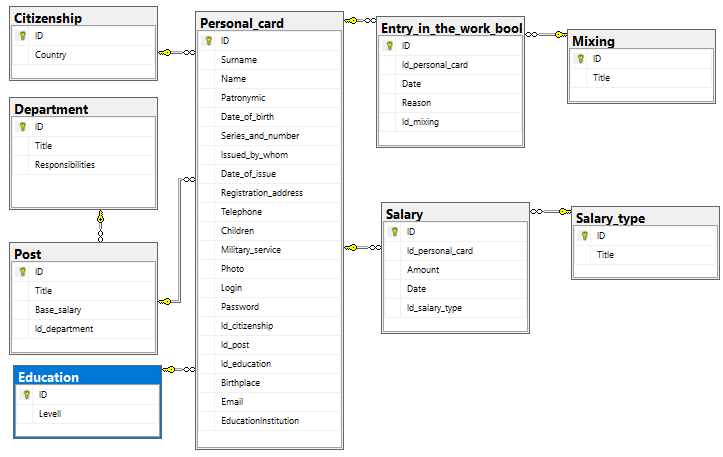


Рисунок 8 – Диаграмма базы данных

После создания базы данных, словаря данных и моделирования структуры с помощью IDEF1X была обеспечена чёткая организация данных и их взаимосвязей. Словарь данных уточнил значения полей, а модель IDEF1X гарантировала корректность структуры.

Эти этапы создали надёжную основу для эффективного хранения и обработки информации, что повысит производительность системы и упростит её развитие.

3.3 Разработка пользовательского интерфейса

Разработка пользовательского приложения – это процесс создания программного обеспечения, ориентированного на конечного пользователя, с целью обеспечения удобного и эффективного взаимодействия с системой. В этом процессе важно учесть множество факторов, таких как функциональность, удобство, производительность и безопасность. Основной акцент при разработке UI/UX должен быть сделан на создание интуитивно понятного интерфейса, который способствует повышению продуктивности пользователя и минимизирует количество ошибок.

Окно авторизации «Отдел кадров» предназначено для входа в информационную систему (рисунок 9). Оно включает в себя элементы для ввода логина и пароля, а также кнопку для авторизации пользователя.

Элементы окна:

Label — элемент интерфейса для отображения текста заголовков и инструкций, например, «Отдел кадров», «Добро пожаловать», «Вход в систему», «Логин» и «Пароль».

TextBox — поле ввода текста, позволяющее пользователю вводить данные логина.

PasswordBox — поле ввода для паролей, скрывающее введённые символы.

Button — кнопка, которая инициирует действие при нажатии. «Войти», которая выполняет вход в систему.

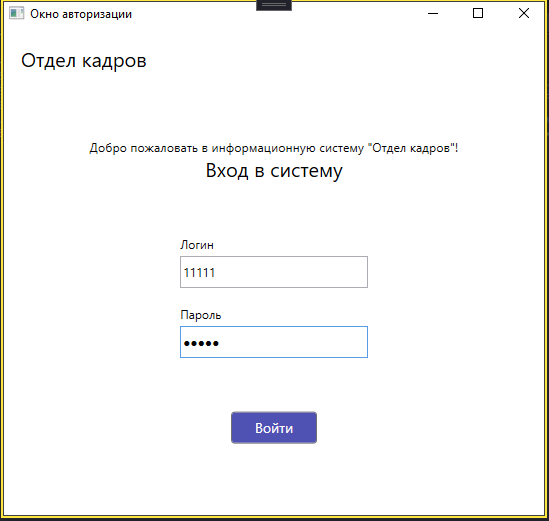


Рисунок 9 – Окно авторизации

Форма «Сотрудники» (рисунок 10) для инспектора предназначена для управления информацией о сотрудниках: поиска сотрудников, отображения их личных данных, добавления нового сотрудника, назначения наград и изменения данных.

Button – «Отдел кадров», «Сотрудники», «Трудовые книжки», «Приказы», «Отчеты» управляют переходом между различными разделами системы. «Добавить сотрудника» запускает команду добавления нового сотрудника. «Выйти», которая привязана к команде выхода из системы. «Найти» для выполнения поиска. «Назначить награду» для назначения награды выбранному сотруднику.

Textbox – для ввода текста поиска по сотрудникам.

**ListView –** контейнер для отображения данных в виде списка с возможностью добавления, редактирования и удаления элементов. Для отображения списка сотрудников с полями: имя, должность, отдел, телефон, дата рождения. При одном нажатии на сотрудника отображаются данные: имя, фамилия, отчество, дата рождения, должность, отдел, а также фото, а при двойном щелчке происходит его выбор и открытие формы «Изменение сотрудника».

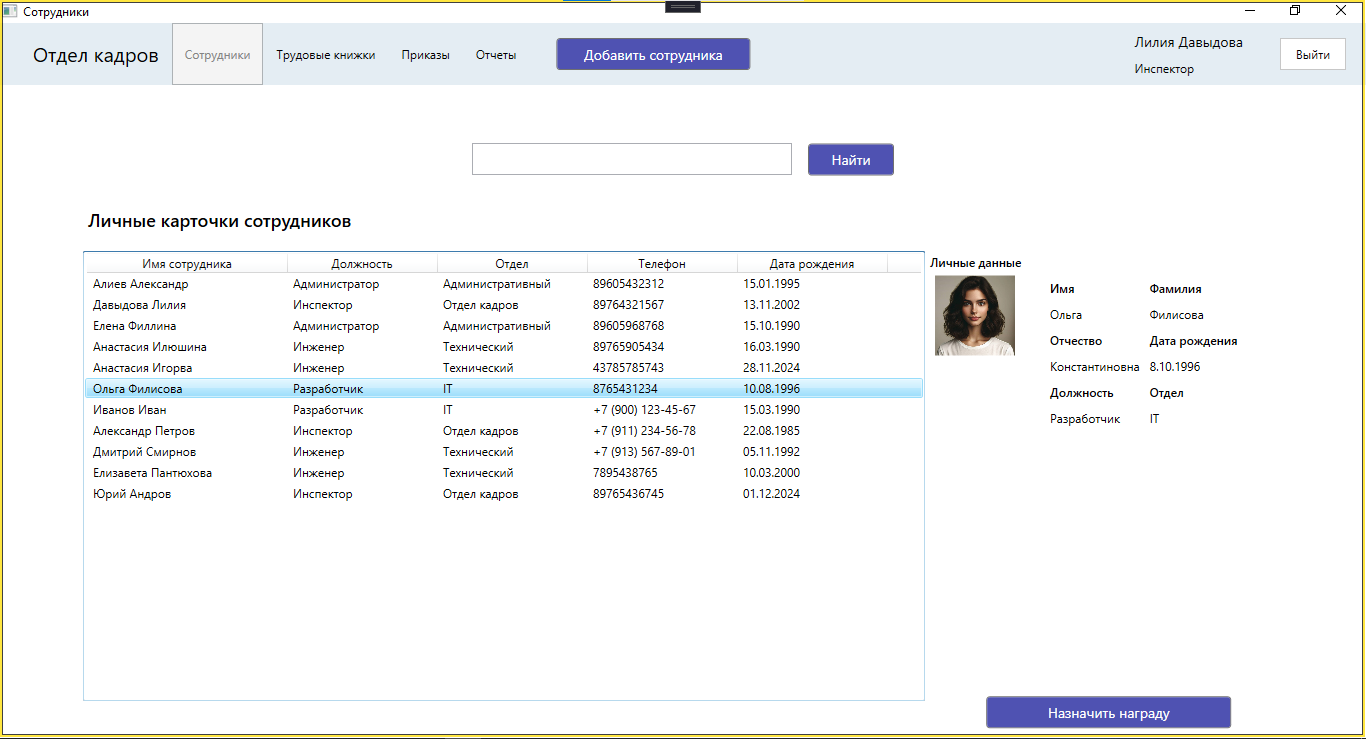


Рисунок 10 – Форма «Сотрудники»

На форме «Добавление сотрудника» (рисунок 11) для инспектора присутствуют элементы интерфейса, такие как кнопки: «Отдел кадров», «Сотрудники», «Трудовые книжки», «Приказы», «Отчеты», которые управляют переходом между различными разделами системы. Кнопка «Отмена» отменяет текущие изменения и возвращает на предыдущий экран, а кнопка «Создать приказ» инициирует создание приказа для выбранного сотрудника. Кнопка «Выйти» привязана к команде выхода из системы, а также есть кнопка, которая предназначена для загрузки аватара сотрудника.

Отображены label, такие как «Имя», «Фамилия», «Должность», «Личные данные», «Контактные данные», «Трудовые данные», «Паспортные данные», «Образование», «Гражданство», «Прочие данные», «Есть дети?», «Военнообязанный?», которые обозначают различные категории данных, которые необходимо ввести.

Textbox используются для ввода информации о сотруднике: имя, фамилия, отчество, дата рождения, место рождения, адрес регистрации, телефон, электронная почта, паспортные данные, учебное заведение и прочие данные.

Combobox – это элемент управления, который предоставляет пользователю список вариантов на выбор, при этом позволяет также вводить собственное значение в текстовое поле. Для выбора значений в полях «Отдел», «Должность», «Образование», «Гражданство».

DatePicker – это элемент управления, который позволяет пользователю выбрать дату из календаря или ввести её вручную. Для ввода даты имеются элементы «Дата рождения» и «Дата выдачи паспорта», а также радиокнопки для выбора информации о наличии детей и статусе военнообязанных.

Image- это элемент управления, который отображает изображение из указанного источника. В разделе с личными данными присутствует изображение «Аватар», которое можно загрузить.

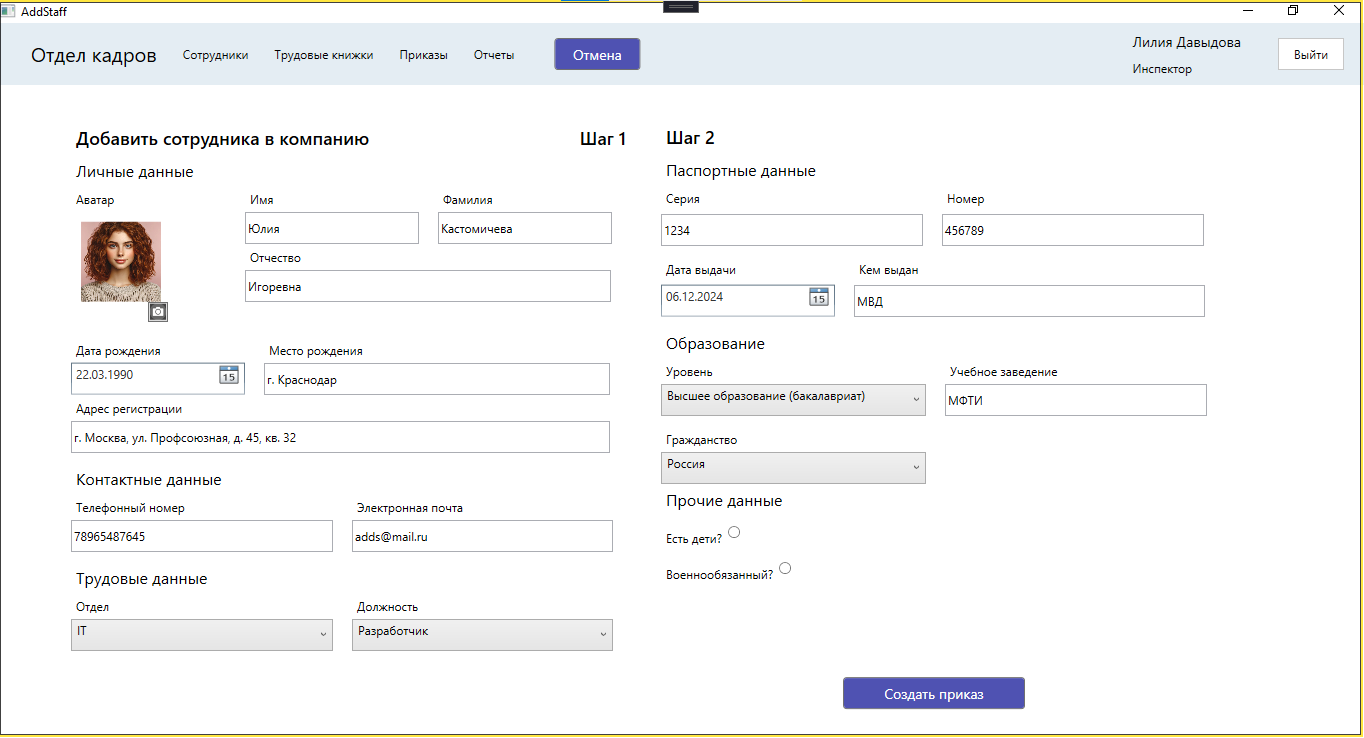


Рисунок 11 – Форма «Добавить сотрудника»

На форме «База данных» (рисунок 12) для администратора используются элементы интерфейса, предназначенные для управления данными в базе данных и навигации по системе.

Кнопки «Отдел кадров», «Сотрудники» и «База данных» отвечают за переходы между различными разделами системы. Кнопка «Выйти» используется для выхода из системы.

ComboBox позволяет выбрать таблицу для добавления новых данных в систему, а TextBox предназначен для ввода этих данных.

Кнопка «Добавить» запускает процесс добавления записи в базу данных.

ListView отображает записи из базы данных, а каждая строка списка имеет кнопку "Удалить" для удаления выбранной записи.

Label служат для отображения информации и подсказок пользователю, а также для уточнения назначения полей ввода и списков.

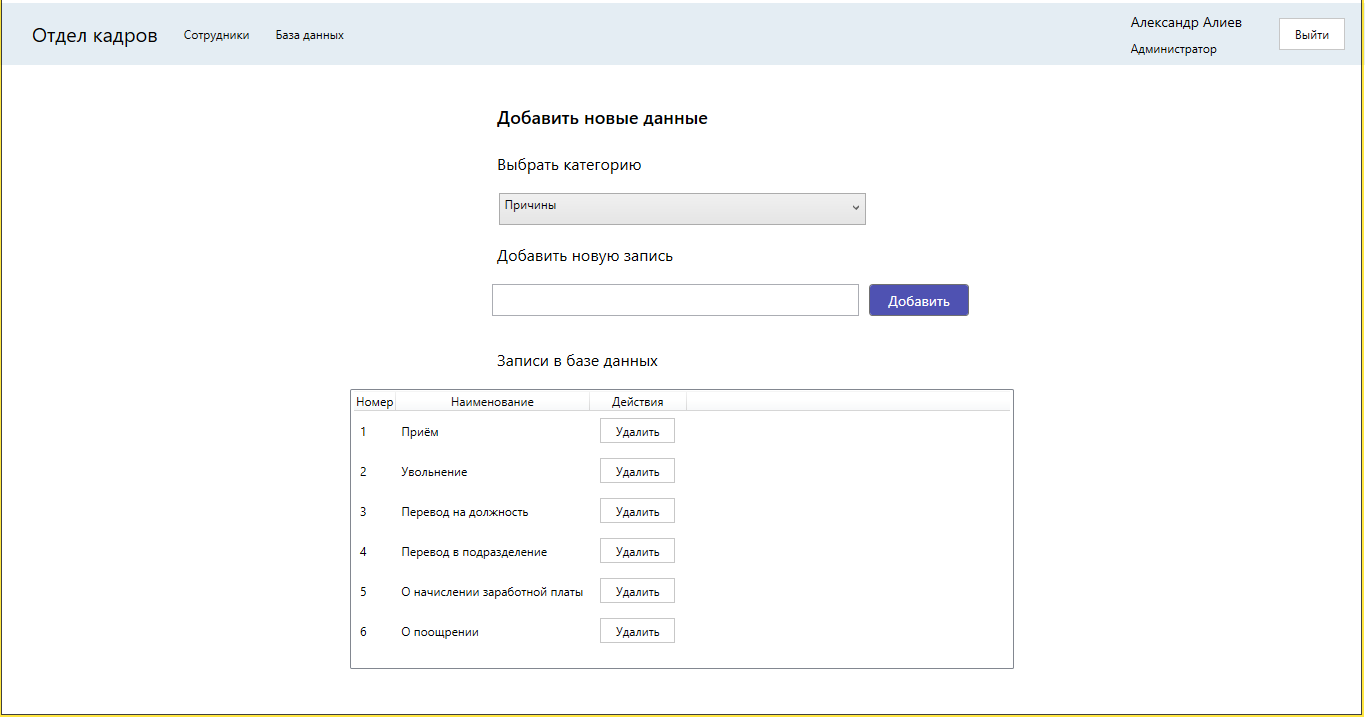


Рисунок 12 – Форма «База данных»

Остальные формы реализованы аналогичным образом с использованием выше указанных компонентов.

3.4 Описание реализации программного продукта

3.4.1 Разработка классов сущностей

**Разработка классов сущностей** – это процесс создания классов, которые представляют основные объекты и данные системы. Эти классы обычно содержат свойства и методы для хранения, обработки и манипуляции данными, связанными с реальными сущностями, такими как пользователи, заказы, продукты и т.д. Классы сущностей могут быть связаны с базой данных и служат для моделирования данных, которые будут сохраняться, извлекаться и обрабатываться системой.

За создание классов сущностей отвечает ADO .NET (ActiveX Data Objects .NET) – это набор классов в .NET Framework, предназначенных для работы с данными в приложениях. Он предоставляет высокоуровневые средства для подключения к различным источникам данных, включая базы данных SQL, веб-сервисы, файлы XML и другие. ADO.NET позволяет разработчикам выполнять операции с данными, такие как извлечение, добавление, обновление и удаление записей, а также работать с транзакциями и управлением соединениями.

ADO.NET использует две основные концепции: соединения и командные объекты. Соединения устанавливают связь с источником данных, а командные объекты выполняют SQL-запросы или хранимые процедуры. Важным аспектом ADO.NET является работа с данными в памяти, например, с помощью объектов DataSet, которые позволяют работать с данными в виде таблиц без необходимости поддержания постоянного соединения с базой данных. Это позволяет создавать эффективные и масштабируемые приложения, которые могут работать с данными в реальном времени, а также обеспечивать поддержку офлайн-режимов работы. На рисунке 13 представлены созданные классы сущностей, сгенерированные из контекста подключенной базы данных.

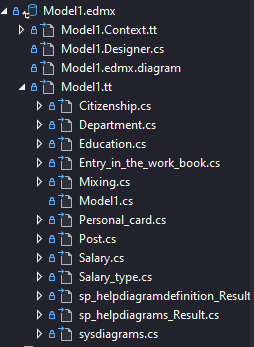


Рисунок 13 – Классы сущностей

Код класса сущности **Entry\_in\_the\_work\_book** представлен на рисунке 14, который используется для хранения данных о записях в трудовой книжке сотрудника. Этот класс включает несколько свойств, таких как уникальный идентификатор записи, ссылка на личную карточку сотрудника, дата записи, причина внесения записи, а также ссылка на сущность **Mixing**, которая содержат дополнительные сведения. Класс использует механизмы связи с другими сущностями, такими как **Personal\_card,** что позволяет эффективно управлять данными, связанными с трудовой деятельностью сотрудников.

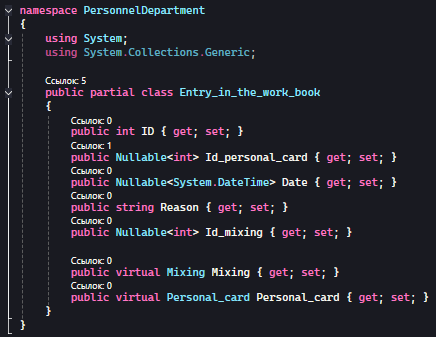


Рисунок 14 – Код класса сущности Entry\_in\_the\_work\_book

Код класса сущности Salary представлен на рисунке 15, который используется для хранения информации о заработной плате сотрудников. Класс включает свойства, такие как уникальный идентификатор, ссылка на личную карточку сотрудника, сумма заработной платы, дата начисления и тип заработной платы. Этот класс связывается с другими сущностями, такими как Personal\_card и Salary\_type, что позволяет эффективно управлять данными о заработной плате и их классификации.

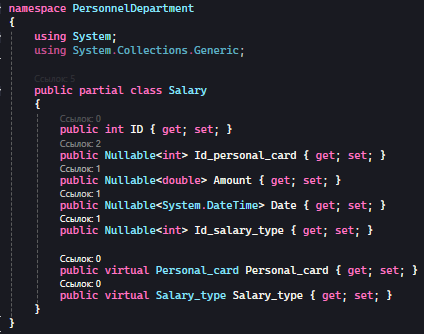


Рисунок 15 – Класс сущности Salary

Код класса сущности Mixing представлен на рисунке 16, который используется для хранения информации о различных типах сведений, связанных с трудовой деятельностью сотрудников. Этот класс включает уникальный идентификатор и название типа сведений. Кроме того, он имеет связь с сущностью Entry\_in\_the\_work\_book, что позволяет хранить множество записей, связанных с конкретным типом сведения. Класс использует коллекцию, чтобы управлять отношениями между типами сведений и соответствующими записями в трудовых книжках.

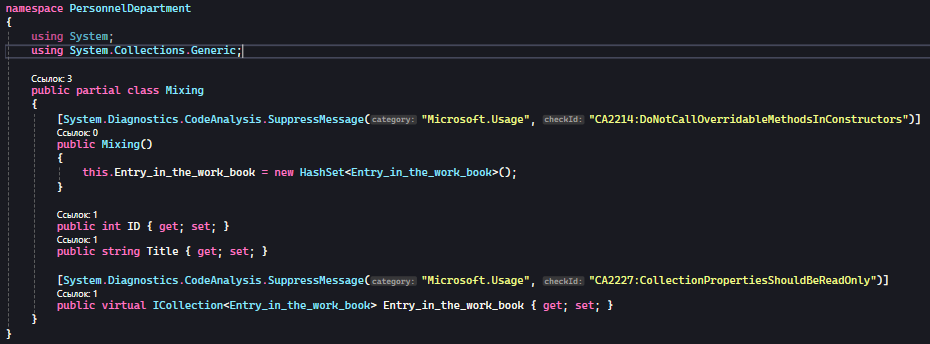


Рисунок 16 – Класс сущностей Mixing

3.4.2 Разработка управляющих классов

**Разработка управляющих классов** включает создание классов, которые управляют логикой работы системы, взаимодействуют с сущностями и реализуют основные операции, такие как создание, обновление, удаление и поиск объектов. Управляющие классы отвечают за выполнение бизнес-логики и взаимодействие между пользователем и сущностями, а также могут обеспечивают работу с базой данных и другими внешними ресурсами. Они служат связующим звеном между интерфейсом пользователя и моделями данных, обеспечивая корректную обработку запросов и реализацию функционала.

На рисунке 17 изображены управляющие классы приложения.

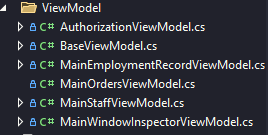


Рисунок 17 – Управляющие классы

AuthorizationViewModel представляет собой класс для авторизации пользователя в приложении. Он наследует BaseViewModel, что позволяет использовать механизм уведомлений об изменении свойств через OnPropertyChanged (таблица 19).

Основные свойства класса включают Login и Password, которые привязаны к соответствующим полям ввода на форме авторизации. Эти свойства используются для хранения логина и пароля пользователя.

Кроме того, класс содержит команду LoginCommand, которая привязана к кнопке "Войти" и выполняет метод ExecuteLogin. В методе ExecuteLogin происходит поиск пользователя в базе данных по введённым логину и паролю. Если пользователь не найден, отображается сообщение об ошибке. Если пользователь найден, проверяется его роль (по ID должности). В зависимости от роли открывается соответствующее главное окно: для администратора – MainWindowAdmin, для инспектора – MainWindowInspector. После этого текущее окно закрывается.

Таким образом, этот ViewModel управляет процессом авторизации, а также осуществляет переход в различные части приложения в зависимости от роли пользователя.

Таблица 19 – Листинг кода управляющего класса «AuthorizationViewModel».

|  |
| --- |
| using PersonnelDepartment.Commands;  using PersonnelDepartment.ControlClasses;  using System.Linq;  using System.Windows;  using System.Windows.Input;  namespace PersonnelDepartment.ViewModel  {  public class AuthorizationViewModel : BaseViewModel  {  private string \_login;  private string \_password;  private ICommand \_loginCommand;  public string Login  {  get => \_login;  set  {  \_login = value;  OnPropertyChanged(nameof(Login));  }  }  public string Password  {  get => \_password;  set  {  \_password = value;  OnPropertyChanged(nameof(Password));  }  }  public ICommand LoginCommand => \_loginCommand ??= new RelayCommand(ExecuteLogin);  private void ExecuteLogin()  {  var user = AppConnect.dbModel.Personal\_card  .FirstOrDefault(p => p.Login == Login && p.Password == Password);  if (user == null)  {  MessageBox.Show("Пользователь не найден!");  return;  }  if (user.Post.ID == 1)  {  var mainWindow = new MainWindowAdmin(user);  mainWindow.Show();  Application.Current.MainWindow.Close();  }  else if (user.Post.ID == 2)  {  var mainWindow = new MainWindowInspector(user);  mainWindow.Show();  Application.Current.MainWindow.Close();  }  }  }  } |

BaseViewModel представляет собой базовый класс, реализующий интерфейс INotifyPropertyChanged, который используется для уведомления об изменении значений свойств в привязанных объектах UI. Он содержит событие PropertyChanged, которое срабатывает каждый раз, когда одно из свойств объекта изменяется (таблица 20).

Метод OnPropertyChanged вызывается внутри дочерних классов, чтобы уведомить систему привязки данных о том, что значение свойства изменилось. Когда это событие срабатывает, оно передает имя изменившегося свойства через PropertyChangedEventArgs, что позволяет обновить отображение данных в интерфейсе.

Таким образом, BaseViewModel предоставляет функциональность для реализации механизма отслеживания изменений в данных и автоматического обновления UI, что является важной частью паттерна MVVM (Model-View-ViewModel).

Таблица 20 – Листинг кода управляющего класса «BaseViewModel».

|  |
| --- |
| using System.ComponentModel;  namespace PersonnelDepartment.ViewModel  {  public class BaseViewModel : INotifyPropertyChanged  {  public event PropertyChangedEventHandler PropertyChanged;  protected void OnPropertyChanged(string propertyName)  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  }  } |

MainEmploymentRecordViewModel управляет функциональностью главного окна «Запись о трудовой деятельности» в информационной системе для отдела кадров. Он инкапсулирует логику для загрузки списка сотрудников и их записей о трудовой деятельности, фильтрации сотрудников по введенному тексту, а также для навигации между различными окнами приложения (таблица 21).

ViewModel включает несколько коллекций: Employees для хранения списка всех сотрудников, FilteredEmployees для отображения отфильтрованных данных и EmploymentRecords для загрузки записей трудовой деятельности выбранного сотрудника. Все эти коллекции привязаны к интерфейсу и обновляются автоматически при изменении данных.

Ключевая функциональность также включает команды, такие как FilterEmployeesCommand для фильтрации списка сотрудников, AddRecordCommand для добавления новой записи о трудовой деятельности, а также навигационные команды для перехода между окнами (например, NavigateToMainWindowCommand, NavigateToOrdersCommand и другие).

Метод LoadEmployeesFromDatabase загружает список сотрудников из базы данных с помощью SQL-запроса, а метод LoadEmploymentRecordsForSelectedEmployee загружает записи о трудовой деятельности для выбранного сотрудника. Если пользователь выбирает сотрудника, система автоматически подгружает его записи и отображает их в интерфейсе.

Также предусмотрены функции для обработки изображений, таких как конвертация байтов в BitmapImage для отображения фото сотрудника, а также обработка открытия окон для добавления записей и навигации.

Таблица 21 –Листинг кода управляющего класса «MainEmploymentRecordViewModel».

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.Commands;  using PersonnelDepartment.EmploymentRecord;  using System.Collections.ObjectModel;  using System.Data.SqlClient;  using System.Linq;  using System.Windows.Input;  using System.Windows;  using System.Windows.Media.Imaging;  using PersonnelDepartment.Staff;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  namespace PersonnelDepartment.ViewModel  {  public class MainEmploymentRecordViewModel : BaseViewModel  {  private Personal\_card \_currentUser;  private Window \_currentWindow;  private string \_userName;  private string \_userRole;  private BitmapImage \_userImage;  private Employee \_selectedEmployee;  private ObservableCollection<Employee> \_employees;  private ObservableCollection<Employee> \_filteredEmployees;  private ObservableCollection<EntryInWorkBook> \_employmentRecords;  public ObservableCollection<Employee> Employees  {  get => \_employees;  set  {  \_employees = value;  OnPropertyChanged(nameof(Employees));  }  }  public ObservableCollection<Employee> FilteredEmployees  {  get => \_filteredEmployees;  set  {  \_filteredEmployees = value;  OnPropertyChanged(nameof(FilteredEmployees));  }  }  public ObservableCollection<EntryInWorkBook> EmploymentRecords  {  get => \_employmentRecords;  set  {  \_employmentRecords = value;  OnPropertyChanged(nameof(EmploymentRecords));  }  }  public Employee SelectedEmployee  {  get => \_selectedEmployee;  set  {  \_selectedEmployee = value;  OnPropertyChanged(nameof(SelectedEmployee));  LoadEmploymentRecordsForSelectedEmployee();  }  }  public ICommand FilterEmployeesCommand { get; set; }  public ICommand AddRecordCommand { get; set; }  public ICommand NavigateToMainWindowCommand { get; set; }  public ICommand NavigateToOrdersCommand { get; set; }  public ICommand NavigateToReportsCommand { get; set; }  public ICommand NavigateToStaffCommand { get; set; }  // public ICommand ExitApplicationCommand { get; set; }  public MainEmploymentRecordViewModel(Personal\_card currentUser, Window currentWindow)  {  \_currentUser = currentUser;  \_currentWindow = currentWindow;  UserName = $"{currentUser.Surname} {currentUser.Name}";  UserRole = currentUser.Post.Title;  if (currentUser.Photo != null)  {  UserImage = ByteArrayToBitmapImage(currentUser.Photo);  }  Employees = LoadEmployeesFromDatabase();  FilteredEmployees = new ObservableCollection<Employee>(Employees);  EmploymentRecords = new ObservableCollection<EntryInWorkBook>();  FilterEmployeesCommand = new RelayCommand<string>(FilterEmployees);  AddRecordCommand = new RelayCommand(AddRecord);  NavigateToMainWindowCommand = new RelayCommand(NavigateToMainWindow);  NavigateToOrdersCommand = new RelayCommand(NavigateToOrders);  NavigateToReportsCommand = new RelayCommand(NavigateToReports);  NavigateToStaffCommand = new RelayCommand(NavigateToStaff);    }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  public void FilterEmployees(string searchText)  {  var filteredList = Employees.Where(emp =>  (!string.IsNullOrEmpty(emp.FirstName) && emp.FirstName.ToLower().Contains(searchText)) ||  (!string.IsNullOrEmpty(emp.LastName) && emp.LastName.ToLower().Contains(searchText)) ||  ($"{emp.FirstName} {emp.LastName}".ToLower().Contains(searchText))  ).ToList();  FilteredEmployees.Clear();  foreach (var emp in filteredList)  {  FilteredEmployees.Add(emp);  }  }  private ObservableCollection<Employee> LoadEmployeesFromDatabase()  {  var employees = new ObservableCollection<Employee>();  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT p.ID, p.Name AS FirstName, p.Surname AS LastName, p.Patronymic, po.Title AS Position, d.Title AS Department, p.Telephone AS Phone, p.Date\_of\_birth AS BirthDate, p.Photo FROM Personal\_card p JOIN Post po ON p.Id\_post = po.ID JOIN Department d ON po.Id\_department = d.ID;", connection);  using var reader = command.ExecuteReader();  while (reader.Read())  {  employees.Add(new Employee  {  ID = reader.GetInt32(0),  FirstName = reader.GetString(1),  LastName = reader.GetString(2),  Patronymic = reader.GetString(3),  Position = reader.GetString(4),  Department = reader.GetString(5),  Phone = reader.GetString(6),  BirthDate = reader.GetDateTime(7),  PhotoData = reader.IsDBNull(8) ? null : (byte[])reader[8]  });  }  }  return employees;  }  public string UserName  {  get => \_userName;  set  {  \_userName = value;  OnPropertyChanged(nameof(UserName));  }  }  public string UserRole  {  get => \_userRole;  set  {  \_userRole = value;  OnPropertyChanged(nameof(UserRole));  }  }  public BitmapImage UserImage  {  get => \_userImage;  set  {  \_userImage = value;  OnPropertyChanged(nameof(UserImage));  }  }  private void LoadEmploymentRecordsForSelectedEmployee()  {  if (SelectedEmployee == null) return;  EmploymentRecords.Clear();  using var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework");  connection.Open();  var command = new SqlCommand(@"SELECT e.ID, e.Date, e.Reason, m.Title AS MixingTitle FROM Entry\_in\_the\_work\_book e LEFT JOIN Mixing m ON e.Id\_mixing = m.ID WHERE e.Id\_personal\_card = @EmployeeId", connection);  command.Parameters.AddWithValue("@EmployeeId", SelectedEmployee.ID);  using var reader = command.ExecuteReader();  while (reader.Read())  {  EmploymentRecords.Add(new EntryInWorkBook  {  ID = reader.GetInt32(0),  Date = reader.GetDateTime(1),  Reason = reader.IsDBNull(2) ? string.Empty : reader.GetString(2),  MixingTitle = reader.IsDBNull(3) ? string.Empty : reader.GetString(3)  });  }  }  private void AddRecord()  {  if (SelectedEmployee == null)  {  MessageBox.Show("Пожалуйста, выберите сотрудника.");  return;  }  AddRecordWindow addRecordWindow = new AddRecordWindow(SelectedEmployee.ID);  addRecordWindow.RecordAdded += (s, args) =>  {  LoadEmploymentRecordsForSelectedEmployee();  };  addRecordWindow.Show();  }  private void NavigateToMainWindow()  {  MainStaff mainStaffWindow = new MainStaff(\_currentUser);  mainStaffWindow.Show();  \_currentWindow.Close();  }  private void NavigateToOrders()  {  MainOrders mainOrdersWindow = new MainOrders(\_currentUser);  mainOrdersWindow.Show();  \_currentWindow.Close();  }  private void NavigateToReports()  {  MainReports mainReportsWindow = new MainReports(\_currentUser);  mainReportsWindow.Show();  \_currentWindow.Close();  }  private void NavigateToStaff()  {  MainStaff mainStaffWindow = new MainStaff(\_currentUser);  mainStaffWindow.Show();  \_currentWindow.Close();  } |

MainStaffViewModel управляет логикой главного окна для сотрудников. Он обрабатывает отображение списка сотрудников, их фильтрацию по введенному поисковому запросу, а также предоставляет команды для выполнения действий с сотрудниками (таблица 22). В частности, ViewModel поддерживает команды для добавления новых сотрудников, назначения наград, а также навигацию между различными окнами, такими как окна для записи трудовой деятельности, приказы и отчеты.

Основной функционал включает фильтрацию списка сотрудников по имени или фамилии с помощью команды SearchCommand. Список сотрудников загружается из базы данных в коллекцию Employees, а отфильтрованные данные хранятся в FilteredEmployees, которые автоматически обновляются при изменении текста поиска.

Для работы с отдельными сотрудниками, ViewModel поддерживает команду EditEmployeeCommand, которая открывает окно редактирования информации о сотруднике, а также команду AssignRewardCommand для назначения награды выбранному сотруднику. Кроме того, реализованы команды для перехода между различными окнами приложения, такими как окно для работы с трудовыми книжками, заказы и отчеты.

Для отображения фотографий сотрудников используется метод ByteArrayToBitmapImage, который конвертирует байтовые данные фотографии в объект BitmapImage для отображения в интерфейсе.

Таблица 22 – Листинг кода управляющего класса «MainStaffViewModel».

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.Commands;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  using PersonnelDepartment.Staff;  using System.Collections.ObjectModel;  using System.Data.SqlClient;  using System.Linq;  using System.Windows;  using System.Windows.Input;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.ViewModel  {  public class MainStaffViewModel : BaseViewModel  {  private ObservableCollection<Employee> \_employees;  private ObservableCollection<Employee> \_filteredEmployees;  private string \_searchText;  private Employee \_selectedEmployee;  private Personal\_card \_currentUser;  private Window \_currentWindow;  private string \_userName;  private string \_userRole;  private BitmapImage \_userImage;  public ObservableCollection<Employee> Employees  {  get => \_employees;  set  {  \_employees = value;  OnPropertyChanged(nameof(Employees));  FilterEmployees();  }  }  public ObservableCollection<Employee> FilteredEmployees  {  get => \_filteredEmployees;  set  {  \_filteredEmployees = value;  OnPropertyChanged(nameof(FilteredEmployees));  }  }  public string SearchText  {  get => \_searchText;  set  {  \_searchText = value;  OnPropertyChanged(nameof(SearchText));  FilterEmployees();  }  }  public Employee SelectedEmployee  {  get { return \_selectedEmployee; }  set  {  \_selectedEmployee = value;  OnPropertyChanged(nameof(SelectedEmployee));  }  }  public ICommand SearchCommand { get; }  public ICommand AddStaffCommand { get; }  public ICommand ExitCommand { get; }  public ICommand AssignRewardCommand { get; }  public ICommand NavigateToMainWindowCommand { get; }  public ICommand NavigateToEmploymentRecordCommand { get; }  public ICommand NavigateToOrdersCommand { get; }  public ICommand NavigateToReportsCommand { get; }  private ICommand \_editEmployeeCommand;  public ICommand EditEmployeeCommand => \_editEmployeeCommand ??= new RelayCommand<Employee>(OpenEditEmployeeWindow);  public MainStaffViewModel(Personal\_card currentUser, Window currentWindow)  {  \_currentUser = currentUser;  \_currentWindow = currentWindow;  \_employees = LoadEmployeesFromDatabase();  \_filteredEmployees = new ObservableCollection<Employee>(\_employees);  UserName = $"{currentUser.Surname} {currentUser.Name}";  UserRole = currentUser.Post.Title;  if (currentUser.Photo != null)  {  UserImage = ByteArrayToBitmapImage(currentUser.Photo);  }  SearchCommand = new RelayCommand(Search);  AddStaffCommand = new RelayCommand(AddStaff);  AssignRewardCommand = new RelayCommand(AssignReward);  NavigateToMainWindowCommand = new RelayCommand(NavigateToMainWindow);  NavigateToEmploymentRecordCommand = new RelayCommand(NavigateToEmploymentRecord);  NavigateToOrdersCommand = new RelayCommand(NavigateToOrders);  NavigateToReportsCommand = new RelayCommand(NavigateToReports);  }  public void OpenEditEmployeeWindow(Employee employee)  {  if (employee != null)  {  EditStaff editWindow = new EditStaff(employee, \_currentUser);  editWindow.Show();  \_currentWindow.Close();  }  }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  public string UserName  {  get => \_userName;  set  {  \_userName = value;  OnPropertyChanged(nameof(UserName));  }  }  public string UserRole  {  get => \_userRole;  set  {  \_userRole = value;  OnPropertyChanged(nameof(UserRole));  }  }  public BitmapImage UserImage  {  get => \_userImage;  set  {  \_userImage = value;  OnPropertyChanged(nameof(UserImage));  }  }  private void Search()  {  FilterEmployees();  }  private void FilterEmployees()  {  string searchText = SearchText?.Trim().ToLower();  if (string.IsNullOrEmpty(searchText))  {  FilteredEmployees.Clear();  foreach (var emp in Employees)  {  FilteredEmployees.Add(emp);  }  }  else  {  var filteredList = Employees.Where(emp =>  (!string.IsNullOrEmpty(emp.FirstName) && emp.FirstName.ToLower().Contains(searchText)) ||  (!string.IsNullOrEmpty(emp.LastName) && emp.LastName.ToLower().Contains(searchText)) ||  ($"{emp.FirstName} {emp.LastName}".ToLower().Contains(searchText))  ).ToList();  FilteredEmployees.Clear();  foreach (var emp in filteredList)  {  FilteredEmployees.Add(emp);  }  }  }  private ObservableCollection<Employee> LoadEmployeesFromDatabase()  {  ObservableCollection<Employee> employees = new ObservableCollection<Employee>();  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT p.ID, p.Name AS FirstName, p.Surname AS LastName, p.Patronymic, po.Title AS Position, d.Title AS Department, p.Telephone AS Phone, p.Date\_of\_birth AS BirthDate, p.Photo FROM Personal\_card p JOIN Post po ON p.Id\_post = po.ID JOIN Department d ON po.Id\_department = d.ID;", connection);  using (var reader = command.ExecuteReader())  {  while (reader.Read())  {  employees.Add(new Employee  {  ID = reader.GetInt32(0),  FirstName = reader.GetString(1),  LastName = reader.GetString(2),  Patronymic = reader.GetString(3),  Position = reader.GetString(4),  Department = reader.GetString(5),  Phone = reader.GetString(6),  BirthDate = reader.GetDateTime(7),  PhotoData = reader.IsDBNull(8) ? null : (byte[])reader[8]  });  }  }  }  return employees;  }  private void AddStaff()  {  var addStaff = new AddStaff(\_currentUser);  addStaff.Show();  \_currentWindow.Close();  }  private void AssignReward()  {  if (SelectedEmployee == null)  {  MessageBox.Show("Выберите сотрудника из списка.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);  return;  }  var assignRewardForm = new AssignRewardForm(SelectedEmployee.ID, $"{SelectedEmployee.LastName} {SelectedEmployee.FirstName} {SelectedEmployee.Patronymic}");  assignRewardForm.ShowDialog();  }  private void NavigateToMainWindow()  {  new MainWindowInspector(\_currentUser).Show();  Application.Current.Windows.Cast<Window>().FirstOrDefault(w => w is MainStaff)?.Close();  }  private void NavigateToEmploymentRecord()  {  new MainEmploymentRecord(\_currentUser).Show();  \_currentWindow.Close();  }  private void NavigateToOrders()  {  new MainOrders(\_currentUser).Show();  \_currentWindow.Close();  }  private void NavigateToReports()  {  new MainReports(\_currentUser).Show();  \_currentWindow.Close();  }  }  } |

MainWindowInspectorViewModel управляет функциональностью главного окна инспектора. Он инкапсулирует данные о текущем пользователе и предоставляет команды для навигации между различными разделами приложения (таблица 23).

Основные свойства включают UserName, UserRole и UserImage, которые отображают информацию о пользователе, вошедшем в систему. Для отображения фотографии пользователя используется метод ByteArrayToBitmapImage, который преобразует байтовые данные в объект BitmapImage.

Команды, такие как StaffCommand, EmploymentRecordCommand, OrdersCommand и ReportCommand, обеспечивают навигацию по разделам приложения. Каждая команда открывает соответствующее окно (например, для управления сотрудниками, записи трудовой деятельности, заказами и отчетами) и закрывает текущее окно.

Этот ViewModel предоставляет основные механизмы для работы с интерфейсом инспектора, помогая ему легко переходить между различными разделами системы и отображать актуальную информацию о пользователе.

Таблица 23 – Листинг кода управляющего класса «MainWindowInspectorViewModel».

|  |
| --- |
| using PersonnelDepartment.Commands;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  using PersonnelDepartment.Staff;  using System.Windows;  using System.Windows.Input;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.ViewModel  {  public class MainWindowInspectorViewModel : BaseViewModel  {  private Personal\_card \_currentUser;  private string \_userName;  private string \_userRole;  private BitmapImage \_userImage;  private Window \_currentWindow;  public MainWindowInspectorViewModel(Personal\_card currentUser, Window currentWindow)  {  \_currentUser = currentUser;  \_currentWindow = currentWindow;  UserName = $"{currentUser.Surname} {currentUser.Name}";  UserRole = currentUser.Post.Title;  if (currentUser.Photo != null)  {  UserImage = ByteArrayToBitmapImage(currentUser.Photo);  }  StaffCommand = new RelayCommand(ExecuteStaff);  EmploymentRecordCommand = new RelayCommand(ExecuteEmploymentRecord);  OrdersCommand = new RelayCommand(ExecuteOrders);  ReportCommand = new RelayCommand(ExecuteReports);  }  public string UserName  {  get => \_userName;  set  {  \_userName = value;  OnPropertyChanged(nameof(UserName));  }  }  public string UserRole  {  get => \_userRole;  set  {  \_userRole = value;  OnPropertyChanged(nameof(UserRole));  }  }  public BitmapImage UserImage  {  get => \_userImage;  set  {  \_userImage = value;  OnPropertyChanged(nameof(UserImage));  }  }  public ICommand StaffCommand { get; }  public ICommand EmploymentRecordCommand { get; }  public ICommand OrdersCommand { get; }  public ICommand ReportCommand { get; }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  private void ExecuteStaff()  {  MainStaff mainStaffWindow = new MainStaff(\_currentUser);  mainStaffWindow.Show();  \_currentWindow.Close();  }  private void ExecuteEmploymentRecord()  {  MainEmploymentRecord mainEmploymentRecordWindow = new MainEmploymentRecord(\_currentUser);  mainEmploymentRecordWindow.Show();  \_currentWindow.Close();  }  private void ExecuteOrders()  {  MainOrders mainOrdersWindow = new MainOrders(\_currentUser);  mainOrdersWindow.Show();  \_currentWindow.Close();  }  private void ExecuteReports()  {  MainReports mainReportsWindow = new MainReports(\_currentUser);  mainReportsWindow.Show();  \_currentWindow.Close();  }  }  } |

**3.5 Тестирование ИС**

**3.5.1. Создание тест-кейсов**

Тест-кейсы – это детализированные сценарии, которые описывают шаги для проверки функциональности программного обеспечения. Они являются основным инструментом в процессе тестирования и помогают убедиться, что приложение или система работает согласно заданным требованиям. Каждый тест-кейс включает информацию о входных данных, ожидаемом результате и критериях успешности. Это позволяет систематически проверять различные функции приложения, выявлять ошибки и гарантировать соответствие конечного продукта установленным стандартам.

Тест-кейсы обычно создаются тестировщиками или разработчиками, чтобы описать как проводить тестирование отдельных функций или пользовательских сценариев. Они могут включать позитивные тесты, чтобы убедиться, что система правильно работает при корректных данных, а также негативные тесты, проверяющие её поведение при ошибочных входных данных или нештатных ситуациях. Каждый тест-кейс должен быть независимым и легко повторяемым, чтобы результат тестирования был надежным и воспроизводимым в разных условиях. Тестирование с использованием тест-кейсов помогает минимизировать количество дефектов и улучшить качество программного обеспечения.

Тест-кейс «Добавление сотрудника в базу данных», представленный в таблице 24, проверяет успешное добавление сотрудника в базу данных. Ожидается, что при вводе корректных данных в соответствующие поля формы все введенные значения будут отображаться правильно, а при нажатии на кнопку «Создать приказ» данные будут успешно добавлены в таблицу «Personal\_card». Также тест проверяет корректную работу поля для загрузки фото, выбор значений из выпадающих списков и радиокнопок, а также правильность отображения выбранных значений.

Таблица 24 – Позитивные тест-кейсы для формы «Добавление сотрудника».

|  |  |
| --- | --- |
| Номер: | 1 |
| Заголовок: | Добавление сотрудника в базу данных. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». |
| Шаг | Ожидаемый результат |
| В поле «Имя» ввести «Иван». | В поле «Имя» отобразился введенный текст. |
| В поле «Фамилия» ввести «Иванов». | В поле «Фамилия» отобразился введенный текст. |
| В поле «Отчество» ввести «Иванович». | В поле «Отчество» отобразился введенный текст. |

Продолжение таблицы 24.

|  |  |
| --- | --- |
| В поле «Телефон» ввести 78965431234. | В поле «Телефон» отобразился введенный текст. |
| В поле «Электронная почта» ввести «abcd@mail.ru». | В поле «Электронная почта» отобразился введенный текст. |
| В поле «Дата рождения» ввести 15.03.2020. | В поле «Дата рождения» отобразился введенный текст. |
| В поле «Место рождения» ввести «г. Тула». | В поле «Место рождения» отобразился введенный текст. |
| В поле «Адрес регистрации» ввести «г. Тула, ул. Ленина, д. 15, кв. 15». | В поле «Адрес регистрации» отобразился введенный текст. |
| В поле «Серия» ввести «1234». | В поле «Серия» отобразился введенный текст. |
| В поле «Номер» ввести «123456». | В поле «Номер» отобразился введенный текст. |
| В поле «Дата выдачи» ввести «14.10.2022». | В поле «Дата выдачи» отобразился введенный текст. |
| В поле «Кем выдан» ввести «МВД». | В поле «Кем выдан» отобразился введенный текст. |
| В поле «Учебное заведение» ввести «МГУ». | В поле «Учебное заведение» отобразился введенный текст. |
| В поле со списком «Отдел» выбрать «Технический». | В поле со списком «Должность» отсортировано в поле «Отдел» отображается значение «Инженер». |
| В поле со списком «Должность» выбрать «Инженер». | В поле со списком «Должность» отображается выбранное значение. |
| В поле со списком «Уровень» «Высшее образование (бакалавриат)». | В поле со списком «Уровень» отображается выбранное значение. |
| В поле со списком «Гражданство» выбрать «Россия». | В поле со списком «Гражданство» отображается выбранное значение. |
| Выбрать радиокнопку в поле «Есть дети?». | Радиокнопка приняла значение true. |
| Выбрать радиокнопку в поле «Военнообязанный?». | Радиокнопка приняла значение true. |
| Нажать на кнопку для загрузки фото | Открылся проводник. |
| Выбрать фото | Фото отображается на форме в поле «Аватар». |
| Нажать на кнопку «Создать приказ». | Все данные добавлены в таблицу «Personal\_card». |

Тест-кейс «Ввод строчной буквы в поле «Имя», который представлен в таблице 25, проверяет корректность обработки ввода строчной буквы, чтобы убедиться, что система правильно отображает введённый текст без ошибок валидации.

Таблица 25 – Негативный тест-кейс для поля «Имя».

|  |  |
| --- | --- |
| Номер: | 2 |
| Заголовок: | Ввод строчной буквы в поле «Имя». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Имя» ввести «иван». | В поле «Имя» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Имя». |

Тест-кейс «Ввод в поле «Фамилия» чисел», который представлен в таблице 26, проверяет, как система реагирует на ввод чисел, гарантируя, что при попытке создать приказ будет отображено сообщение об ошибке из-за некорректного ввода.

Таблица 26 – Негативный тест-кейс для поля «Фамилия».

|  |  |
| --- | --- |
| Номер: | 3 |
| Заголовок: | Ввод в поле «Фамилия» чисел. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Фамилия» ввести «Иванов123». | В поле «Фамилия» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Фамилия». |

Тест-кейс «Ввод специального символа в поле «Отчество», представленный в таблице 27, проверяет, как система обрабатывает ввод специального символа. Цель – убедиться, что система выявляет некорректные данные и уведомляет пользователя об ошибке.

Таблица 27 – Негативный тест-кейс для поля «Отчество».

|  |  |
| --- | --- |
| Номер: | 4 |
| Заголовок: | Ввод специального символа в поле «Отчество». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Отчество» ввести «Иванович@». | В поле «Отчество» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Отчество». |

Тест-кейс «Ввод номера с буквой в поле «Телефон», описанный в таблице 28, направлен на проверку обработки некорректного ввода в поле «Телефон», содержащего букву. Целью теста является удостовериться, что система распознаёт ошибку и информирует пользователя о некорректности данных.

Таблица 28 – Негативный тест-кейс для поля «Телефон».

|  |  |
| --- | --- |
| Номер: | 5 |
| Заголовок: | Ввод номера с буквой в поле «Телефон». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Телефон» ввести «7896 543123а». | В поле «Телефон» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Телефон». |

Тест-кейс «Ввод значения поля «Электронная почта» без знака @», представленный в таблице 29, проверяет, как система обрабатывает ввод адреса электронной почты без обязательного знака «@». Цель – убедиться, что система корректно определяет некорректный формат ввода и уведомляет пользователя об ошибки.

Таблица 29 – Негативный тест-кейс для поля «Электронная почта».

|  |  |
| --- | --- |
| Номер: | 6 |
| Заголовок: | Ввод значения поля «Электронная почта» без знака @. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Электронная почта» ввести «abcdmail.ru». | В поле «Электронная почта» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Электронная почта». |

Тест-кейс «Выбор даты в поле «Дата рождения» больше системной», описанный в таблице 30, проверяет поведение системы при вводе даты рождения, превышающей текущую системную дату. Цель теста – убедиться, что система обнаруживает ошибку в данных и уведомляет пользователя о некорректном вводе при попытке создания приказа.

Таблица 30 – Негативный тест-кейс для поля «Дата рождения».

|  |  |
| --- | --- |
| Номер: | 7 |
| Заголовок: | Выбор даты в поле «Дата рождения» больше системной. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Дата рождения» ввести «13.09.2028». | В поле «Дата рождения» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Дата рождения». |

Тест-кейс «Ввод значения в поле «Место рождения» с прописной буквы», описанный в таблице 31, проверяет корректность обработки ввода текста, начинающегося с маленькой буквы. Система должна правильно распознать неверный формат данных и отобразить сообщение об ошибке при попытке сохранения.

Таблица 31 – Негативный тест-кейс для поля «Место рождения».

|  |  |
| --- | --- |
| Номер: | 8 |
| Заголовок: | Ввод значения в поле «Место рождения» с прописной буквы. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |

Продолжение таблицы 31.

|  |  |
| --- | --- |
| В поле «Место рождения» ввести «тула». | В поле «Место рождения» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Место рождения». |

Тест-кейс «Ввод чисел в поле «Адрес регистрации», описанный в таблице 32, проверяет реакцию системы на ввод числового значения. Система должна идентифицировать ошибку и уведомить пользователя о некорректности данных при попытке создания приказа.

Таблица 32 – Негативный тест-кейс для поля «Адрес регистрации».

|  |  |
| --- | --- |
| Номер: | 9 |
| Заголовок: | Ввод чисел в поле «Адрес регистрации». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Адрес регистрация» ввести «12345». | В поле «Адрес регистрации» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Адрес регистрации». |

Тест-кейс «Ввод буквы в поле «Серия», представленный в таблице 33, проверяет обработку ввода буквенного значения. Ожидается, что система обнаружит некорректные данные и выдаст сообщение об ошибке при попытке сохранении.

Таблица 33 – Негативный тест-кейс для поля «Серия».

|  |  |
| --- | --- |
| Номер: | 10 |
| Заголовок: | Ввод буквы в поле «Серия». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Серия» ввести «123а». | В поле «Серия» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Серия». |

Тест-кейс «Ввод специального символа в поле «Номер», представленный в таблице 34, проверяет поведение системы при вводе специального символа. Ожидается, что система корректно распознает неверные данные и отобразит сообщение об ошибке при попытке создать приказ.

Таблица 34 – Негативный тест-кейс для поля «Номер».

|  |  |
| --- | --- |
| Номер: | 11 |
| Заголовок: | Ввод специального символа в поле «Номер». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |

Продолжение таблицы 34.

|  |  |
| --- | --- |
| В поле «Номер» ввести «123456@». | В поле «Номер» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Номер». |

Тест-кейс «Ввод даты с специальным символом в поле «Дата выдачи», описанный в таблице 35, проверяет, как система реагирует на ввод даты с использованием специального символа. Ожидается, что система определит неправильный формат и сообщит пользователю о некорректном вводе при попытке создания приказа.

Таблица 35 – Негативный тест-кейс для поля «Дата выдачи».

|  |  |
| --- | --- |
| Номер: | 12 |
| Заголовок: | Ввод даты с специальным символом в поле «Дата выдачи». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Дата выдачи» ввести «14,10.2024». | В поле «Дата выдачи» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Дата выдачи». |

Тест-кейс «Ввод чисел в поле «Кем выдан», представленный в таблице 36, проверяет, как система обрабатывает ввод числовых значений. Ожидается, что при попытке создать приказ система выявит ошибку и уведомит пользователя о некорректности данных.

Таблица 36 – Негативный тест-кейс для поля «Кем выдан».

|  |  |
| --- | --- |
| Номер: | 13 |
| Заголовок: | Ввод чисел в поле «Кем выдан». |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Кем выдан» ввести «12345». | В поле «Кем выдан» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Кем выдан». |

Тест-кейс «Ввод значения со специальным символом в поле «Учебное заведение»., описанный в таблице 37, направлен на проверку системы при вводе значения с использованием специального символа. Ожидается, что система обнаружит некорректный ввод и отобразит сообщение об ошибке при попытке создать приказ.

Таблица 37 – Негативный тест-кейс для поля «Учебное заведение».

|  |  |
| --- | --- |
| Номер: | 14 |
| Заголовок: | Ввод значения со специальным символом в поле «Учебное заведение». |
|  |  |

Продолжение таблицы 37.

|  |  |
| --- | --- |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| В поле «Учебное заведение» ввести «МГУ@». | В поле «Учебное заведение» отобразился введенный текст. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Некорректный ввод в поле «Учебное заведение». |

Тест-кейс, «Поле со списком «Отдел» пустое» представленный в таблице 38, проверяет, как система реагирует на незаполненное поле. Ожидается, что при попытке создания приказа система уведомит пользователя о том, что не может быть пустым, а также поле «Должность» остается пустым.

Таблица 38 – Негативный тест-кейс для поля со списком «Отдел».

|  |  |
| --- | --- |
| Номер: | 15 |
| Заголовок: | Поле со списком «Отдел» пустое. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| Оставить поле «Отдел» не заполненным. | Поле со списком «Должность» пустое.  Поле со списком «Отдел» пустое. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Поле со списком «Отдел» пустое. |

Тест-кейс «Поле со списком «Должность» пустое», описанный в таблице 39, проверяет, как система обрабатывает незаполненное поле. Ожидается, что при попытке создания приказа система сообщит, что поле не может быть не заполнено.

Таблица 39 – Негативный тест-кейс для поля со списком «Должность».

|  |  |
| --- | --- |
| Номер: | 16 |
| Заголовок: | Поле со списком «Должность» пустое. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| Оставить поле «Должность» не заполненным. | Поле со списком «Должность» пустое. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Поле со списком «Должность» пустое. |

Тест-кейс «Поле со списком «Уровень» пустое», представленный в таблице 40, проверяет поведение системы при незаполненном поле. Ожидается, что при попытке создать приказ система уведомит пользователя о необходимости выбрать значение.

Таблица 40 – Негативный тест-кейс для поля со списком «Уровень».

|  |  |
| --- | --- |
| Номер: | 17 |
| Заголовок: | Поле со списком «Уровень» пустое. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| Оставить поле со списком «Уровень» не заполненным. | Поле со списком «Уровень» пустое. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Выберите значение в поле со списком «Уровень». |

Тест-кейс «Поле со списком «Гражданство» пустое», описанный в таблице 41, проверяет, как система реагирует на незаполненное поле. Ожидается, что при попытке создания приказа система сообщит пользователю о необходимости выбрать значение.

Таблица 41 – Негативный тест-кейс для поля со списком «Гражданство».

|  |  |
| --- | --- |
| Номер: | 18 |
| Заголовок: | Поле со списком «Гражданство» пустое. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| Оставить поле со списком «Гражданство» не заполненным. | Поле со списком «Гражданство» пустое. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Выберите значение в поле со списком «Гражданство». |

Тест-кейс «Поле со списком «Фото» пустое», описанный в таблице 42, проверяет, как система обрабатывает незаполненное поле. Ожидается, что при попытке создания заказа система уведомит пользователя о необходимости загрузить изображение.

Таблица 42 – Негативный тест-кейс для поля «Аватар».

|  |  |
| --- | --- |
| Номер: | 19 |
| Заголовок: | Поле со списком «Аватар» пустое. |
| Предусловие: | Запущена программа. Открыта форма «Добавить сотрудника». Поля заполнены корректно. |
| Шаг | Ожидаемый результат |
| Оставить поле «Аватар» не заполненным. | Поле со списком «Аватар» пустое. |
| Нажать на кнопку «Создать приказ». | Сообщение об ошибке: Выберите значение в поле «Аватар». |

После выполнения тестов можно сделать вывод, что система корректно проверяет данные, выводит правильные сообщения об ошибках и корректно сохраняет информацию в базе данных. Тесты подтвердили стабильность работы системы в различных сценариях ввода данных.

Модульное тестирование доказало свою эффективность, обеспечив надёжность функционала и повысив качество программного обеспечения.

**3.5.2 Модульный тест**

Модульные тесты – это автоматизированные тесты, которые проверяют корректность работы отдельных компонентов или модулей программы, таких как функции, методы или классы. Модульные тесты направлены на проверку каждого модуля изолированно, чтобы убедиться, что он выполняет свою задачу правильно. В процессе тестирования модуля проверяются различные входные данные и ожидаются соответствующие результаты, что помогает обнаружить ошибки на ранней стадии разработки. Модульные тесты обычно выполняются с помощью фреймворков тестирования MSTest.

Проект модульного теста (.NET Framework) –это отдельный проект, включающий в себя тесты для компонентов приложения. Проект модульного теста в .NET Framework предназначен для создания и выполнения тестов с использованием фреймворков тестирования, таких как MSTest. В таком проекте создаются классы тестов, которые обеспечивают проверку различных функций или методов программы. Модульные тесты изолируют компоненты системы, что позволяет тестировать каждый модуль отдельно, а не всю систему в целом.

После рассмотрения теоретической основы модульных тестов можно перейти к практике их применения при тестировании системы. Модульные тесты играют важную роль в обеспечении качества программного обеспечения, проверяя корректность работы отдельных компонентов системы, таких как функции, методы или классы. В данном случае основным объектом тестирования является валидатор данных, который проверяет корректность ввода различных пользовательских данных. Это критически важная часть системы, так как ошибки на этом этапе могут привести к некорректному сохранению или обработке данных.

В процессе тестирования мы будем рассматривать различные сценарии, включая как правильный ввод данных, так и ситуации, когда пользователь ошибается при заполнении формы. Модульные тесты направлены на проверку того, что система корректно обрабатывает каждый из таких случаев. Важно убедиться, что в случае ошибки валидации система выводит соответствующие сообщения, что позволяет пользователю быстро исправить введённые данные.

1 тест: Test\_ValidData – проверяется, что корректно введенные данные в поля (имя, фамилия, отчество, телефон, электронная почта, дата рождения и другие) проходят проверку без ошибок. Также проверяется, что данные корректно сохраняются в базе данных и могут быть извлечены и проверены.

2 тест: Test\_InvalidName\_Lowercase – проверяется, что если имя вводится с маленькой буквы, то валидация возвращает ошибку с соответствующим сообщением (например, «Некорректный ввод в поле «Имя»»).

3 тест: Test\_InvalidSurname\_WithNumbers – проверяется, что если в поле «Фамилия» вводятся цифры, то валидация выводит ошибку, так как фамилия не может содержать цифры.

4 тест: Test\_InvalidPatronymic\_WithSpecialChar – проверяется, что если в поле «Отчество» используются специальные символы (например, «Иванович@»), то проверка вызовет ошибку, указывающую на некорректный ввод.

5 тест: Test\_InvalidPhone\_WithLetter – проверяется, что если в поле для ввода номера телефона есть буквы (например, «78965431234a»), то проверка возвращает ошибку, так как номер телефона должен содержать только цифры.

6 тест: Test\_InvalidEmail\_WithoutAtSign – проверяется, что если электронная почта не содержит символа «@» (например, «abcdmail.ru»), то проверка возвращает ошибку, так как почта должна быть в правильном формате.

7 тест: Test\_InvalidBirthDate\_FutureDate – проверяется, что если введенная дата рождения больше текущей системной даты, то проверка возвращает ошибку, так как дата рождения не может быть в будущем.

8 тест: Test\_InvalidPlace\_WithLowercase – проверяется, что если место рождения введено с маленькой буквы (например, «Тула»), то проверка вернет ошибку, так как место рождения должно начинаться с заглавной буквы.

9 тест: Test\_InvalidAddress\_WithNumbers – проверяется, что если в поле «Адрес регистрации» введены только цифры (например, «12345»), то проверка возвращает ошибку, так как адрес должен содержать более полное описание, включая текст.

10 тест: Test\_InvalidPassportSeries\_WithLetter – проверяется, что если в поле «Серия паспорта» введена буква (например, «123a»), то проверка возвращает ошибку, так как серия паспорта должна состоять только из цифр.

11 тест: Test\_InvalidPassportNumber\_WithSpecialChar – проверяется, что если в поле «Номер паспорта» введен специальный символ (например, «12345@»), то проверка возвращает ошибку, так как номер паспорта должен содержать только цифры.

12 тест: Test\_InvalidDateOfIssue\_WithSpecialChar – проверяется, что если дата выдачи паспорта введена с ошибками (например, с запятой «14.10.2024»), то проверка возвращает ошибку.

13 тест: Test\_InvalidIssuedBy\_WithNumbers – проверяется, что если в поле «Кем выдан» введены только цифры (например, «12345»), то валидация выдаст ошибку, так как это поле должно содержать название органа, выдавшего документ.

14 тест: Test\_InvalidEducationInstitution\_WithSpecialChar – проверяется, что если в поле «Учебное заведение» введен специальный символ (например, «МГУ@»), то валидация возвращает ошибку, так как название учебного заведения не должно содержать символов.

15 тест: Test\_EmptyDepartment – проверяется, что если в поле «Отдел» не выбрано значение (пустое поле), то проверка вернет ошибку, так как это поле обязательно для заполнения.

16 тест: Test\_EmptyPost – проверяется, что если в поле «Должность» не выбрано значение (пустое поле), то проверка возвращает ошибку, так как должность обязательно должна быть указана.

17 тест: Test\_EmptyEducationLevel – проверяется, что если в поле «Уровень образования» не выбрано значение (пустое поле), то валидация выводит ошибку, так как этот параметр обязательно должен быть указан.

18 тес: Test\_EmptyCitizenship – проверяется, что если в поле «Гражданство» не выбрано значение (пустое поле), то проверка возвращает ошибку, так как гражданство является обязательным полем.

19 тест: Test\_InvalidPhoto\_FileNotFound – проверяется, что если в поле «Аватар» не выбрана фотография или файл не найден, то валидация выводит ошибку, требуя загрузить изображение.

Представленные тесты играют ключевую роль в обеспечении надежности и стабильности системы, поскольку они позволяют выявить возможные ошибки на ранних этапах разработки и гарантируют, что в процессе работы с данными не возникнет проблем с их обработкой или сохранением. Эти тесты являются важной частью процесса разработки, поскольку позволяют убедиться в правильности реализации функционала и проверки данных. В таблице 43 будет написан листинг кода тестов, проверяющие различные сценарии ввода данных, которые гарантируют корректную работу системы при взаимодействии с пользователем.

Таблица 43 – Листинг кода модульных тестов.

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using PersonnelDepartment;  using PersonnelDepartment.ClassHelper;  using System;  using System.IO;  using System.Linq;  namespace HumanDepartmaentTests  {  [TestClass]  public class ValidatorTests  {  private HumanResourcesDepartmentEntities dbModel;  [TestInitialize]  public void Setup()  {  dbModel = new HumanResourcesDepartmentEntities();  }  // Тест 1 - Ввод с заглавной буквы значения в поле «Имя»  [TestMethod]  public void Test\_ValidData()  {  string errorMessage;  // Проверяем ввод правильных данных для каждого поля  bool isValid = Validator.IsValidName("Иван", out errorMessage);  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidName("Иванов", out errorMessage); // Фамилия  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidName("Иванович", out errorMessage); // Отчество  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPhone("78965431234", out errorMessage); // Телефон  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidEmail("abcd@mail.ru", out errorMessage); // Электронная почта  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  string validDate = "15.03.2020";  isValid = Validator.IsValidBirthDate(validDate, out errorMessage);  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPlace("г. Тула", out errorMessage); // Место рождения  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidAddress("г. Тула, ул. Ленина, д. 15, кв. 15", out errorMessage); // Адрес регистрации  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPassportSeries("1234", out errorMessage); // Серия паспорта  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPassportNumber("123456", out errorMessage); // Номер паспорта  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  validDate = "14.10.2022";  isValid = Validator.IsValidDateOfIssue(validDate, out errorMessage);  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidIssuedBy("МВД", out errorMessage); // Кем выдан  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidEducationInstitution("МГУ", out errorMessage); // Учебное заведение  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  // Списки  isValid = Validator.IsValidDepartment("Технический", out errorMessage); // Отдел  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidEducationLevel("Высшее образование (бакалавриат)", out errorMessage); // Уровень  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidCitizenship("Россия", out errorMessage); // Гражданство  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  byte[] photoBytes = File.ReadAllBytes(@"C:\PDepartment\Media\face.jpg");  // Сохраняем данные в БД  var personalCard = new Personal\_card  {  Name = "Иван",  Surname = "Иванов",  Patronymic = "Иванович",  Date\_of\_birth = DateTime.Parse("15.03.2020"),  Email = "abcd@mail.ru",  Telephone = "78965431234",  Birthplace = "г. Тула",  Registration\_address = "г. Тула, ул. Ленина, д. 15, кв. 15",  Series\_and\_number = "123456",  Date\_of\_issue = DateTime.Parse("14.10.2022"),  Issued\_by\_whom = "МВД",  EducationInstitution = "МГУ",  Id\_post = 4,  Id\_education = 1,  Id\_citizenship = 1,  Children = true,  Military\_service = true,  Photo = photoBytes  };  dbModel.Personal\_card.Add(personalCard);  dbModel.SaveChanges();  // Проверка, что запись была добавлена  var addedRecord = dbModel.Personal\_card.FirstOrDefault(x => x.Name == "Иван" && x.Surname == "Иванов");  Assert.IsNotNull(addedRecord);  Assert.AreEqual("Иван", addedRecord.Name);  Assert.AreEqual("Иванов", addedRecord.Surname);  }  // Тест 2 - Ввод строчной буквы в поле «Имя»  [TestMethod]  public void Test\_InvalidName\_Lowercase()  {  string errorMessage;  bool isValid = Validator.IsValidName("иван", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Имя»", errorMessage);  }  // Тест 3 - Ввод в поле «Фамилия» чисел  [TestMethod]  public void Test\_InvalidSurname\_WithNumbers()  {  string errorMessage;  bool isValid = Validator.IsValidSurName("Иванов123", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Фамилия»", errorMessage);  }  // Тест 4 - Ввод специального символа в поле «Отчество»  [TestMethod]  public void Test\_InvalidPatronymic\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidPatronymic("Иванович@", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Отчество»", errorMessage);  }  // Тест 5 - Ввод номера с буквой в поле «Телефон»  [TestMethod]  public void Test\_InvalidPhone\_WithLetter()  {  string errorMessage;  bool isValid = Validator.IsValidPhone("78965431234a", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Телефонный номер»", errorMessage);  }  // Тест 6 - Ввод значения поля «Электронная почта» без знака @  [TestMethod]  public void Test\_InvalidEmail\_WithoutAtSign()  {  string errorMessage;  bool isValid = Validator.IsValidEmail("abcdmail.ru", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Электронная почта»", errorMessage);  }  // Тест 7 - Выбор даты в поле «Дата рождения» больше системной  [TestMethod]  public void Test\_InvalidBirthDate\_FutureDate()  {  string errorMessage;  DateTime invalidDate = DateTime.Now.AddYears(1);  string invalidDateString = invalidDate.ToString("dd.MM.yyyy"); // Преобразуем в строку  bool isValid = Validator.IsValidBirthDate(invalidDateString, out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Дата рождения»", errorMessage);  }  // Тест 8 - Ввод значения в поле «Место рождения» с прописной буквы  [TestMethod]  public void Test\_InvalidPlace\_WithLowercase()  {  string errorMessage;  bool isValid = Validator.IsValidPlace("тула", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Место рождения»", errorMessage);  }  // Тест 9 - Ввод чисел в поле «Адрес регистрации»  [TestMethod]  public void Test\_InvalidAddress\_WithNumbers()  {  string errorMessage;  bool isValid = Validator.IsValidAddress("12345", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Адрес регистрации»", errorMessage);  }  // Тест 10 - Ввод буквы в поле «Серия»  [TestMethod]  public void Test\_InvalidPassportSeries\_WithLetter()  {  string errorMessage;  bool isValid = Validator.IsValidPassportSeries("123a", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Серия паспорта»", errorMessage);  }  // Тест 11 - Ввод специального символа в поле «Номер»  [TestMethod]  public void Test\_InvalidPassportNumber\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidPassportNumber("12345@", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Номер паспорта»", errorMessage);  }  // Тест 12 - Ввод даты с специальным символом в поле «Дата выдачи»  [TestMethod]  public void Test\_InvalidDateOfIssue\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidDateOfIssue("14,10.2024", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Дата выдачи»", errorMessage);  }  // Тест 13 - Ввод чисел в поле «Кем выдан»  [TestMethod]  public void Test\_InvalidIssuedBy\_WithNumbers()  {  string errorMessage;  bool isValid = Validator.IsValidIssuedBy("12345", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Кем выдан»", errorMessage);  }  // Тест 14 - Ввод значения со специальным символом в поле «Учебное заведение»  [TestMethod]  public void Test\_InvalidEducationInstitution\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidEducationInstitution("МГУ@", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Учебное заведение»", errorMessage);  }  // Тест 15 - Поле со списком «Отдел» пустое  [TestMethod]  public void Test\_EmptyDepartment()  {  string errorMessage;  bool isValid = Validator.IsValidDepartment("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Отдел» пустое", errorMessage);  }  // Тест 16 - Поле со списком «Должность» пустое  [TestMethod]  public void Test\_EmptyPost()  {  string errorMessage;  bool isValid = Validator.IsValidPost("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Должность» пустое", errorMessage);  }  // Тест 17 - Поле со списком «Уровень» пустое  [TestMethod]  public void Test\_EmptyEducationLevel()  {  string errorMessage;  bool isValid = Validator.IsValidEducationLevel("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Уровень» пустое", errorMessage);  }  // Тест 18 - Поле со списком «Гражданство» пустое  [TestMethod]  public void Test\_EmptyCitizenship()  {  string errorMessage;  bool isValid = Validator.IsValidCitizenship("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Гражданство» пустое", errorMessage);  }  // Тест 19 - Поле "Аватар" пустое  [TestMethod]  public void Test\_InvalidPhoto\_FileNotFound()  {  string errorMessage;  bool isValid = Validator.IsValidPhoto("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Выберите значение в поле «Аватар»", errorMessage);  }  }  } |

Модульное тестирование играет ключевую роль в разработке, выявляя ошибки на ранних стадиях и предотвращая их распространение. Все проведённые тесты показали, что система корректно обрабатывает как правильные, так и ошибочные данные, а также корректно выводит сообщения об ошибках. Это подтверждает стабильность и надёжность системы в реальных условиях эксплуатации.

Таким образом, модульное тестирование существенно повышает качество программного обеспечения, помогая заблаговременно обнаруживать и исправлять ошибки, что обеспечивает создание более надёжных систем.

**ЗАКЛЮЧЕНИЕ**

В рамках выполнения курсового проекта была успешно разработана информационная система для отдела кадров, которая решает задачи автоматизации учета и обработки данных сотрудников. В ходе работы был проведен анализ предметной области и требований к системе, что позволило определить ключевые функциональные потребности и направления для дальнейшей разработки.

Процесс разработки включал модификацию диаграмм вариантов использования и классов, что позволило четко структурировать логику взаимодействия компонентов системы. Структура программного проекта и его архитектура были определены с учетом оптимизации работы системы и обеспечения масштабируемости.

В ходе разработки был создан полноценный функционал базы данных, обеспечивающий надежное хранение информации о сотрудниках, должностях и других кадровых данных. Также был реализован интуитивно понятный пользовательский интерфейс, что значительно упрощает взаимодействие с системой и повышает ее удобство для конечных пользователей.

Разработаны классы сущностей и управляющие классы, что позволило обеспечить стабильную и эффективную работу системы. Важной частью работы было проведение тестирования, которое позволило выявить и устранить возможные ошибки и недостатки, улучшив общую надежность приложения.

Таким образом, в процессе разработки информационной системы «Отдел кадров» были достигнуты все поставленные цели и задачи, включая анализ требований, проектирование, реализацию базы данных и интерфейса, а также проведение тестирования, что обеспечивает эффективное функционирование системы в реальных условиях эксплуатации.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. ГОСТ 2.301-68: Форматы ЕСКД. – М.: Изд-во cтандартов, 1997 г.
2. ГОСТ 19.101-77 Единая система программной документации. Виды программ и программных документов. – М.: Изд-во cтандартов, 1977 г.
3. ГОСТ 19.102-77 Единая система программной документации. Стадии разработки. – М.:Изд-во cтандартов, 1977 г.
4. ГОСТ 19.104-78 Единая система программной документации. Основные надписи. – М.:Изд-во cтандартов, 1978 г.
5. ГОСТ 19.105-78 Единая система программной документации. Общие требования к программным документам. – М.:Изд-во cтандартов, 1978 г.
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. – М.:Изд-во cтандартов, 1978 г.
7. ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению – М.:Изд-во cтандартов, 1978 г.
8. ГОСТ 19.202-78 Единая система программной документации. Спецификация. Требования к содержанию и оформлению. – М.:Изд-во cтандартов, 1978 г.
9. ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. – М.:Изд-во cтандартов, 1979 г.
10. ГОСТ 19.402-78 Единая система программной документации. Описание программы. – М.:Изд-во cтандартов, 1978 г.
11. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. – М.:Изд-во cтандартов, 1979 г.
12. ГОСТ 34.601-90 Автоматизированные системы. Стадии создания. – М.:Изд-во cтандартов, 1990 г.
13. ГОСТ Р ИСО/МЭК 9126-93 Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. – М.:Изд-во cтандартов, 2003 г.
14. ГОСТ Р ИСО/МЭК ТО 9294-93 Информационная технология. Руководство по управлению документированием программного обеспечения. – М.:Изд-во cтандартов, 2003 г.
15. Федорова, Г.Н. Участие в интеграции программных модулей: учебное пособие для студентов учреждений сред. проф. образования / Г.Н. Федорова. – М.: Издательский центр «Академия», 2021. – 304 с.
16. Федорова, Г.Н. Разработка программных модулей программного обеспечения для компьютерных систем: учебник для студентов учреждений сред. проф. образования / Г.Н. Федорова. – М.: Издательский центр «Академия», 2021. – 336 с.
17. Единое окно доступа к образовательным ресурсам [Электронный ресурс]. – режим доступа: http://real.tepkom.ru/Real\_OM-CM\_A.asp.
18. Электронный образовательный ресурс по программированию в C# [Электронный ресурс]. – Режим доступа: https://metanit.com, свободный.
19. Интернет-университет информационных технологий [Электронный ресурс]. – Режим доступа: www.intuit.ru.
20. CIT-Forum: Центр информационных технологий: материалы сайта [Электронный ресурс]. - Режим доступа: http://citforum.ru/, свободный.
21. CodeNet - все для программиста [Электронный ресурс]. – Режим доступа: http://www.codenet.ru/, свободный.
22. Библиотека учебных курсов Microsoft [Электронный ресурс]. – Режим доступа: http://msdn.microsoft.com/ru-ru/gg638594, свободный.
23. Библиотека учебных курсов Microsoft. Документация [Электронный ресурс].– Режим доступа: http://msdn.microsoft.com/library/, свободный.
24. ГОСТ Эксперт: единая база ГОСТов РФ. Документация на разработку программного обеспечения и системная документация [Электронный ресурс]. - Режим доступа: http://gostexpert.ru/oks/35/80, свободный.
25. Документирование программных средств [Электронный ресурс]//Федеральный центр информационно-образовательных ресурсов. – Режим доступа: http://fcior.edu.ru/card/29134/ dokumentirovanie-programmnyh-sredstv.html, свободный.
26. Единая система программной документации [Электронный ресурс]. – Режим доступа: http://prog-cpp.ru/espd/, свободный.

**Дополнительные источники**

1. Васильев. Р. Стратегическое управление информационными системами. Учебник / Р. Васильев, Г. Калянов, Г. Левочкина, О. Лукинова.- Бином. Лаборатория знаний, Интернет-университет информационных технологи, 2021. – 512 с. ISBN 978-5-9963-0350-2
2. Винниченко И. В. Автоматизация процессов тестирования. – СПб.: Питер, 2018. – 203 с.
3. Гагарина, Л.Г., Виснадул, Б.Д., Игошин, А.В. Основы технологии разработки программных продуктов: учеб. пособие. – М.: ФОРУМ: ИНФРА-М, 2021 г.

**ПРИЛОЖЕНИЕ А.**

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

**СОДЕРЖАНИЕ**

1. Введение…………………………………………………………………………75
   1. Область применения……………………………………………………………75
   2. Краткое описание возможностей………………………………………………75
   3. Уровень подготовки пользователя…………………………………………......75
   4. Перечень эксплуатационной документации…………………………………..75
2. Назначение и условия применения…………………………………………....76
   1. Виды деятельности, функции………………………………………………….76
   2. Программные и аппаратные требования к системе…………………………..76
3. Подготовка к работе………………………………………………………….....77
   1. Состав дистрибутива…………………………………………………………....77
   2. Запуск системы……………………………………………………………….....77
   3. Проверка работоспособности системы………………………………………..78

4 Описание операций……………………………………………………………..80

4.1 Описание операций «Инспектор»……………………………………………..80

4.1.1 Описание операции 1: Создание личной карточки сотрудника и приказа о приёме на работу……………………………………………………………………………...80

4.1.2 Описание операции 2: Создание отчёта……………………………………….86

4.1.3 Описание операции 3: Создание приказа……………………………………..87

4.1.4 Описание операции 4: Изменение данных о сотруднике…………………….89

4.1.5 Описание операции 5: Удаление сотрудника…………………………………91

4.1.6 Описание операции 6: Назначение награды…………………………………..92

4.1.7 Описание операции 7: Добавление записи в трудовую книжку……………..94

4.2 Описание операции роли «Администратор»………………………………….96

4.2.1 Описание операции 1: Добавление записей в базу данных………………….96

4.2.2 Наименование операции: Добавление логина, пароля и изменении роли пользователей…………………………………………………………………………………...98

* 1. Наименование операции: Выход из системы…………………………………99

1. Аварийные ситуации, восстановление базы данных……….………………..100

6 Рекомендации по освоению ……………………………………….………….101

**1** **Введение**

**1.1 Область применения**

Данное руководство предназначено для пользователей автоматизированной информационной системы «Отдел кадров», созданной для упрощения и оптимизации работы с персоналом в организации. Система предназначена для использования инспекторами отдела кадров, администраторами системы.

**1.2 Краткое описание возможностей**

Система позволяет автоматизировать весь спектр задач по управлению персоналом: от ведения личных карточек сотрудников и учета их трудовой деятельности до формирования отчетности по кадрам, а также позволяет составлять приказы.

**1.3 Уровень подготовки пользователя**

Руководство предполагает, что пользователь имеет базовые навыки работы с компьютером и знаком с основными принципами работы в информационных системах. Пользователь должен иметь опыт работы с Excel.

**1.4 Перечень эксплуатационной документации**

В дополнение к данному руководству пользователя, система сопровождается технической документацией, включая описание архитектуры системы и инструкции по установке и настройке.

**2 Назначение и условия применения**

**2.1 Виды деятельности, функции**

Информационная система охватывает следующие функции:

* Ведение и управление личными карточками сотрудников;
* Добавление записей в трудовую книжку;
* Регистрация и учет трудовых договоров, приказов о приеме и увольнении, переводах;
* Формирование отчетности по кадрам.

**2.2 Программные и аппаратные требования к системе**

Для корректной работы системы необходим компьютер с операционной системой Windows 7 и выше, не менее 4 ГБ оперативной памяти и доступ к локальной сети предприятия.

**3 Подготовка к работе**

**3.1 Состав дистрибутива**

Дистрибутив системы включает инсталляционные файлы, необходимые библиотеки и документацию. Для работы с ИС «Отдел кадров» необходимо следующее программное обеспечение: Excel.

Дистрибутивным носителем данных является флеш-накопитель.

На нем хранятся:

* Установщик ИС.
* Руководство пользователя.

**3.2 Запуск системы**

Для запуска системы необходимо выполнить установку ИС «Отдел кадров».

* В программе MS SQL Server создать базу данных «HumanResourcesDepartment» (рисунок А.1);

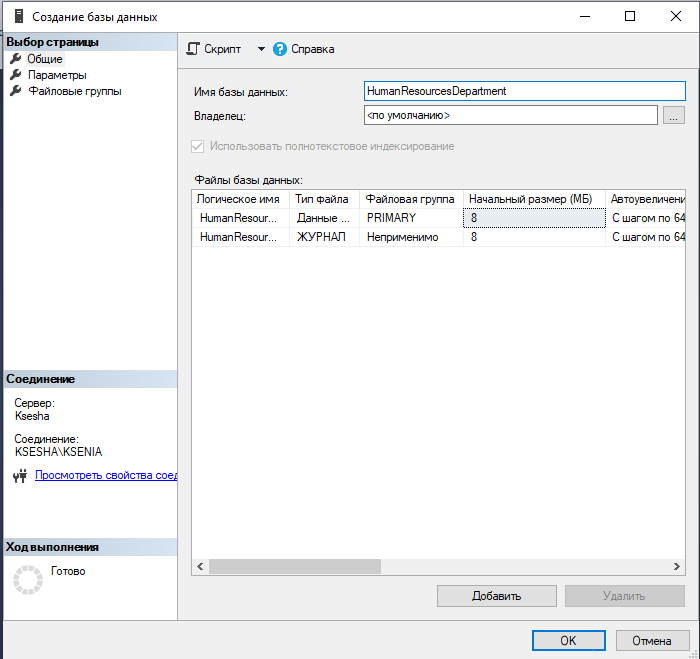


Рисунок А.1 – Создание базы данных

* Запустить скрипт на выполнение «HumanResourcesDepartmentScript.sql» с базой данных в MS SQL Server (рисунок А.2).



Рисунок А.2 – Скрипт «HumanResourcesDepartmentScript.sql»

**3.3 Проверка работоспособности системы**

* На рабочем месте пользователя открыть папку «Отдел кадров» и запустить ярлык «PersonalDepartment».
* После чего запуститься информационная система.
* Далее пользователь должен авторизоваться в системе, введя свой логин и пароль: инспектор, администратор (рисунок А.3).
* После выполнения всех этих действий, пользователь может начать работать с системой. Перед ним откроется главное окно, из которого можно открыть другие (рисунок А.3).

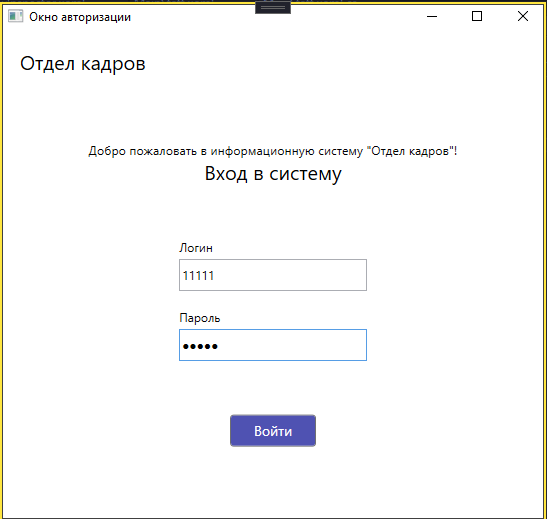


Рисунок А.3 – Авторизация пользователя

* После выполнения всех этих действий, пользователь может начать работать с системой. Перед ним открывается главная форма, позволяющая перемещаться по всем формам приложения (рисунок А.4).

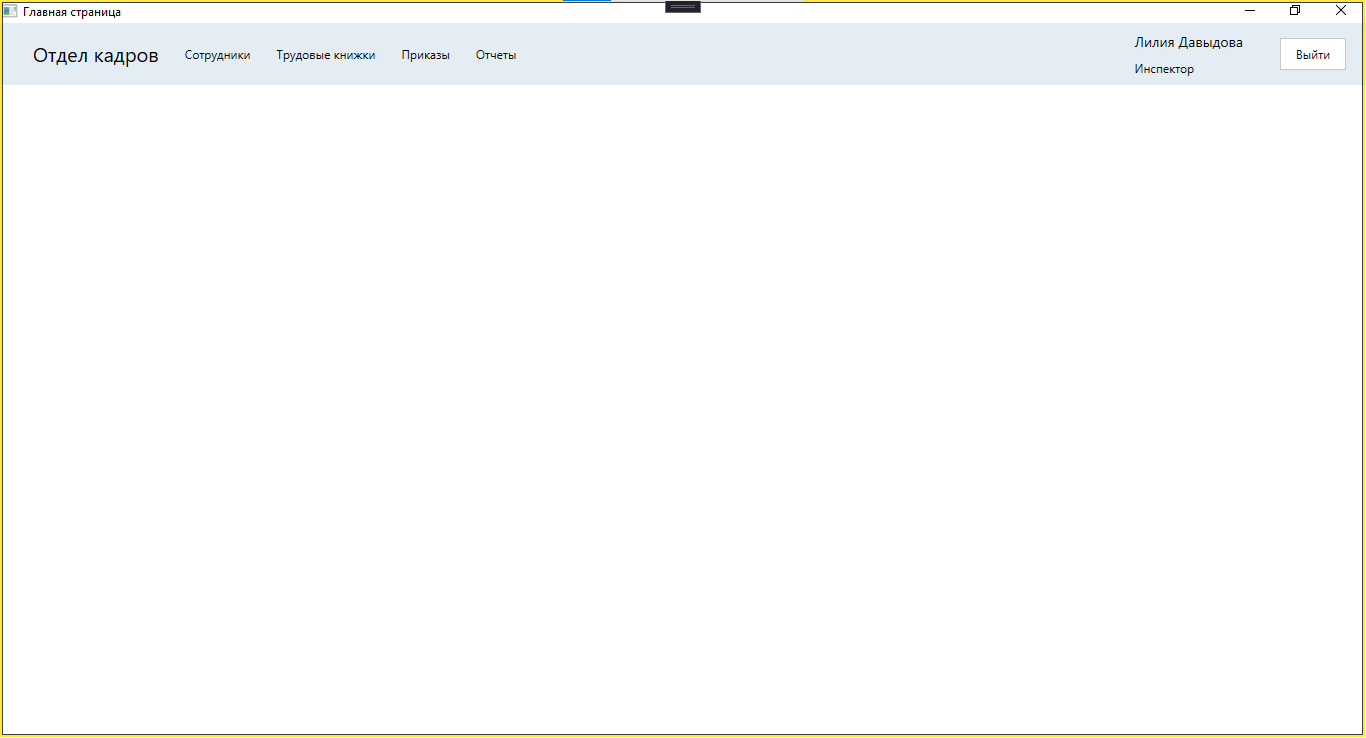


Рисунок А.4 – Главная форма

**4 Описание операций**

**4.1 Описание операций «Инспектор»**

ИС выполняет операции, приведенные в таблице А.1.

Таблица А.1 – Операции, выполняемые в ИС, для роли «Инспектор».

|  |  |
| --- | --- |
| **Задачи** | **Описание** |
| Создание личной карточки и приказа о приеме. | Инспектор добавляет запись о сотруднике и приказ о приеме. |
| Создание отчета. | Инспектор создает отчёт. |
| Создание приказа. | Инспектор создает приказ. |
| Изменение данных о сотруднике. | Инспектор просматривает, изменяет или удаляет запись о сотруднике. |
| Удаление сотрудника. | Инспектор удаляет сотрудника из системы. |
| Назначение награды. | Инспектор назначает награду сотруднику. |
| Добавление записи в трудовую книжку. | Инспектор добавляет запись в трудовую книжку. |

**4.1.1 Описание операции 1: Создание личной карточки сотрудника и приказа о приёме на работу**

Условия выполнения операции: Доступ к операции имеется у пользователя с правами инспектора отдела кадров. Операция может быть выполнена в любое время при работающей системе.

Подготовительные действия: перед началом работы с личными карточками сотрудников необходимо убедиться в правильности и актуальности исходных данных. Подготовьте все необходимые документы сотрудника, такие как паспорт, документы об образовании.

Основные действия:

1 Авторизуйтесь в системе под своей учетной записью (рисунок А.5).

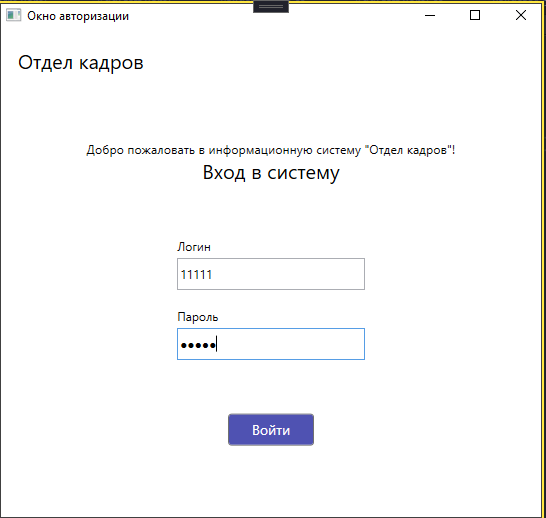


Рисунок А.5 – Форма авторизации

При неверном логине или пароле система выведет сообщение «Пользователь не найден!» (рисунок А.6).

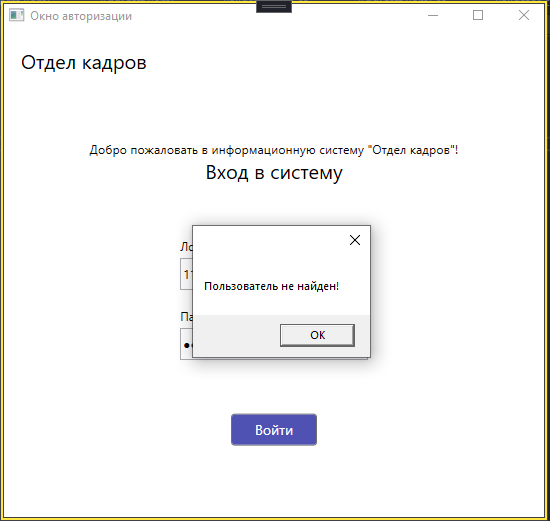


Рисунок А.6 – Сообщение о неверном вводе

1. После входа откроется главная форма (рисунок А.7).

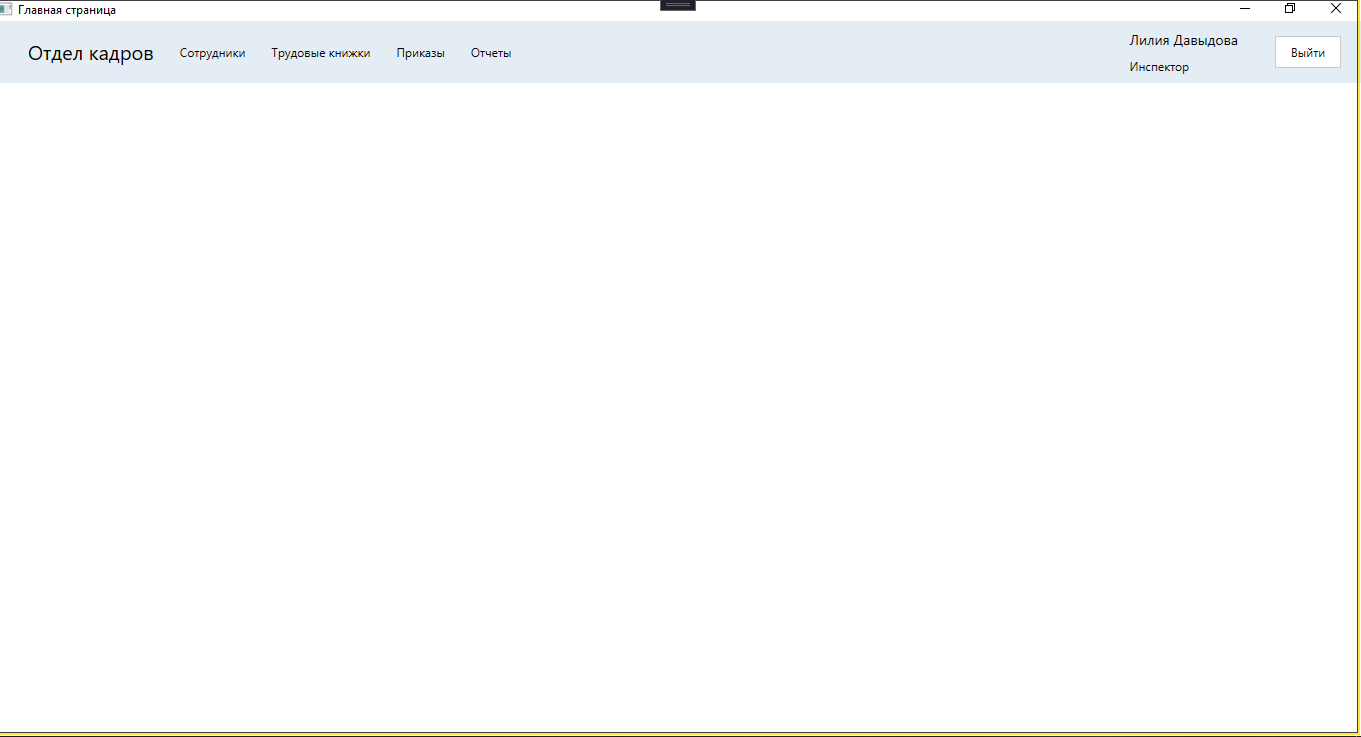


Рисунок А.7 – Главная форма

1. Перейти по кнопке «Сотрудники». Открылась форма (рисунок А.8).

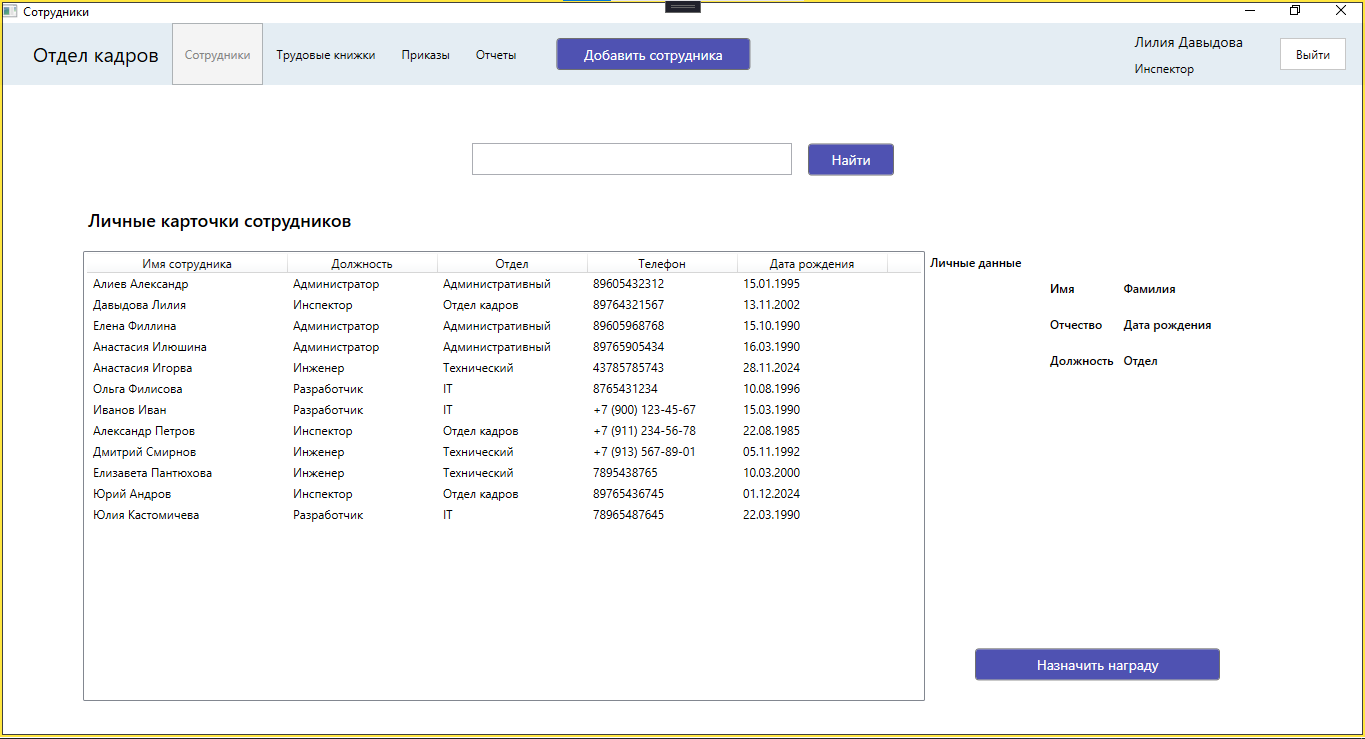


Рисунок А.8 – Форма «Сотрудники»

4 Нажать на кнопку «Добавить сотрудника». Откроется форма (рисунок А.9).

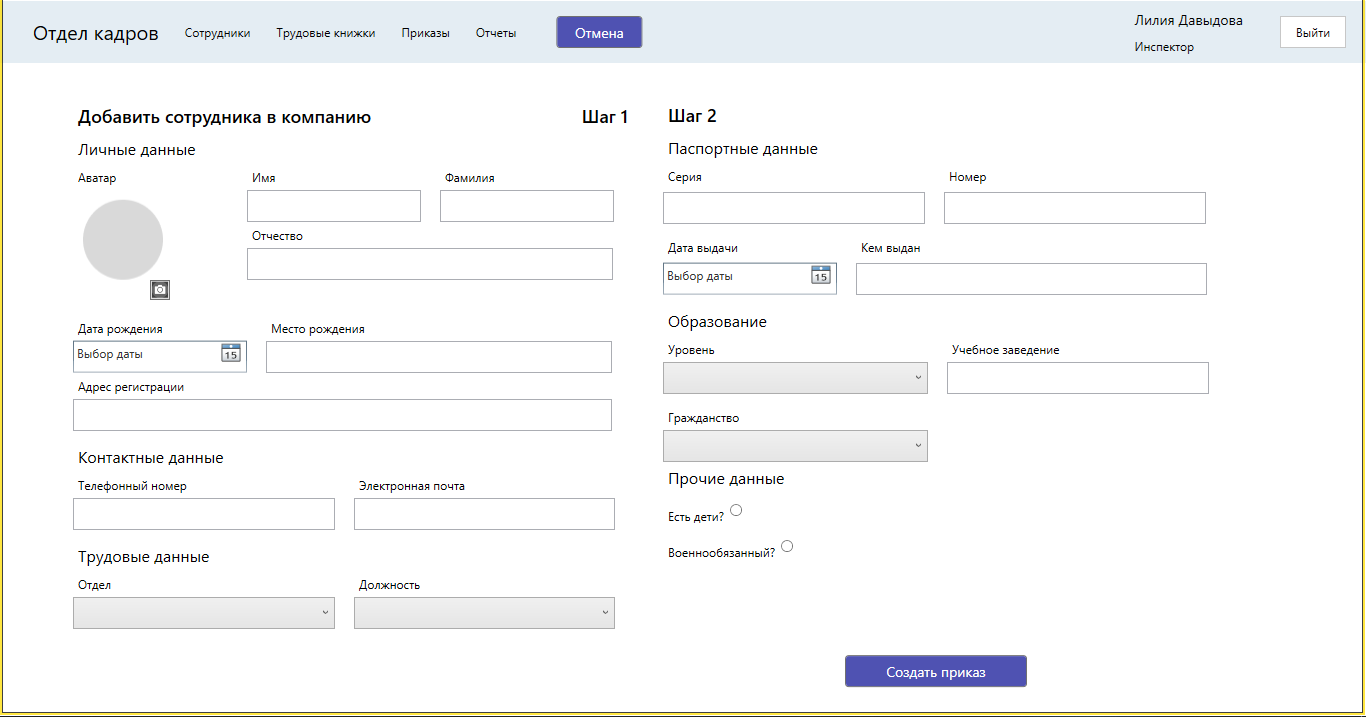


Рисунок А.9 – Форма «Добавление сотрудника»

5 Если данные были введены не во все обязательные поля, то появится сообщение (рисунок А.10).

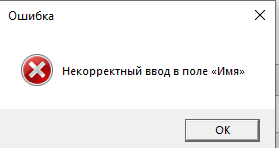


Рисунок А.10 – Уведомление о незаполненном поле

6 Введите в поля данные. Для выбора даты появится всплывающее окно (рисунок А.11).

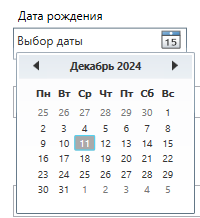


Рисунок А.11 – Всплывающее окно для выбора даты

5 Для заполнения полей «Отдел», «Должность», «Уровень», «Гражданство» выберите значение из выпадающего списка (рисунки А.12-А.15).

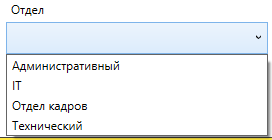


Рисунок А.12 – Выпадающий список для поля «Отдел»

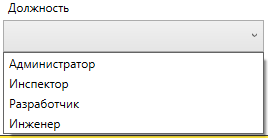


Рисунок А.13 – Выпадающий список для поля «Должность»

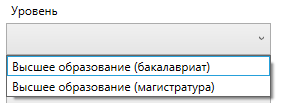


Рисунок А.14 – Выпадающий список для поля «Уровень»



Рисунок А.15 – Выпадающий список для поля «Гражданство»

6 После заполнения личной карточки нажмите на кнопку «Создать приказ». Карточка сохранится и откроется форма для создания приказа о приёме (рисунок А.16). Сотрудник и тип приказа будут выбраны автоматически.

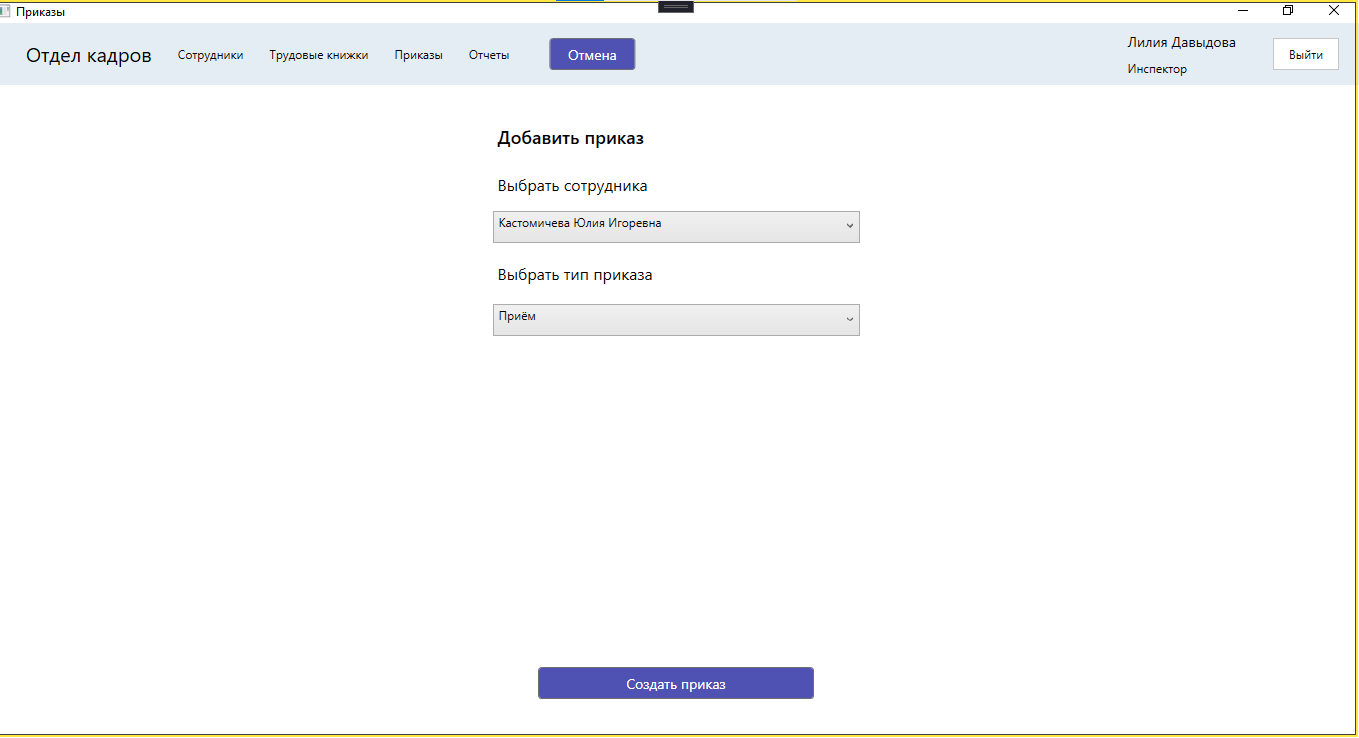


Рисунок А.16 – Создание приказа на прием сотрудника на работу

8 Выбрать место на компьютере куда сохранить приказ (рисунок А.17).

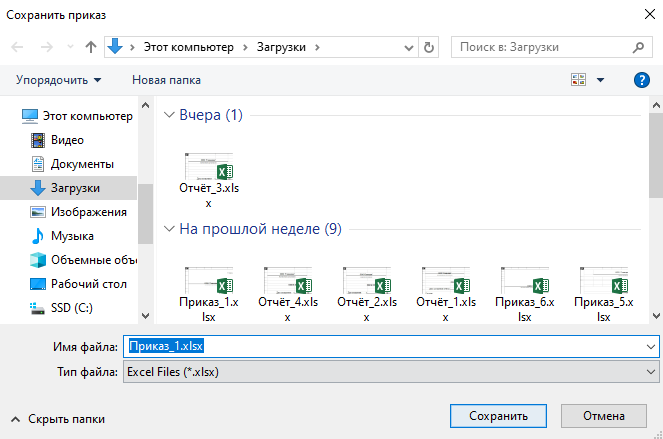


Рисунок А.17 – Окно проводника для выбора места сохранения

9 Нажмите на кнопку «Создать приказ». Появится уведомление о том, что файл сохранен (рисунок А.18).

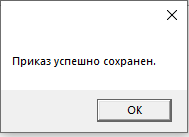


Рисунок А.18 – Сообщение об успешном сохранении

Заключительные действия: после сохранения данных убедитесь, что внесенная информация отображается корректно в общем списке сотрудников и в приказе. В случае необходимости внесите дополнительные изменения.

Ресурсы, расходуемые на операцию: время на выполнение операции, зависит от объема вносимых данных, в среднем занимает от 5 до 15 минут. Для работы требуется ПК.

**4.1.2 Описание операции 2: Создание отчёта**

Условия выполнения операции: доступ к данной операции предоставляется пользователям с правами инспектора отдела кадров. Для выполнения операции необходимо наличие актуальной базы данных сотрудников.

Подготовительные действия: убедитесь, что все данные в системе актуальны и корректны. Определите параметры отчета, который требуется сформировать (период, отдел, тип отчета).

Основные действия:

1 В главном меню выберите пункт «Отчеты».

2 Укажите необходимые параметры для формирования отчета: тип отчета, период (рисунок А.19). В зависимости от типа отчёта могут появляется дополнительные поля или наоборот.

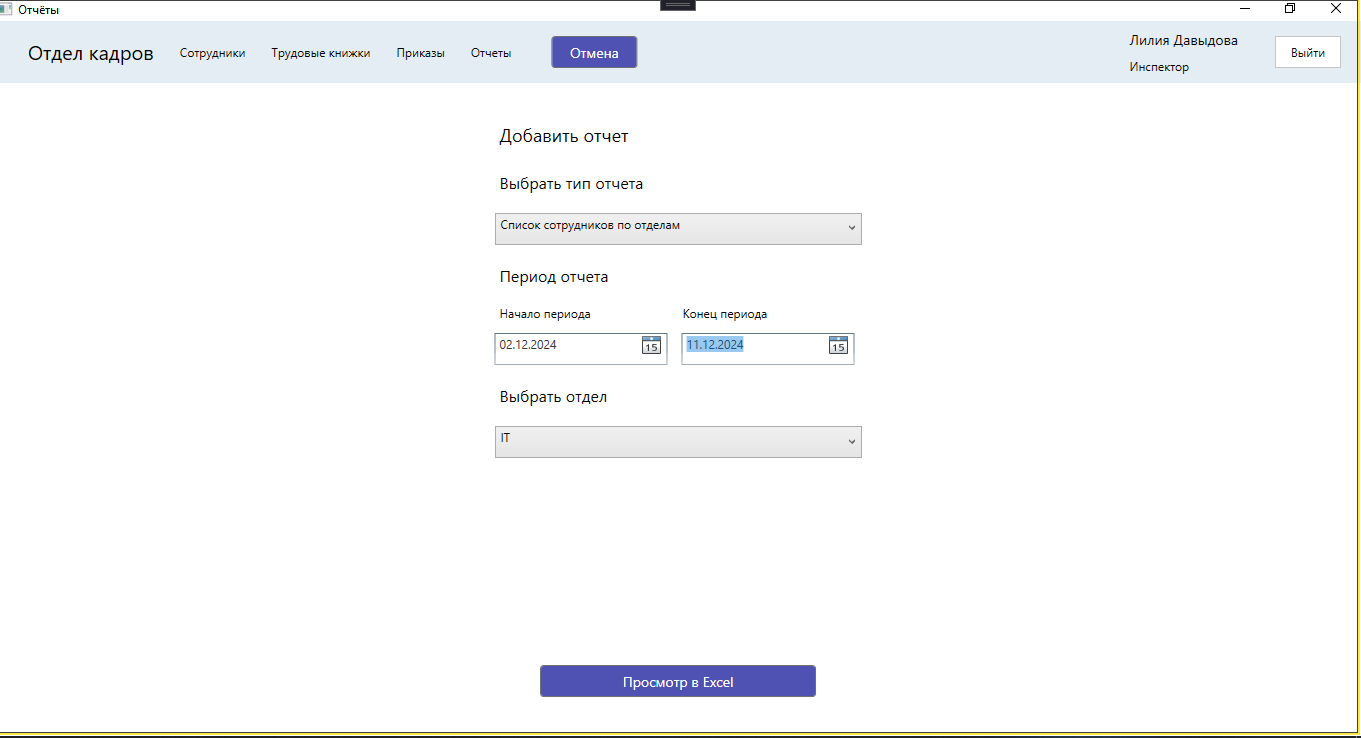


Рисунок А.19 – Форма «Отчёты»

3 Нажмите кнопку «Просмотр в Excel».

4 Проанализируйте полученные результаты в Excel.

Заключительные действия: Проверьте корректность сформированного отчета. При необходимости выполните операцию повторно с измененными параметрами.

Ресурсы, расходуемые на операцию: Время на формирование отчета может варьироваться от нескольких секунд до нескольких минут, в зависимости от объема запрашиваемых данных и загруженности системы.

**4.1.3 Описание операции 3: Создание приказа**

Условия выполнения операции: Доступ к этой операции предоставляется пользователям с правами доступа инспектора отдела кадров. Операция возможна при условии наличия активных данных о сотрудниках, для которых оформляется приказ.

Подготовительные действия: Убедитесь в наличии всех необходимых данных о сотруднике и событии (прием на работу, увольнение, перевод), для которого оформляется приказ.

Основные действия:

1 В главном меню системы выберите раздел «Приказы».

2 В открывшейся форме заполните все необходимые поля, включая информацию о сотруднике и тип приказа (рисунок А.20). В зависимости от типа приказа могут появляться дополнительные поля и наоборот.

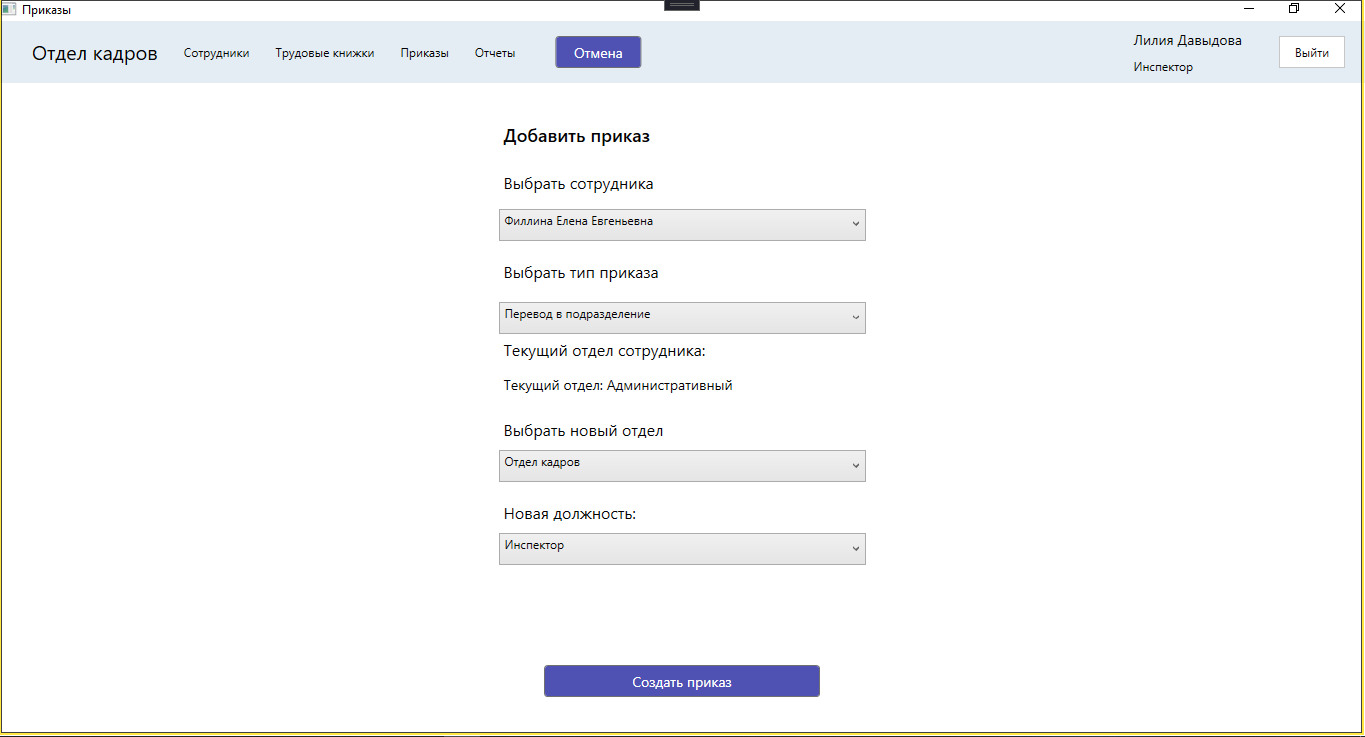


Рисунок А.20 – Форма «Приказы»

1. Выберите место на компьютере куда сохранить приказ (рисунок А.21).

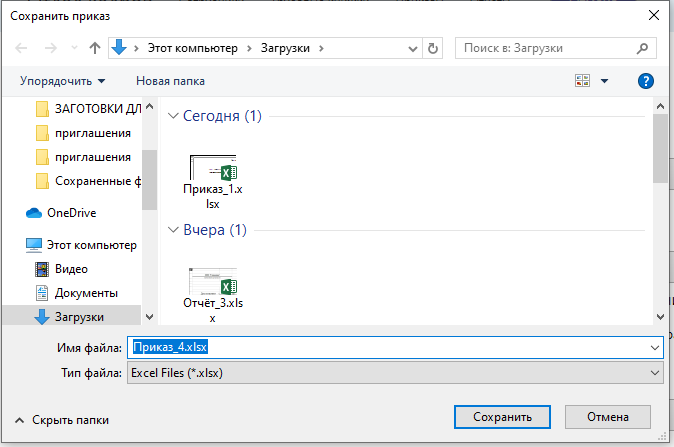


Рисунок А.21 – Окно проводника для выбора места сохранения

5 Проверьте корректность введенной информации и нажмите кнопку «Создать приказ» для сохранения и распечатки документа. Появится сообщение, что сохранен файл с приказом (рисунок А.22).

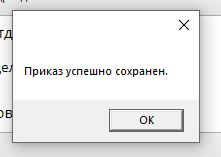


Рисунок А.22 – Уведомление о сохранении файла

Заключительные действия: убедитесь, что приказ корректно отображается в системе и доступен для просмотра и печати. передайте приказ на подпись руководителю, если это необходимо.

Ресурсы, расходуемые на операцию: примерное время на выполнение операции — от 5 до 10 минут, в зависимости от сложности приказа и скорости ввода данных.

**4.1.4 Описание операции 4: Изменение данных о сотруднике**

Условия выполнения операции: доступно для пользователей с правами инспектора отдела кадров. Операция может быть выполнена в случае необходимости обновления информации о сотруднике.

Подготовительные действия: подготовьте актуальные данные о сотруднике, которые требуется внести в личную карточку.

Основные действия:

1 Откройте раздел «Сотрудники» (рисунок А.23).

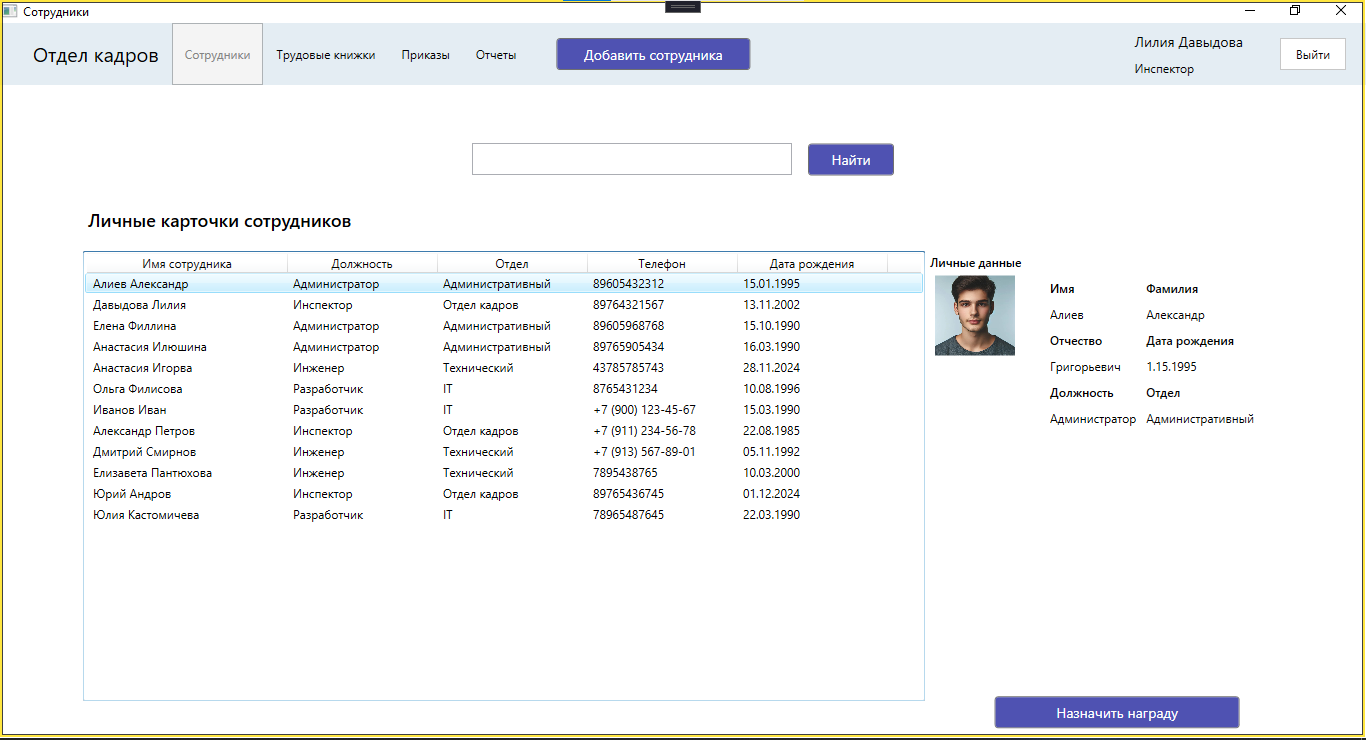


Рисунок А.23 – Форма «Сотрудники»

2 Нажмите 2 раза по строчке сотрудника чьи данные нужно отредактировать.

3 Внесите изменения на открывшейся форме (рисунок А.24).

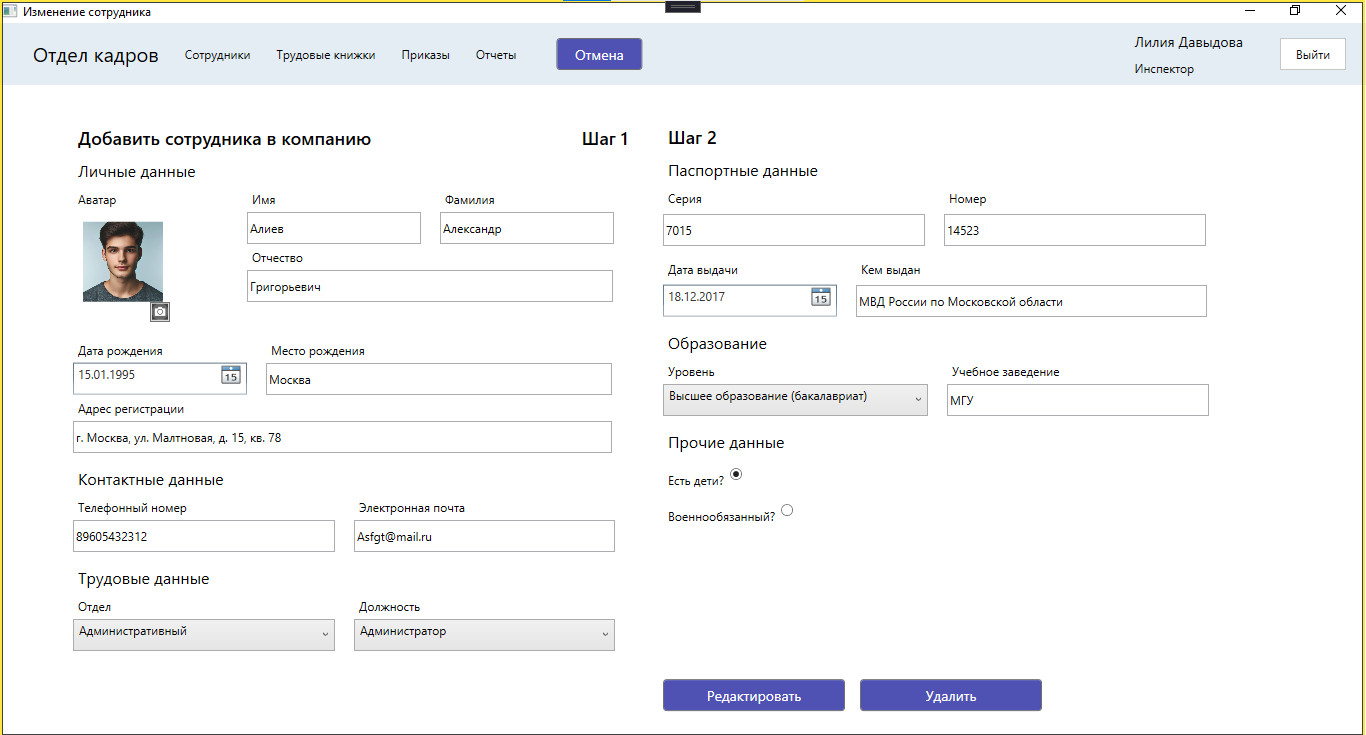


Рисунок А.24 – Форма для редактирования данных

5 Нажмите кнопку «Редактировать». Появится сообщение «Данные успешно сохранены» (рисунок А.25).

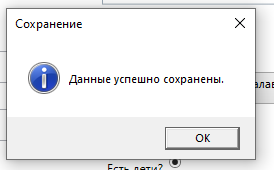


Рисунок А.25 – Уведомление об успешном редактировании

Заключительные действия: проверьте, что все изменения корректно отражены в личной карточке сотрудника.

Ресурсы, расходуемые на операцию: время на выполнение операции, зависит от объема изменений, обычно занимает от 3 до 5 минут.

**4.1.5 Описание операции 5: Удаление сотрудника**

Условия выполнения операции: доступно для пользователей с правами инспектора отдела кадров. Операция может быть выполнена в случае необходимости обновления информации о сотруднике.

Подготовительные действия: подготовьте актуальные данные о сотруднике, которые требуется внести в личную карточку.

Основные действия:

1 Откройте раздел «Сотрудники» (рисунок А.26).

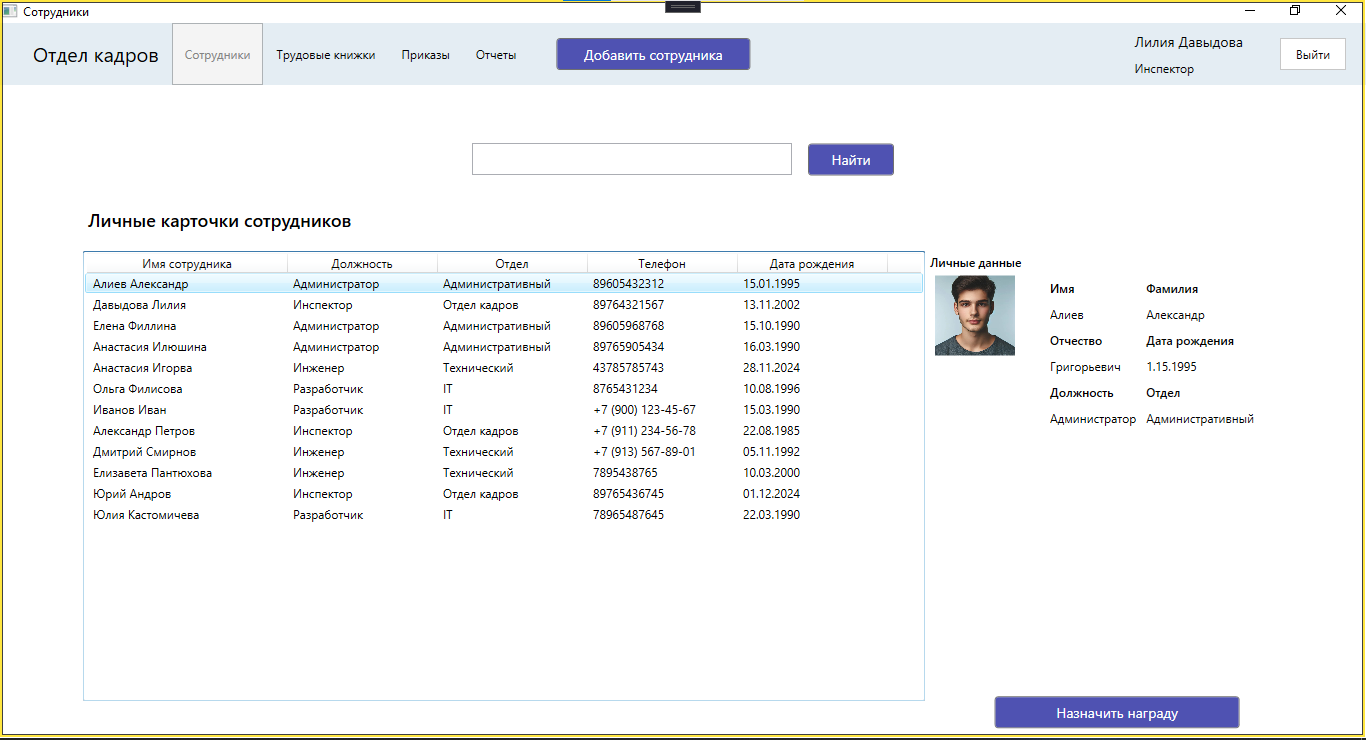


Рисунок А.26 – Форма «Сотрудники»

2 Нажмите 2 раза по строчке сотрудника чьи данные нужно отредактировать.

3 Нажмите на кнопку «Удалить» на открывшейся форме (рисунок А.27).

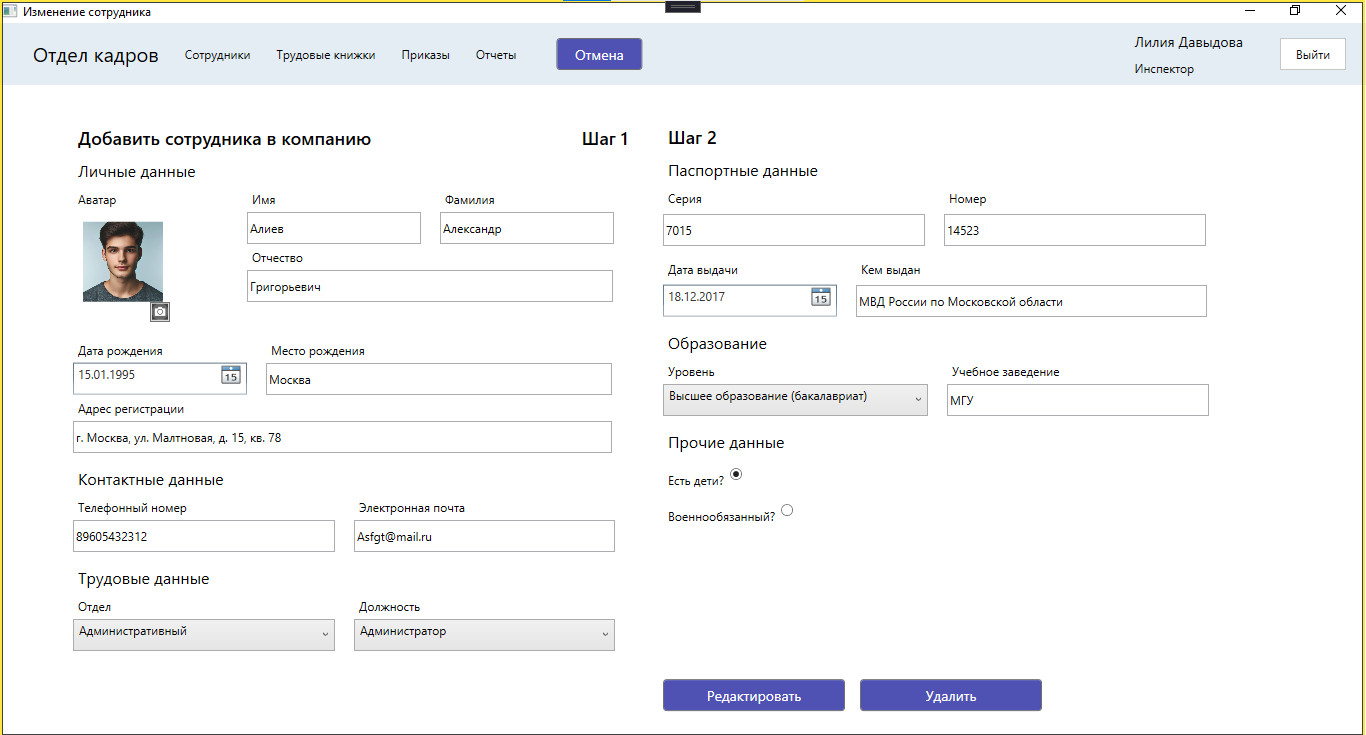


Рисунок А.27 – Форма для редактирования данных

5 Появится сообщение «Данные успешно сохранены». Возврат на форму «Сотрудники» (рисунок А.28).

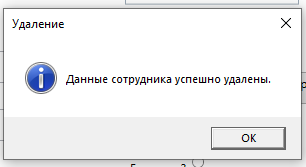


Рисунок А.28 – Уведомление об успешном редактировании

Заключительные действия: проверьте, что все изменения корректно отражены в личной карточке сотрудника.

Ресурсы, расходуемые на операцию: время на выполнение операции, зависит от объема изменений, обычно занимает от 1 до 3 минут.

**4.1.6 Описание операции 6: Назначение награды**

Условия выполнения операции: доступно для пользователей с правами инспектора отдела кадров. Операция может быть выполнена в случае необходимости обновления информации о сотруднике.

Подготовительные действия: подготовьте актуальные данные о сотруднике, которые требуется внести в личную карточку.

Основные действия:

1 Откройте раздел «Сотрудники» (рисунок А.29).

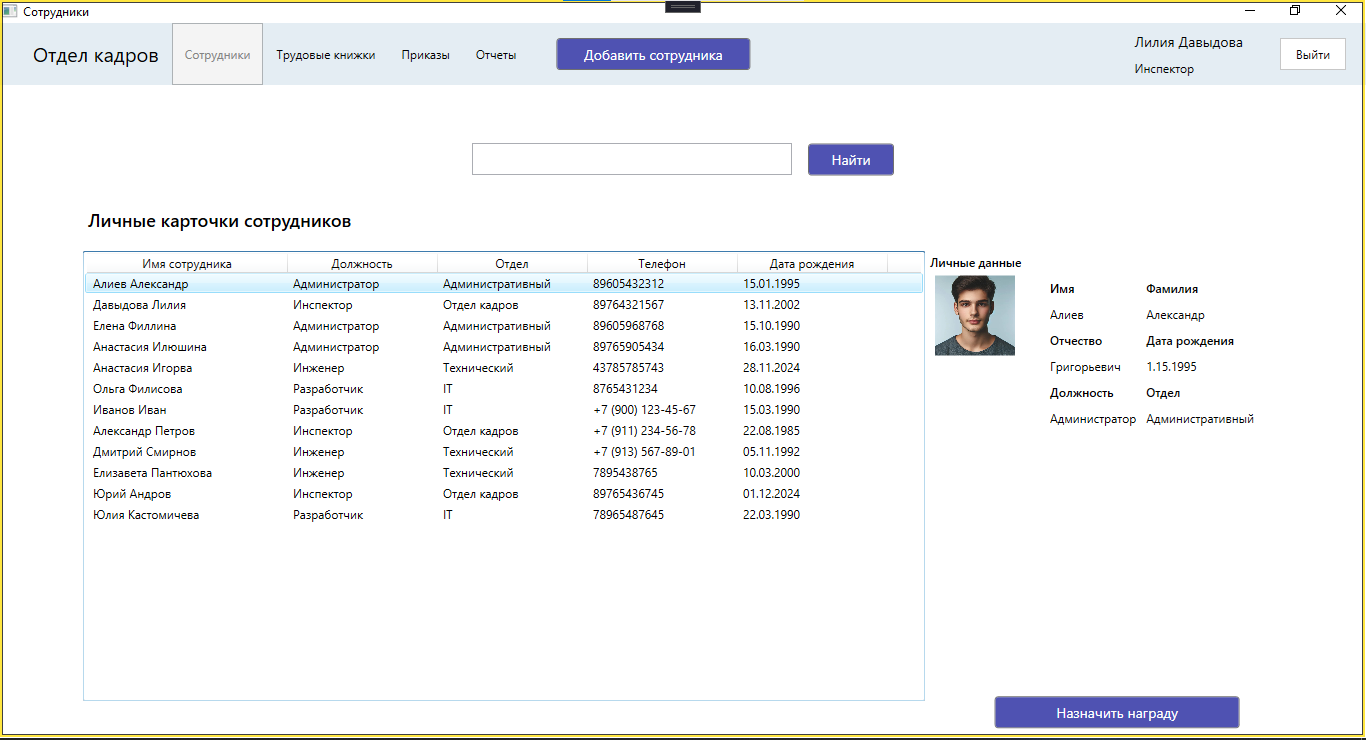


Рисунок А.29 – Форма «Сотрудники»

2 Нажмите 1 раз по строчке сотрудника чьи данные нужно добавить.

3 Нажмите на кнопку «Назначить награду».

4 Открылась форма для добавления награды. Заполните на форме Тип награды и сумма, так как ФИО уже будет заполнено автоматически. Если после выбора из поля со списком такой тип награды был, то в текстовом блоке автоматически появится значение, которое можно изменить (рисунок А.30).

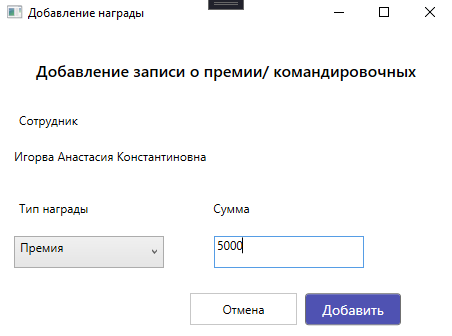


Рисунок А.30 – Форма «Добавление сотрудника»

5 Нажмите кнопку «Добавить». Появится сообщение «Запись успешно добавлена.» (рисунок А.31). Возврат на форму «Сотрудники». Если нажать на кнопку «Отмена», то произойдёт возвращение на форму «Сотрудники».

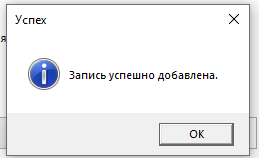


Рисунок А.21 – Сообщение об успешном добавлении

**4.1.7 Описание операции 7: Добавление записи в трудовую книжку**

Условия выполнения операции: доступно для пользователей с правами инспектора отдела кадров. Операция может быть выполнена в случае необходимости обновления информации о сотруднике.

Подготовительные действия: подготовьте актуальные данные о сотруднике, которые требуется внести в личную карточку.

Основные действия:

1 Откройте раздел «Трудовые книжки» (рисунок А.32).

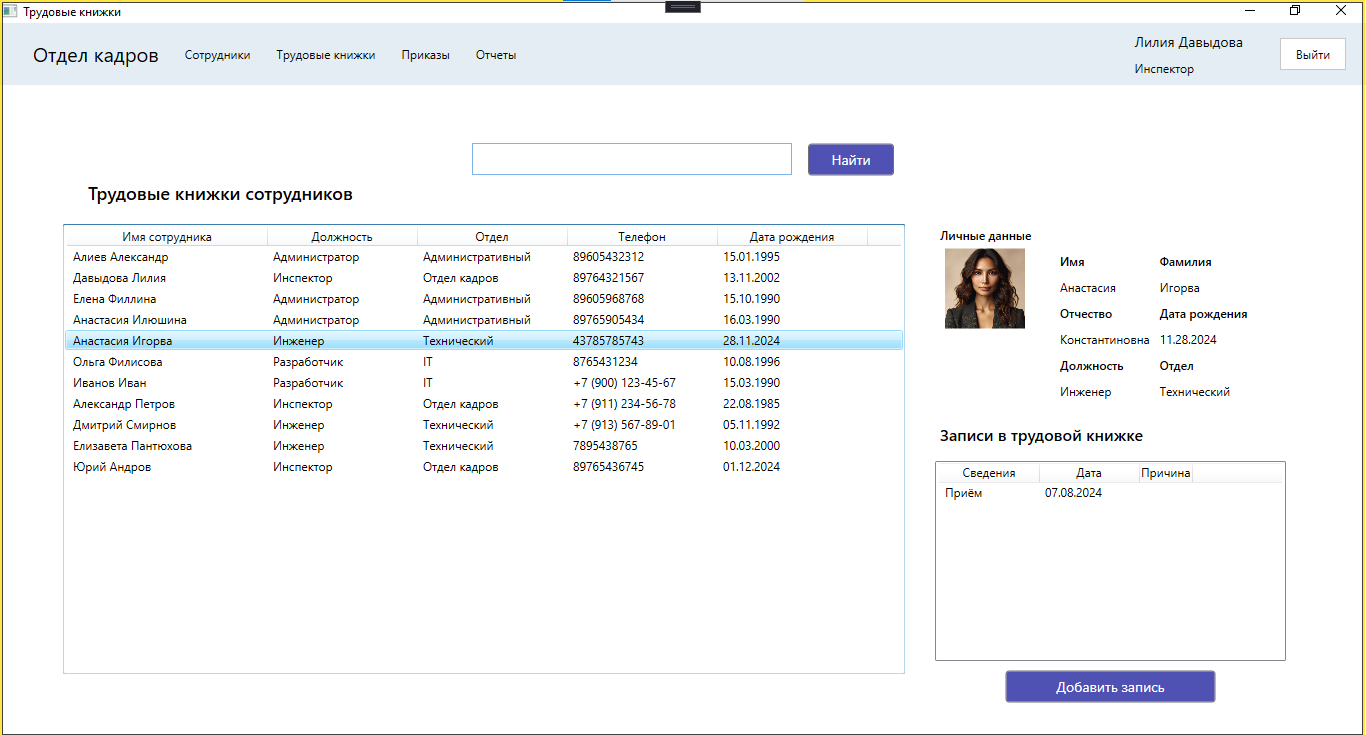


Рисунок А.32 – Форма «Сотрудники»

2 Нажмите 1 раз по строчке сотрудника чьи данные нужно добавить.

3 Запись о приёме на работу будет написано автоматически после создание личной карточки. Откроется окно «Добавление записи» (рисунок А.33). Нажмите на кнопку «Добавить».

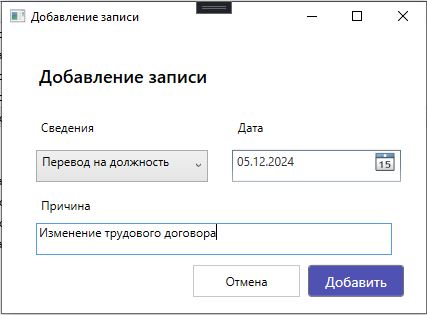


Рисунок А.33 – Окно «Добавление записи»

4 Нажмите кнопку «Добавить». Появится сообщение «Запись успешно добавлена.» (рисунок А.34). Возврат на форму «Сотрудники». Если нажать на кнопку «Отмена», то произойдёт возвращение на форму «Сотрудники».

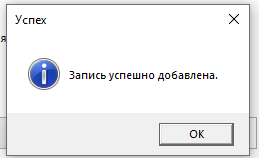


Рисунок А.34 – Сообщение об успешном добавлении

**4.2 Описание операции роли «Администратор»**

ИС выполняет операции, приведенные в таблице А.2.

Таблица А.2 – Операции, выполняемые в ИС, для роли «Администратор».

|  |  |
| --- | --- |
| **Задачи** | **Описание** |
| Добавление записей в базу данных. | Администратор добавляет запись в базу данных. |
| Добавление логина, пароля и изменение роли пользователей системы. | Администратор добавляет логин, пароль и изменяет роль пользователей системы. |

**4.2.1 Описание операции 1: Добавление записей в базу данных.**

Условия выполнения операции: Доступ к этой операции имеют администраторы системы.

Подготовительные действия: соберите все необходимые данные для ввода в систему, включая сведения о сотрудниках, отделах, должностях и других аспектах работы предприятия.

Основные действия:

1 Перейдите в раздел «База данных».

2 Выберите соответствующий раздел в системе.

3 Введите новую запись (рисунок А.35).

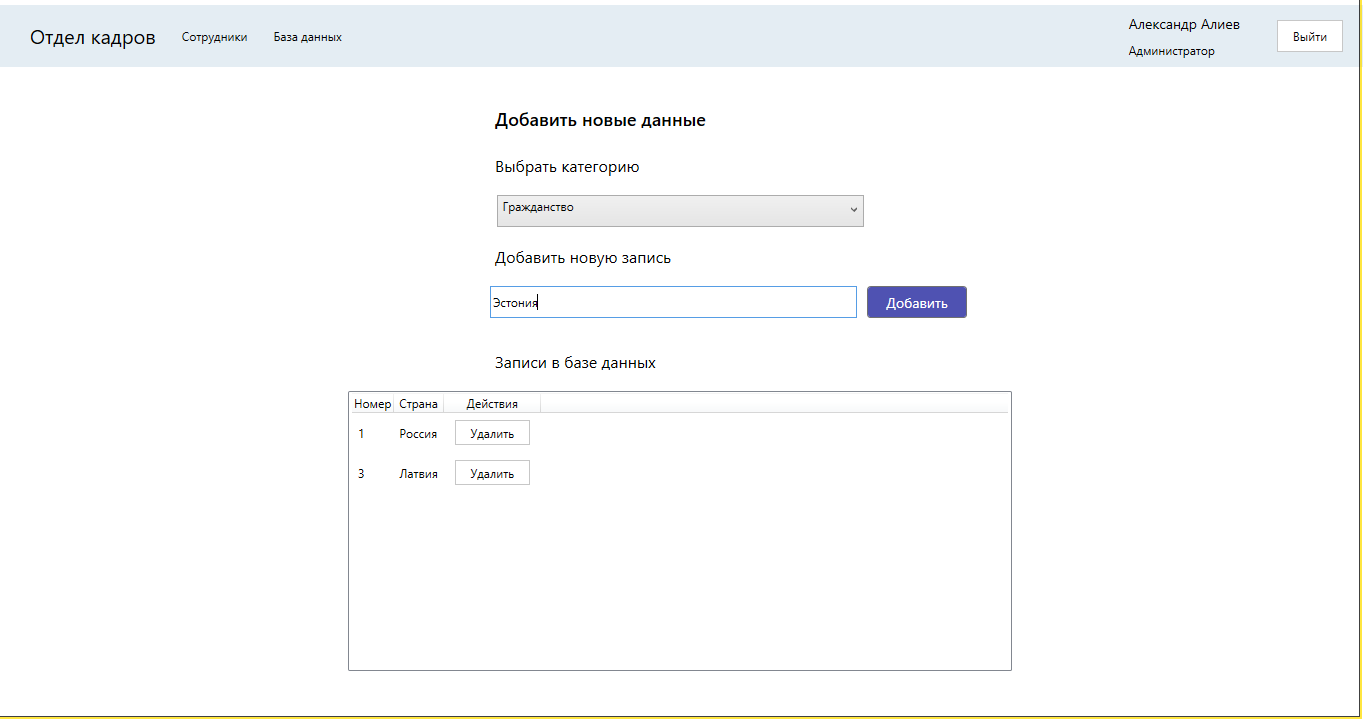


Рисунок А.35 – Форма «База данных»

4 Нажмите кнопку «Добавить» для подтверждения сохранения.

5 Появится сообщение «Данные успешно добавлены!» (рисунок А.36).

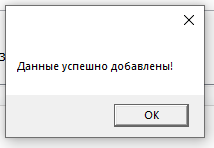


Рисунок А.36 – Уведомление об успешном добавлении

6 Данные можно удалить, нажав на кнопку «Удалить» на нужной строке. При успешном выполнении появится уведомление «Данные успешно удалены!» (рисунок А.37).

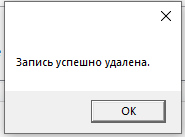


Рисунок А.37 – Сообщение об успешном удалении

Заключительные действия: убедитесь, что новая запись корректно отображается в системе.

Ресурсы, расходуемые на операцию: время на выполнение, зависит от объема и сложности данных, в среднем занимает от 2 до 10 минут на каждую запись.

**4.2.2 Наименование операции: Добавление логина, пароля и изменении роли пользователей**

Условия выполнения операции: доступ к операции предоставляется только администраторам системы.

Подготовительные действия: подготовьте список пользователей для добавления уровня доступа.

Основные действия:

1 Перейдите в раздел «Сотрудники» в системе.

2 Произведите поиск и выберите пользователя.

3 Откроется подробная информация о пользователе.

4 Произведите ввод пароля и логина, а также можно поменять роль (рисунок А.38).

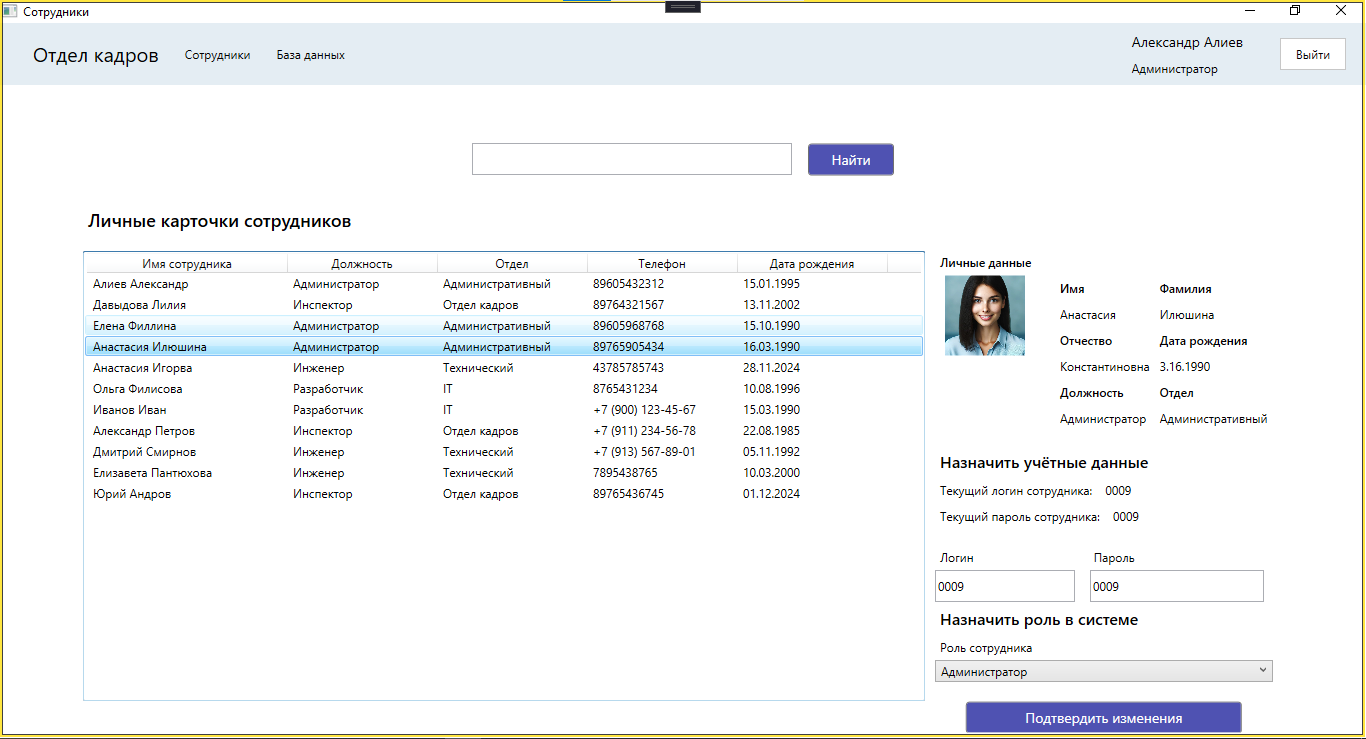


Рисунок А.38 – Форма «Сотрудники»

5 Подтвердите создание нового пользователя, нажав кнопку «Подтвердить изменения». Появится уведомление «Изменения сохранены» (рисунок А.39).

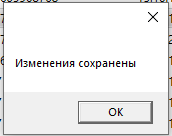


Рисунок А.39 – Сообщение об успешном сохранении

Заключительные действия: проверьте, что пользователь добавлен в систему и имеет соответствующие права доступа.

Ресурсы, расходуемые на операцию: Примерное время на добавление одного пользователя — от 3 до 5 минут.

**4.3 Наименование операции: Выход из системы**

Условия выполнения операции: доступ к операции имеет все пользователи.

Подготовительные действия: убедитесь, что все записи сохранены.

Основные действия:

1 Нажать на кнопку «Выйти» в правом верхнем углу.

2 Появится диалоговое окно для подтверждения вашего действия.

**5. Аварийные ситуации, восстановление базы данных**

В случае возникновения аварийных ситуаций, таких как потеря данных или сбои в работе системы, следуйте следующим шагам для восстановления работы:

* Резервное копирование и восстановление: Всегда поддерживайте актуальное резервное копирование базы данных. В случае потери данных используйте последнюю резервную копию для восстановления.
* Перезагрузка системы: Если система не отвечает, попробуйте перезагрузить компьютер и вновь запустить систему.
* Обращение за технической поддержкой: Если проблема не устранена, обратитесь к службе технической поддержки или к разработчикам системы для профессиональной помощи.

**6. Рекомендации по освоению**

Для более эффективного освоения системы рекомендуется:

* Пройти начальное обучение по работе с системой, предусмотренное разработчиками.
* Изучить все разделы руководства пользователя.
* Начать использование системы с выполнения базовых операций, постепенно переходя к более сложным функциям.
* Регулярно использовать систему для укрепления навыков работы с ней.

**ПРИЛОЖЕНИЕ Б. ЛИСТИНГ КОДА ПРИЛОЖЕНИЯ**

Код Autorization.xaml в таблице Б.1..

Таблица Б.1 – Листинг хода Autorization.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.AutorizationForm"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment"  xmlns:vm="clr-namespace:PersonnelDepartment.ViewModel"  mc:Ignorable="d"  WindowStartupLocation="CenterScreen"  WindowState="Normal"  Title="Окно авторизации" Height="528" Width="562">  <Window.DataContext>  <vm:AuthorizationViewModel />  </Window.DataContext>  <Grid>  <StackPanel>  <StackPanel>  <TextBlock FontSize="20" Margin="20,16,0,24">Отдел кадров</TextBlock>  </StackPanel>  <StackPanel>  <TextBlock FontSize="12.36" HorizontalAlignment="Center" Margin="0, 43,0,0">Добро пожаловать в информационную систему "Отдел кадров"!</TextBlock>  <TextBlock FontSize="20" HorizontalAlignment="Center">Вход в систему</TextBlock>  </StackPanel>  <StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">  <TextBlock FontSize="12" Margin="0,54,0,4">Логин</TextBlock>  <TextBox Margin="0,0,0,18" Height="32" Width="187.6" Style="{StaticResource CenteredTextBox}" Name="LoginTb" Text="{Binding Login}"/>  <TextBlock FontSize="12" Margin="0,0,0,4">Пароль</TextBlock>  <PasswordBox Height="32" Width="187.6" VerticalContentAlignment="Center" Name="PassTb" PasswordChanged="PasswordBox\_PasswordChanged"/>  <Button Style="{StaticResource MainButton}" Width="86" Height="32" Margin="0,54,0,0" Content="Войти" Command="{Binding LoginCommand}"/>  </StackPanel>  </StackPanel>  </Grid>  </Window> |

Код C# Autorization.xaml.cs в таблице Б.2.

Таблица Б.2 – Листинг кода Autorization.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.ViewModel;  using System.Windows;  using System.Windows.Controls;  namespace PersonnelDepartment  {  public partial class AutorizationForm : Window  {  public AutorizationForm()  {  InitializeComponent();  }  private void PasswordBox\_PasswordChanged(object sender, RoutedEventArgs e)  {  var passwordBox = sender as PasswordBox;  if (DataContext is AuthorizationViewModel viewModel)  {  viewModel.Password = passwordBox?.Password;  }  }  }  } |

Код MainStaff.xaml в таблице Б.3.

Таблица Б.3 – Листинг кода MainStaff.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.Staff.MainStaff"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.Staff"  mc:Ignorable="d"  WindowState="Maximized"  xmlns:conv="clr-namespace:PersonnelDepartment.Converter"  xmlns:viewmodel="clr-namespace:PersonnelDepartment.ViewModel"  d:DataContext="{d:DesignInstance Type=viewmodel:MainStaffViewModel}"  Title="Сотрудники" Height="720" Width="1280">    <Window.Resources>  <conv:ByteArrayToImageConverter x:Key="ByteArrayToImageConverter"/>  </Window.Resources>    <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <!-- Верхняя панель с кнопками -->  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" Command="{Binding NavigateToMainWindowCommand}"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" IsEnabled="False"/>  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}" Command="{Binding NavigateToEmploymentRecordCommand}"/>  <Button Content="Приказы" Style="{StaticResource ButtonMain}" Command="{Binding NavigateToOrdersCommand}"/>  <Button Content="Отчеты" Style="{StaticResource ButtonMain}" Command="{Binding NavigateToReportsCommand}"/>  <Button Content="Добавить сотрудника" Command="{Binding AddStaffCommand}" Grid.Row="0" Width="194" Height="32" HorizontalAlignment="Left" Margin="27,0,0,0" Style="{StaticResource MainButton}" x:Name="AddStaff"/>  </StackPanel>  <!-- Панель информации о текущем пользователе -->  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label Content="{Binding UserName}" FontSize="14" Margin="0,0,32,0"/>  <Label Content="{Binding UserRole}" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image Width="32" Height="32" Source="{Binding SelectedEmployee.PhotoData, Converter={StaticResource ByteArrayToImageConverter}}"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Command="{Binding ExitCommand}" Grid.Column="1"/>  </StackPanel>  </Grid>  <!-- Основной контент окна -->  <Grid Grid.Row="1">  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0" Grid.ColumnSpan="7">  <!-- Поиск сотрудников -->  <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">  <TextBox Width="320" Height="32" Style="{StaticResource CenteredTextBox}" Margin="0,52,16,27" Text="{Binding SearchText}"/>  <Button Width="86" Height="32" Margin="0, 53, 0,27" Content="Найти" Command="{Binding SearchCommand}" Style="{StaticResource MainButton}" IsDefault="True"/>  </StackPanel>  <!-- Список сотрудников -->  <StackPanel Orientation="Horizontal">  <StackPanel Width="882" HorizontalAlignment="Left">  <Label Content="Личные карточки сотрудников" FontWeight="DemiBold" FontSize="18" Margin="40,0,0,15"/>  <ListView Width="842" Height="450" x:Name="EmployeeListView" ItemsSource="{Binding FilteredEmployees}" Style="{StaticResource ListViewSt}" SelectedItem="{Binding SelectedEmployee}" MouseDoubleClick="ListView\_MouseDoubleClick" Margin="40,0,0,0">  <ListView.View>  <GridView>  <GridViewColumn Header="Имя сотрудника" DisplayMemberBinding="{Binding FullName}" Width="200"/>  <GridViewColumn Header="Должность" DisplayMemberBinding="{Binding Position}" Width="150"/>  <GridViewColumn Header="Отдел" DisplayMemberBinding="{Binding Department}" Width="150"/>  <GridViewColumn Header="Телефон" DisplayMemberBinding="{Binding Phone}" Width="150"/>  <GridViewColumn Header="Дата рождения" DisplayMemberBinding="{Binding BirthDate, StringFormat=dd.MM.yyyy}" Width="150"/>  </GridView>  </ListView.View>  </ListView>  </StackPanel>  <!-- Личные данные выбранного сотрудника -->  <StackPanel>  <Label Content="Личные данные" FontWeight="DemiBold" Margin="0,47,0,0"/>  <StackPanel Orientation="Horizontal" Margin="0, 0, 0, 15">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Width="120">  <Image Width="80" Height="80" Source="{Binding SelectedEmployee.PhotoData, Converter={StaticResource ByteArrayToImageConverter}}" HorizontalAlignment="Left" Margin="10, 0, 0, 0"/>  </StackPanel>  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <Label Content="Имя" FontSize="12" Grid.Row="0" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.FirstName}" Grid.Row="1" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Фамилия" FontSize="12" Grid.Row="0" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.LastName}" Grid.Row="1" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Отчество" FontSize="12" Grid.Row="2" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.Patronymic}" Grid.Row="3" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Дата рождения" FontSize="12" Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.BirthDate, StringFormat=dd.MM.yyyy}" Grid.Row="3" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Должность" FontSize="12" Grid.Row="4" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.Position}" Grid.Row="5" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Отдел" FontSize="12" Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.Department}" Grid.Row="5" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  </Grid>  </StackPanel>  <!-- Кнопка "Назначить награду" -->  <StackPanel VerticalAlignment="Bottom" HorizontalAlignment="Center">  <Button Width="245" Height="32" Content="Назначить награду" Command="{Binding AssignRewardCommand}" Style="{StaticResource MainButton}" Margin="50,250,0,0"/>  </StackPanel>  </StackPanel>  </StackPanel>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# MainStaff.xaml.cs в таблице Б.4.

Таблица Б.4 – Листинг кода MainStaff.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.ViewModel;  using System.Windows;  namespace PersonnelDepartment.Staff  {  /// <summary>  /// Логика взаимодействия для MainStaff.xaml  /// </summary>  public partial class MainStaff : Window  {  private Personal\_card \_currentUser;  public MainStaff(Personal\_card currentUser)  {  \_currentUser = currentUser;  var viewModel = new MainStaffViewModel(currentUser, this);  DataContext = viewModel;  InitializeComponent();  }  private void ListView\_MouseDoubleClick(object sender, System.Windows.Input.MouseButtonEventArgs e)  {  if (EmployeeListView.SelectedItem is Employee selectedEmployee)  {  // Вызываем команду для редактирования сотрудника  var viewModel = DataContext as MainStaffViewModel;  viewModel?.OpenEditEmployeeWindow(selectedEmployee);  }  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код MainWindowInspector.xaml а таблице Б.5.

Таблица Б.5 – Листинг кода MainWindowInspector.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.MainWindowInspector"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment"  xmlns:vm="clr-namespace:PersonnelDepartment.ViewModel"  mc:Ignorable="d"  WindowStartupLocation="CenterScreen"  WindowState="Maximized"  Title="Главная страница" Height="720" Width="1280">  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" Command="{Binding StaffCommand}"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" Command="{Binding StaffCommand}"/>  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}" Command="{Binding EmploymentRecordCommand}"/>  <Button Content="Приказы" Style="{StaticResource ButtonMain}" Command="{Binding OrdersCommand}"/>  <Button Content="Отчеты" Style="{StaticResource ButtonMain}" Command="{Binding ReportCommand}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label Content="{Binding UserName}" FontSize="14" Margin="0,0,32,0"/>  <Label Content="{Binding UserRole}" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image Width="32" Height="32" Source="{Binding UserImage}" />-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" Command="{Binding ExitCommand}" x:Name="ExitBtn" Click="ExitBtn\_Click"/>  </StackPanel>  <Grid>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <Label Content="Выберите подсистему" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Center" Margin="0,287"/>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# MainWindowInspector.xaml.cs а таблице Б.6.

Таблица Б.6 – Листинг кода MainWindowInspector.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.ViewModel;  using System.Windows;  namespace PersonnelDepartment  {  public partial class MainWindowInspector : Window  {  private Personal\_card \_currentUser;  public MainWindowInspector(Personal\_card currentUser)  {  InitializeComponent();  \_currentUser = currentUser;  var viewModel = new MainWindowInspectorViewModel(\_currentUser, this);  DataContext = viewModel;  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код AddStaff.xaml в таблице Б.7.

Таблица Б.7 – Листинг кода AddStaff.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.Staff.AddStaff"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.Staff"  mc:Ignorable="d"  WindowState="Maximized"  Title="AddStaff" Height="720" Width="1280">  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" x:Name="MainWindowBtn" Click="MainWindowBtn\_Click"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" />  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}" x:Name="EmploymentRecordBtn" Click="EmploymentRecordBtn\_Click"/>  <Button Content="Приказы" x:Name="OrdersBtn" Style="{StaticResource ButtonMain}" Click="OrdersBtn\_Click"/>  <Button Content="Отчеты" x:Name="ReportsBtn" Style="{StaticResource ButtonMain}" Click="ReportsBtn\_Click"/>  <Button Content="Отмена" x:Name="CancelBtn" Grid.Row="0" Width="86" Height="32" Click="CancelBtn\_Click" HorizontalAlignment="Left" Margin="27,0,0,0" Style="{StaticResource MainButton}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="Namelbl" Content="Имя Фамилия" FontSize="14" Margin="0,0,32,0"/>  <Label x:Name="Rolelbl" Content="Идолжность" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image Width="32" Height="32"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>    <Grid>  <StackPanel Orientation="Horizontal" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <!--Шаг 1 (левая часть окна)-->  <StackPanel Width="560" Margin="30">  <Grid>  <Label Content="Добавить сотрудника в компанию" FontWeight="SemiBold" FontSize="18" />  <Label Content="Шаг 1" FontSize="18" FontWeight="SemiBold" HorizontalAlignment="Right"/>  </Grid>  <Label Content="Личные данные" FontSize="16" Margin="0, 0, 0, 0"/>  <!--Аватар + ФИО-->  <StackPanel Orientation="Horizontal" Margin="0, 0, 0, 15">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Width="174">  <Label Content="Аватар" FontSize="12" HorizontalAlignment="Left" Margin="0, 0, 0, 10"/>  <Image x:Name="AvatarImg" Width="80" Height="80" Source="../Media/template.png" HorizontalAlignment="Left" Margin="10, 0, 0, 0"/>  <Button Width="20" Height="20" x:Name="LoadImage" Click="LoadImage\_Click">  <Image Source="C:\Users\Lenovo L530\source\repos\PersonnelDepartment\ImageBtn\ImgBtn.png" />  </Button>  </StackPanel>  <!--Имя, Фамилия, Отчество-->  <StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Имя" FontSize="12"/>  <TextBox x:Name="nameTextBox" Style="{StaticResource CenteredTextBox}" Width="174" Height="32"/>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left">  <Label Content="Фамилия" FontSize="12"/>  <TextBox x:Name="surnameTextBox" Style="{StaticResource CenteredTextBox}" Width="174" Height="32"/>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Отчество" FontSize="12"/>  <TextBox x:Name="patronymicTextBox" Style="{StaticResource CenteredTextBox}" Width="366" Height="32"/>  </StackPanel>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Дата рождения" FontSize="12"/>  <DatePicker x:Name="dateOfBirthPicker" Width="174" Height="32"/>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left">  <Label Content="Место рождения" FontSize="12"/>  <TextBox x:Name="birthplaceTextBox" Style="{StaticResource CenteredTextBox}" Width="346" Height="32"/>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical" Margin="0,0,0,10">  <Label Content="Адрес регистрации" FontSize="12"/>  <TextBox x:Name="registrationAddressTextBox" Style="{StaticResource CenteredTextBox}" HorizontalAlignment="Left" Width="539" Height="32"/>  </StackPanel>  <StackPanel>  <Label Content="Контактные данные" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Margin="0,0,0,10">  <Label Content="Телефонный номер" FontSize="12"/>  <TextBox x:Name="telephoneTextBox" Style="{StaticResource CenteredTextBox}" HorizontalAlignment="Left" Width="262" Height="32"/>  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Электронная почта" FontSize="12"/>  <TextBox x:Name="EmailTxb" Style="{StaticResource CenteredTextBox}" Width="261" Height="32"/>  </StackPanel>  </StackPanel>  <StackPanel>  <Label Content="Трудовые данные" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel>  <Label Content="Отдел" FontSize="12"/>  <ComboBox x:Name="departmentComboBox" HorizontalAlignment="Left" Width="262" Height="32" SelectionChanged="departmentComboBox\_SelectionChanged" />  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Должность" FontSize="12"/>  <ComboBox x:Name="postComboBox" Width="261" Height="32" />  </StackPanel>  </StackPanel>    </StackPanel>  <!--Шаг 2 (правая часть окна)-->  <StackPanel>  <StackPanel>  <Label Content="Шаг 2" FontSize="18" FontWeight="SemiBold" Margin="0,29,56,0"/>  </StackPanel>  <StackPanel>  <Label Content="Паспортные данные" FontSize="16" Margin="0,0,0,0"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel>  <Label Content="Серия" FontSize="12" Margin="0,0,0,3"/>  <TextBox x:Name="seriesTextBox" Style="{StaticResource CenteredTextBox}" VerticalAlignment="Top" HorizontalAlignment="Left" Width="262" Height="32" Margin="0,0,0,0"/>  </StackPanel>  <StackPanel Margin="19,0,0,10">  <Label Content="Номер" FontSize="12" Margin="0,0,0,3"/>  <TextBox x:Name="numberTextBox" Style="{StaticResource CenteredTextBox}" VerticalAlignment="Top" Width="262" Height="32" />  </StackPanel>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">  <StackPanel>  <Label Content="Дата выдачи" FontSize="12" Margin="0,0,0,3"/>  <DatePicker x:Name="dateOfIssuePicker" VerticalAlignment="Top" HorizontalAlignment="Left" Width="174" Height="32" Margin="0,0,0,0"/>  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Кем выдан" FontSize="12" Margin="0,0,0,3"/>  <TextBox x:Name="issuedByTextBox" Style="{StaticResource CenteredTextBox}" HorizontalAlignment="Left" Width="351" Height="32" Margin="0,0,0,0"/>  </StackPanel>  </StackPanel>  <StackPanel>  <Label Content="Образование" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">  <StackPanel>  <Label Content="Уровень" FontSize="12"/>  <ComboBox x:Name="educationComboBox" HorizontalAlignment="Left" Width="265" Height="32" />  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Учебное заведение" FontSize="12"/>  <TextBox x:Name="EducationInstitutionTextBox" Style="{StaticResource CenteredTextBox}" Width="262" Height="32" />  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical">  <Label FontSize="12" Content="Гражданство"/>  <ComboBox x:Name="citizenshipComboBox" HorizontalAlignment="Left" Width="265" Height="32"/>  </StackPanel>  <StackPanel Margin="0,0,0,10">  <Label Content="Прочие данные" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">  <Label Content="Есть дети?"/>  <RadioButton x:Name="childrenRadioButton"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content="Военнообязанный?"/>  <RadioButton x:Name="militaryServiceRadioButton"/>  </StackPanel>  <Button Height="32" Width="182" Margin="0,90,0,0" VerticalAlignment="Bottom" x:Name="CreateOrderBtn" Style="{StaticResource MainButton}" Content="Создать приказ" Click="CreateOrderBtn\_Click"/>  </StackPanel>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# AddStaff.xalm.cs в таблице Б.8.

Таблица Б.8 – Листинг кода AddStaff.xalm.

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  using System;  using System.Collections.Generic;  using System.Data;  using System.Data.SqlClient;  using System.IO;  using System.Linq;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.Staff  {  /// <summary>  /// Логика взаимодействия для AddStaff.xaml  /// </summary>  public partial class AddStaff : Window  {  private Personal\_card \_currentUser; // Хранение данных текущего пользователя  public AddStaff(Personal\_card currentUser)  {  \_currentUser = currentUser;  InitializeComponent();  LoadComboBoxData();    DisplayUserInfo(); // Отображение информации о текущем пользователе  }  private void DisplayUserInfo()  {  if (\_currentUser != null)  {  Namelbl.Content = $"{\_currentUser.Surname} {\_currentUser.Name}";  Rolelbl.Content = \_currentUser.Post.Title;  }  }  private void LoadComboBoxData()  {  string connectionString = "data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework";  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Заполнение citizenshipComboBox  string queryCitizenship = "SELECT ID, Country FROM Citizenship";  using (SqlCommand command = new SqlCommand(queryCitizenship, connection))  using (SqlDataReader reader = command.ExecuteReader())  {  var citizenshipList = new List<KeyValuePair<int, string>>();  while (reader.Read())  {  citizenshipList.Add(new KeyValuePair<int, string>((int)reader["ID"], (string)reader["Country"]));  }  citizenshipComboBox.ItemsSource = citizenshipList;  citizenshipComboBox.DisplayMemberPath = "Value";  citizenshipComboBox.SelectedValuePath = "Key";  }  // Заполнение departmentComboBox  string queryDepartment = "SELECT ID, Title FROM Department";  using (SqlCommand command = new SqlCommand(queryDepartment, connection))  using (SqlDataReader reader = command.ExecuteReader())  {  var departmentList = new List<KeyValuePair<int, string>>();  while (reader.Read())  {  departmentList.Add(new KeyValuePair<int, string>((int)reader["ID"], (string)reader["Title"]));  }  departmentComboBox.ItemsSource = departmentList;  departmentComboBox.DisplayMemberPath = "Value";  departmentComboBox.SelectedValuePath = "Key";  }  // Загрузка всех должностей в postComboBox для дальнейшего фильтра  string queryPost = "SELECT ID, Title, Id\_department FROM Post";  using (SqlCommand command = new SqlCommand(queryPost, connection))  using (SqlDataReader reader = command.ExecuteReader())  {  var postList = new List<dynamic>();  while (reader.Read())  {  postList.Add(new  {  Id = (int)reader["ID"],  Title = (string)reader["Title"],  IdDepartment = (int)reader["Id\_department"]  });  }  postComboBox.ItemsSource = postList;  postComboBox.DisplayMemberPath = "Title";  postComboBox.SelectedValuePath = "Id";  postComboBox.Tag = postList; // Сохраняем полный список должностей в Tag для фильтрации  }  // Заполнение educationComboBox  string queryEducation = "SELECT ID, Levell FROM Education";  using (SqlCommand command = new SqlCommand(queryEducation, connection))  using (SqlDataReader reader = command.ExecuteReader())  {  var educationList = new List<KeyValuePair<int, string>>();  while (reader.Read())  {  educationList.Add(new KeyValuePair<int, string>((int)reader["ID"], (string)reader["Levell"]));  }  educationComboBox.ItemsSource = educationList;  educationComboBox.DisplayMemberPath = "Value";  educationComboBox.SelectedValuePath = "Key";  }  }  }  public int AddEmployeeAndSaveSalary(PersonalCard personalCard)  {  string connectionString = "data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework";  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  SqlTransaction transaction = connection.BeginTransaction();  try  {  // Добавление сотрудника  string personalCardQuery = @"  INSERT INTO Personal\_card (Surname, Name, Patronymic, Date\_of\_birth, Series\_and\_number,  Issued\_by\_whom, Date\_of\_issue, Registration\_address, Telephone,  Children, Military\_service, Photo, Id\_citizenship, Id\_post,  Id\_education, Birthplace, Email, EducationInstitution)  OUTPUT INSERTED.ID  VALUES (@Surname, @Name, @Patronymic, @DateOfBirth, @SeriesAndNumber,  @IssuedByWhom, @DateOfIssue, @RegistrationAddress, @Telephone,  @Children, @MilitaryService, @Photo, @IdCitizenship, @IdPost,  @IdEducation, @Birthplace, @Email, @EducationInstitution)";  int newPersonalCardId;  using (SqlCommand personalCardCommand = new SqlCommand(personalCardQuery, connection, transaction))  {  personalCardCommand.Parameters.AddWithValue("@Surname", personalCard.Surname);  personalCardCommand.Parameters.AddWithValue("@Name", personalCard.Name);  personalCardCommand.Parameters.AddWithValue("@Patronymic", personalCard.Patronymic);  personalCardCommand.Parameters.AddWithValue("@DateOfBirth", personalCard.DateOfBirth);  personalCardCommand.Parameters.AddWithValue("@SeriesAndNumber", personalCard.SeriesAndNumber);  personalCardCommand.Parameters.AddWithValue("@IssuedByWhom", personalCard.IssuedByWhom);  personalCardCommand.Parameters.AddWithValue("@DateOfIssue", personalCard.DateOfIssue);  personalCardCommand.Parameters.AddWithValue("@RegistrationAddress", personalCard.RegistrationAddress);  personalCardCommand.Parameters.AddWithValue("@Telephone", personalCard.Telephone);  personalCardCommand.Parameters.AddWithValue("@Children", personalCard.Children);  personalCardCommand.Parameters.AddWithValue("@MilitaryService", personalCard.MilitaryService);  personalCardCommand.Parameters.AddWithValue("@Photo", personalCard.Photo);  personalCardCommand.Parameters.AddWithValue("@IdCitizenship", personalCard.IdCitizenship);  personalCardCommand.Parameters.AddWithValue("@IdPost", personalCard.IdPost);  personalCardCommand.Parameters.AddWithValue("@IdEducation", personalCard.IdEducation);  personalCardCommand.Parameters.AddWithValue("@Birthplace", personalCard.Birthplace);  personalCardCommand.Parameters.AddWithValue("@Email", personalCard.Email ?? (object)DBNull.Value);  personalCardCommand.Parameters.AddWithValue("@EducationInstitution", personalCard.EducationInstitution ?? (object)DBNull.Value);  newPersonalCardId = (int)personalCardCommand.ExecuteScalar();  }  // Получение базовой зарплаты для выбранной должности  string salaryQuery = @"  SELECT Base\_salary  FROM Post  WHERE ID = @IdPost";  double baseSalary;  using (SqlCommand salaryCommand = new SqlCommand(salaryQuery, connection, transaction))  {  salaryCommand.Parameters.AddWithValue("@IdPost", personalCard.IdPost);  object result = salaryCommand.ExecuteScalar();  if (result == null || result == DBNull.Value)  {  throw new Exception("Base\_salary не найден для указанной должности.");  }  baseSalary = Convert.ToDouble(result);  }  // Добавление записи в таблицу Salary  string insertSalaryQuery = @"  INSERT INTO Salary (Id\_personal\_card, Amount, Date, Id\_salary\_type)  VALUES (@IdPersonalCard, @Amount, @Date, @IdSalaryType)";  using (SqlCommand insertSalaryCommand = new SqlCommand(insertSalaryQuery, connection, transaction))  {  insertSalaryCommand.Parameters.AddWithValue("@IdPersonalCard", newPersonalCardId);  insertSalaryCommand.Parameters.AddWithValue("@Amount", baseSalary);  insertSalaryCommand.Parameters.AddWithValue("@Date", DateTime.Now.Date);  insertSalaryCommand.Parameters.AddWithValue("@IdSalaryType", 3);  insertSalaryCommand.ExecuteNonQuery();  }  // Добавление записи в таблицу Entry\_in\_the\_work  string entryQuery = @"  INSERT INTO Entry\_in\_the\_work\_book (Id\_personal\_card, Date, Reason, Id\_Mixing)  VALUES (@IdPersonalCard, @Date, @Reason, @IdMixing)";  using (SqlCommand entryCommand = new SqlCommand(entryQuery, connection, transaction))  {  entryCommand.Parameters.AddWithValue("@IdPersonalCard", newPersonalCardId);  entryCommand.Parameters.AddWithValue("@Date", DateTime.Now.Date);  entryCommand.Parameters.AddWithValue("@Reason", DBNull.Value); // Пустое значение для причины  entryCommand.Parameters.AddWithValue("@IdMixing", 1); // Значение Id\_Mixing всегда 1  entryCommand.ExecuteNonQuery();  }  transaction.Commit();  return newPersonalCardId;  }  catch (Exception ex)  {  transaction.Rollback();  throw new Exception($"Ошибка при добавлении сотрудника: {ex.Message}", ex);  }  }  }  private void departmentComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (departmentComboBox.SelectedValue == null) return;  int selectedDepartmentId = (int)departmentComboBox.SelectedValue;  // Фильтруем должности по выбранному отделу  var allPosts = (List<dynamic>)postComboBox.Tag;  var filteredPosts = allPosts.Where(p => p.IdDepartment == selectedDepartmentId).ToList();  postComboBox.ItemsSource = filteredPosts;  postComboBox.SelectedValue = null;  }  private void CreateOrderBtn\_Click(object sender, RoutedEventArgs e)  {  string errorMessage;  // Валидация имени  if (!Validator.IsValidName(nameTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация фамилии  if (!Validator.IsValidSurName(surnameTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация отчества  if (!Validator.IsValidPatronymic(patronymicTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }    // Валидация даты рождения  if (dateOfBirthPicker.SelectedDate.HasValue)  {  string dateOfBirthString = dateOfBirthPicker.SelectedDate.Value.ToString("dd.MM.yyyy"); // Преобразуем в строку  if (!Validator.IsValidBirthDate(dateOfBirthString, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  }  else  {  MessageBox.Show("Пожалуйста, выберите дату рождения.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  // Валидация места рождения  if (!Validator.IsValidPlace(birthplaceTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация адреса регистрации  if (!Validator.IsValidAddress(registrationAddressTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация телефона  if (!Validator.IsValidPhone(telephoneTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация электронной почты  if (!Validator.IsValidEmail(EmailTxb.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация отдела  if (!Validator.IsValidField(departmentComboBox.Text, out errorMessage, "Отдел не может быть пустым"))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация должности  if (!Validator.IsValidField(postComboBox.Text, out errorMessage, "Должность не может быть пустой"))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация серии паспорта  if (!Validator.IsValidPassportSeries(seriesTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация номера паспорта  if (!Validator.IsValidPassportNumber(numberTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация даты выдачи паспорта  if (dateOfIssuePicker.SelectedDate.HasValue)  {  string dateOfIssueString = dateOfIssuePicker.SelectedDate.Value.ToString("dd.MM.yyyy"); // Преобразуем в строку  if (!Validator.IsValidDateOfIssue(dateOfIssueString, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  }  else  {  MessageBox.Show("Пожалуйста, выберите дату выдачи.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  // Валидация "Кем выдан"  if (!Validator.IsValidField(issuedByTextBox.Text, out errorMessage, "Поле «Кем выдан» не может быть пустым"))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация уровня образования  if (!Validator.IsValidField(educationComboBox.Text, out errorMessage, "Уровень образования не может быть пустым"))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация учебного заведения  if (!Validator.IsValidEducationInstitution(EducationInstitutionTextBox.Text, out errorMessage))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  // Валидация гражданства  if (!Validator.IsValidField(citizenshipComboBox.Text, out errorMessage, "Гражданство не может быть пустым"))  {  MessageBox.Show(errorMessage, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  byte[] photoData = null;  if (AvatarImg.Source != null)  {  var bitmapImage = AvatarImg.Source as BitmapImage;  using (var memoryStream = new MemoryStream())  {  var encoder = new JpegBitmapEncoder();  encoder.Frames.Add(BitmapFrame.Create(bitmapImage));  encoder.Save(memoryStream);  photoData = memoryStream.ToArray();  }  }  var personalCard = new PersonalCard  {  Surname = surnameTextBox.Text,  Name = nameTextBox.Text,  Patronymic = patronymicTextBox.Text,  DateOfBirth = dateOfBirthPicker.SelectedDate ?? DateTime.MinValue,  SeriesAndNumber = seriesTextBox.Text + numberTextBox.Text,  IssuedByWhom = issuedByTextBox.Text,  DateOfIssue = dateOfIssuePicker.SelectedDate ?? DateTime.MinValue,  RegistrationAddress = registrationAddressTextBox.Text,  Telephone = telephoneTextBox.Text,  Children = childrenRadioButton.IsChecked == true,  MilitaryService = militaryServiceRadioButton.IsChecked == true,  IdCitizenship = (int?)citizenshipComboBox.SelectedValue ?? 0,  IdPost = (int?)postComboBox.SelectedValue ?? 0,  IdEducation = (int?)educationComboBox.SelectedValue ?? 0,  Birthplace = birthplaceTextBox.Text,  Email = EmailTxb.Text,  EducationInstitution = EducationInstitutionTextBox.Text,  Photo = photoData  };  try  {  int newEmployeeId = AddEmployeeAndSaveSalary(personalCard);  MessageBox.Show("Сотрудник успешно добавлен.");  // Открытие формы "Приказы"  MainOrders mainOrdersWindow = new MainOrders(\_currentUser);  mainOrdersWindow.Show();  // Закрываем текущее окно, если оно больше не нужно  this.Close();  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при добавлении сотрудника: {ex.Message}");  }  }  private void LoadImage\_Click(object sender, RoutedEventArgs e)  {  var openFileDialog = new Microsoft.Win32.OpenFileDialog  {  Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;\*.bmp"  };  if (openFileDialog.ShowDialog() == true)  {  AvatarImg.Source = new BitmapImage(new Uri(openFileDialog.FileName));  }  }  private void MainWindowBtn\_Click(object sender, RoutedEventArgs e)  {  MainWindowAdmin mainWindowAdmin = new MainWindowAdmin(\_currentUser);  mainWindowAdmin.Show();  Close();  }  private void EmploymentRecordBtn\_Click(object sender, RoutedEventArgs e)  {  MainEmploymentRecord mainEmploymentRecord = new MainEmploymentRecord(\_currentUser);  mainEmploymentRecord.Show();  Close();  }  private void OrdersBtn\_Click(object sender, RoutedEventArgs e)  {  MainOrders mainOrders = new MainOrders(\_currentUser);  mainOrders.Show();  Close();  }  private void ReportsBtn\_Click(object sender, RoutedEventArgs e)  {  MainReports mainReports = new MainReports(\_currentUser);  mainReports.Show();  Close();  }  private void CancelBtn\_Click(object sender, RoutedEventArgs e)  {  MainStaff mainStaff = new MainStaff(\_currentUser);  mainStaff.Show();  Close();  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код AssignRewardForm.xaml в таблице Б.9.

Таблица Б.9 – Листинг кода AssignRewardForm.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.Staff.AssignRewardForm"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.Staff"  mc:Ignorable="d"  WindowStartupLocation="CenterScreen"  WindowState="Normal"  Title="Добавление награды" Height="350" Width="470">  <Grid>  <StackPanel>  <Label Content="Добавление записи о премии/ командировочных" FontSize="16" FontWeight="DemiBold" Margin="32,27,0,21"/>  <StackPanel Margin="15,0,0,15">  <Label Content="Сотрудник" Margin="0,0,0,15"/>  <TextBlock Width="245" x:Name="EmployeeNameTextBlock" Height="32" HorizontalAlignment="Left"/>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="15,0,0,15">  <Label Content="Тип награды" Margin="0,0,115,0"/>  <Label Content="Сумма"/>  </StackPanel>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="15,0,0,15">  <ComboBox Width="150" x:Name="RewardTypeComboBox" Height="32" Margin="0,0,50,0" DisplayMemberPath="Title" SelectionChanged="RewardTypeComboBox\_SelectionChanged"/>  <TextBox x:Name="AmountTextBox" Width="150" Height="32"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Button Width="107" Height="32" Style="{StaticResource ExitButton}" Content="Отмена" Margin="191,10,0,0" x:Name="CancelBtn" Click="CancelBtn\_Click" />  <Button Width="96" Height="32" Style="{StaticResource MainButton}" Content="Добавить" Margin="8,10,0,0" x:Name="AddButton" Click="AddButton\_Click"/>  </StackPanel>  </StackPanel>  </Grid>  </Window> |

Код C# AssignRewardForm.xaml.cs в таблице Б.10.

Таблица Б.10 – Листинг кода AssignRewardForm.xaml.cs.

|  |
| --- |
| using System.Windows;  using System.Data.SqlClient;  using System;  namespace PersonnelDepartment.Staff  {  /// <summary>  /// Логика взаимодействия для AssignRewardForm.xaml  /// </summary>  public partial class AssignRewardForm : Window  {  private int \_employeeId; // ID сотрудника  public AssignRewardForm(int employeeId, string employeeName)  {  InitializeComponent();  \_employeeId = employeeId;  // Отображение имени сотрудника  EmployeeNameTextBlock.Text = $"{employeeName}";  // Загрузка типов наград в ComboBox  LoadRewardTypes();  }  private void LoadRewardTypes()  {  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False"))  {  connection.Open();  var command = new SqlCommand("SELECT ID, Title FROM Salary\_type WHERE ID IN (1, 2, 3)", connection); // Убедитесь, что выбираем 1, 2 и 3  var reader = command.ExecuteReader();  while (reader.Read())  {  RewardTypeComboBox.Items.Add(new { Id = reader.GetInt32(0), Title = reader.GetString(1) }); // Добавляем в ComboBox  }  }  }  private void CancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  private void AddButton\_Click(object sender, RoutedEventArgs e)  {  if (RewardTypeComboBox.SelectedItem == null || string.IsNullOrWhiteSpace(AmountTextBox.Text))  {  MessageBox.Show("Заполните все поля.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  if (!float.TryParse(AmountTextBox.Text, out float amount))  {  MessageBox.Show("Сумма должна быть числом.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  return;  }  var selectedReward = (dynamic)RewardTypeComboBox.SelectedItem;  int rewardTypeId = selectedReward.Id; // Получаем ID награды  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False"))  {  connection.Open();  // Проверка, существует ли запись для выбранного сотрудника и типа награды  var checkCommand = new SqlCommand("SELECT Amount FROM Salary WHERE Id\_personal\_card = @employeeId AND Id\_salary\_type = @rewardTypeId", connection);  checkCommand.Parameters.AddWithValue("@employeeId", \_employeeId);  checkCommand.Parameters.AddWithValue("@rewardTypeId", rewardTypeId);  var existingAmount = checkCommand.ExecuteScalar(); // Проверяем наличие записи  if (existingAmount != null) // Запись найдена  {  // Обновляем запись  var updateCommand = new SqlCommand("UPDATE Salary SET Amount = @amount, Date = @date WHERE Id\_personal\_card = @employeeId AND Id\_salary\_type = @rewardTypeId", connection);  updateCommand.Parameters.AddWithValue("@employeeId", \_employeeId);  updateCommand.Parameters.AddWithValue("@amount", amount);  updateCommand.Parameters.AddWithValue("@date", DateTime.Now);  updateCommand.Parameters.AddWithValue("@rewardTypeId", rewardTypeId);  try  {  updateCommand.ExecuteNonQuery();  MessageBox.Show("Запись успешно обновлена.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при обновлении: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  }  else // Запись не найдена  {  // Добавляем новую запись  var insertCommand = new SqlCommand("INSERT INTO Salary (Id\_personal\_card, Amount, Date, Id\_salary\_type) VALUES (@employeeId, @amount, @date, @rewardTypeId)", connection);  insertCommand.Parameters.AddWithValue("@employeeId", \_employeeId);  insertCommand.Parameters.AddWithValue("@amount", amount);  insertCommand.Parameters.AddWithValue("@date", DateTime.Now);  insertCommand.Parameters.AddWithValue("@rewardTypeId", rewardTypeId);  try  {  insertCommand.ExecuteNonQuery();  MessageBox.Show("Запись успешно добавлена.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при добавлении: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  }  }  this.Close();  }  private void RewardTypeComboBox\_SelectionChanged(object sender, System.Windows.Controls.SelectionChangedEventArgs e)  {  if (RewardTypeComboBox.SelectedItem == null)  return;  var selectedReward = (dynamic)RewardTypeComboBox.SelectedItem;  int rewardTypeId = selectedReward.Id;  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False"))  {  connection.Open();  // Проверяем, есть ли запись для выбранного сотрудника и типа награды  var command = new SqlCommand("SELECT Amount FROM Salary WHERE Id\_personal\_card = @employeeId AND Id\_salary\_type = @rewardTypeId", connection);  command.Parameters.AddWithValue("@employeeId", \_employeeId);  command.Parameters.AddWithValue("@rewardTypeId", rewardTypeId);  var existingAmount = command.ExecuteScalar();  if (existingAmount != null) // Запись найдена  {  AmountTextBox.Text = existingAmount.ToString(); // Отображаем сумму в TextBox  }  else  {  AmountTextBox.Clear(); // Если записи нет, очищаем TextBox  }  }  }  }  } |

Код EditStaff.xaml в таблице Б.11.

Таблица Б.11 – Листинг кода EditStaff.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.Staff.EditStaff"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.Staff"  mc:Ignorable="d"  WindowState="Maximized"  Title="Изменение сотрудника" Height="720" Width="1280">  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" x:Name="MainWindowBtn" Click="MainWindowBtn\_Click"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" />  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}" x:Name="EmploymentRecordBtn" Click="EmploymentRecordBtn\_Click"/>  <Button Content="Приказы" x:Name="OrdersBtn" Click="OrdersBtn\_Click" Style="{StaticResource ButtonMain}"/>  <Button Content="Отчеты" x:Name="ReportsBtn" Click="ReportsBtn\_Click" Style="{StaticResource ButtonMain}"/>  <Button Content="Отмена" x:Name="CancelBtn" Click="CancelBtn\_Click" Grid.Row="0" Width="86" Height="32" HorizontalAlignment="Left" Margin="27,0,0,0" Style="{StaticResource MainButton}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="Namelbl" Content="Имя Фамилия" FontSize="14" Margin="0,0,32,0"/>  <Label x:Name="Rolelbl" Content="Должность" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image Width="32" Height="32"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>    <Grid>  <StackPanel Orientation="Horizontal" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <!--Шаг 1 (левая часть окна)-->  <StackPanel Width="560" Margin="30">  <Grid>  <Label Content="Добавить сотрудника в компанию" FontWeight="SemiBold" FontSize="18" />  <Label Content="Шаг 1" FontSize="18" FontWeight="SemiBold" HorizontalAlignment="Right"/>  </Grid>  <Label Content="Личные данные" FontSize="16" Margin="0, 0, 0, 0"/>  <!--Аватар + ФИО-->  <StackPanel Orientation="Horizontal" Margin="0, 0, 0, 15">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Width="174">  <Label Content="Аватар" FontSize="12" HorizontalAlignment="Left" Margin="0, 0, 0, 10"/>  <Image x:Name="AvatarImage" Width="80" Height="80" Source="../Media/template.png" HorizontalAlignment="Left" Margin="10, 0, 0, 0"/>  <Button Width="20" Height="20" x:Name="ChangeImage" Click="ChangeImage\_Click" >  <Image Source="C:\Users\Lenovo L530\source\repos\PersonnelDepartment\ImageBtn\ImgBtn.png" />  </Button>  </StackPanel>  <!--Имя, Фамилия, Отчество-->  <StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Имя" FontSize="12"/>  <TextBox x:Name="FirstNameTextBox" Style="{StaticResource CenteredTextBox}" Width="174" Height="32"/>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left">  <Label Content="Фамилия" FontSize="12"/>  <TextBox x:Name="SurnameTextBox" Style="{StaticResource CenteredTextBox}" Width="174" Height="32"/>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Отчество" FontSize="12"/>  <TextBox x:Name="PatronymicTextBox" Style="{StaticResource CenteredTextBox}" Width="366" Height="32"/>  </StackPanel>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Дата рождения" FontSize="12"/>  <DatePicker x:Name="DateOfBirthPicker" Width="174" Height="32"/>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left">  <Label Content="Место рождения" FontSize="12"/>  <TextBox x:Name="BirthplaceTextBox" Style="{StaticResource CenteredTextBox}" Width="346" Height="32"/>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical" Margin="0,0,0,10">  <Label Content="Адрес регистрации" FontSize="12"/>  <TextBox x:Name="RegistrationAddressTextBox" Style="{StaticResource CenteredTextBox}" HorizontalAlignment="Left" Width="539" Height="32"/>  </StackPanel>  <StackPanel>  <Label Content="Контактные данные" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Margin="0,0,0,10">  <Label Content="Телефонный номер" FontSize="12"/>  <TextBox HorizontalAlignment="Left" x:Name="TelephoneTextBox" Style="{StaticResource CenteredTextBox}" Width="262" Height="32"/>  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Электронная почта" FontSize="12"/>  <TextBox x:Name="EmailTextBox" Style="{StaticResource CenteredTextBox}" Width="261" Height="32"/>  </StackPanel>  </StackPanel>  <StackPanel>  <Label Content="Трудовые данные" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel>  <Label Content="Отдел" FontSize="12"/>  <ComboBox HorizontalAlignment="Left" Width="262" x:Name="DepartmantCmb" SelectionChanged="DepartmantCmb\_SelectionChanged" Height="32" />  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Должность" FontSize="12"/>  <ComboBox DisplayMemberPath="Title" SelectedValuePath="ID" Width="261" Height="32" x:Name="PostCmd" />  </StackPanel>  </StackPanel>    </StackPanel>  <!--Шаг 2 (правая часть окна)-->  <StackPanel>  <StackPanel>  <Label Content="Шаг 2" FontSize="18" FontWeight="SemiBold" Margin="0,29,56,0"/>  </StackPanel>  <StackPanel>  <Label Content="Паспортные данные" FontSize="16" Margin="0,0,0,0"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel>  <Label Content="Серия" FontSize="12" Margin="0,0,0,3"/>  <TextBox x:Name="PassportSeriesTextBox" Style="{StaticResource CenteredTextBox}" VerticalAlignment="Top" HorizontalAlignment="Left" Width="262" Height="32" Margin="0,0,0,0"/>  </StackPanel>  <StackPanel Margin="19,0,0,10">  <Label Content="Номер" FontSize="12" Margin="0,0,0,3"/>  <TextBox x:Name="PassportNumberTextBox" Style="{StaticResource CenteredTextBox}" VerticalAlignment="Top" Width="262" Height="32" />  </StackPanel>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">  <StackPanel>  <Label Content="Дата выдачи" FontSize="12" Margin="0,0,0,3"/>  <DatePicker x:Name="DateOfIssuePicker" VerticalAlignment="Top" HorizontalAlignment="Left" Width="174" Height="32" Margin="0,0,0,0"/>  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Кем выдан" FontSize="12" Margin="0,0,0,3"/>  <TextBox x:Name="PassportIssuedByTextBox" Style="{StaticResource CenteredTextBox}" HorizontalAlignment="Left" Width="351" Height="32" Margin="0,0,0,0"/>  </StackPanel>  </StackPanel>  <StackPanel>  <Label Content="Образование" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">  <StackPanel>  <Label Content="Уровень" FontSize="12"/>  <ComboBox x:Name="EducationLevelComboBox" HorizontalAlignment="Left" Width="265" Height="32" />  </StackPanel>  <StackPanel Margin="19,0,0,0">  <Label Content="Учебное заведение" FontSize="12"/>  <TextBox x:Name="EducationInstitutionTextBox" Style="{StaticResource CenteredTextBox}" Width="262" Height="32" />  </StackPanel>  </StackPanel>  <StackPanel Margin="0,0,0,10">  <Label Content="Прочие данные" FontSize="16"/>  </StackPanel>  <StackPanel Orientation="Horizontal" Margin="0,0,0,10">  <Label Content="Есть дети?"/>  <RadioButton x:Name="ChildrenRadioButton"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content="Военнообязанный?"/>  <RadioButton x:Name="MilitaryServiceRadioButton" />  </StackPanel>  <StackPanel Orientation="Horizontal">  <Button Height="32" Width="182" Margin="0,150,0,0" VerticalAlignment="Bottom" Style="{StaticResource MainButton}" x:Name="SaveEditBtn" Content="Редактировать" Click="SaveEditBtn\_Click"/>  <Button Height="32" Width="182" Margin="15,150,0,0" VerticalAlignment="Bottom" Style="{StaticResource MainButton}" Content="Удалить" x:Name="DeleteBtn" Click="DeleteBtn\_Click"/>  </StackPanel>  </StackPanel>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# EditStaff.xaml.cs в таблице Б.12.

Таблица Б.12 – Листинг кода EditStaff.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.ControlClasses;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  using System;  using System.Data.Entity;  using System.IO;  using System.Linq;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.Staff  {  public partial class EditStaff : Window  {  private HumanResourcesDepartmentEntities dbModel;  private Employee \_employee;  private Personal\_card \_currentUser;  public event EventHandler EmployeeUpdated;  public event EventHandler EmployeeDeleted;  public EditStaff(Employee employee, Personal\_card currentUser)  {  InitializeComponent();  \_currentUser = currentUser;    \_employee = employee;  LoadDepartmentTitles();  LoadPostTitles();  DisplayUserInfo();  LoadEducationLevels(); // Загружаем уровни образования  if (\_employee != null)  {  LoadEmployeeData(\_employee.ID); // Загружаем данные сотрудника  }  LoadDepartmentTitles();  LoadPostTitles();  }  private void DisplayUserInfo()  {  if (\_currentUser != null)  {  Namelbl.Content = $"{\_currentUser.Surname} {\_currentUser.Name}";  Rolelbl.Content = \_currentUser.Post.Title;  }  }  private void LoadDepartmentTitles()  {  dbModel = new HumanResourcesDepartmentEntities();  var departmentTitles = dbModel.Department.Select(d => d.Title).ToList();  DepartmantCmb.ItemsSource = departmentTitles;  }  private void LoadPostTitles()  {  dbModel = new HumanResourcesDepartmentEntities();  var postTitles = dbModel.Post  .Select(p => new { p.ID, p.Title })  .ToList();  PostCmd.ItemsSource = postTitles;  PostCmd.DisplayMemberPath = "Title"; // Устанавливаем отображаемое значение  PostCmd.SelectedValuePath = "ID"; // Устанавливаем идентификатор для выбора  }  private void LoadEmployeeData(int employeeId)  {  var employeeData = AppConnect.dbModel.Personal\_card  .Include(e => e.Education)  .Include("Post.Department")  .FirstOrDefault(e => e.ID == employeeId);  if (employeeData != null)  {  // Личные данные  SurnameTextBox.Text = employeeData.Surname;  FirstNameTextBox.Text = employeeData.Name;  PatronymicTextBox.Text = employeeData.Patronymic;  DateOfBirthPicker.SelectedDate = employeeData.Date\_of\_birth;  BirthplaceTextBox.Text = employeeData.Birthplace;  RegistrationAddressTextBox.Text = employeeData.Registration\_address;  // Привязка Email  EmailTextBox.Text = employeeData.Email;  // Контактные данные  TelephoneTextBox.Text = employeeData.Telephone;  // Устанавливаем выбранный отдел и загружаем должности для него  if (employeeData.Post?.Department != null)  {  DepartmantCmb.SelectedItem = employeeData.Post.Department.Title;  LoadPostsForDepartment(employeeData.Post.Department.Title);  }  // Устанавливаем выбранную должность  if (employeeData.Post != null)  {  PostCmd.SelectedValue = employeeData.Post.ID;  }  // Устанавливаем уровень образования  if (employeeData.Education != null)  {  EducationLevelComboBox.SelectedValue = employeeData.Id\_education;  }  // Привязка учебного заведения  EducationInstitutionTextBox.Text = employeeData.EducationInstitution;  // Паспортные данные  if (!string.IsNullOrEmpty(employeeData.Series\_and\_number))  {  if (employeeData.Series\_and\_number.Length >= 4)  {  PassportSeriesTextBox.Text = employeeData.Series\_and\_number.Substring(0, 4);  PassportNumberTextBox.Text = employeeData.Series\_and\_number.Substring(4);  }  else  {  PassportSeriesTextBox.Text = employeeData.Series\_and\_number;  PassportNumberTextBox.Text = string.Empty;  }  }  else  {  PassportSeriesTextBox.Text = string.Empty;  PassportNumberTextBox.Text = string.Empty;  }  PassportIssuedByTextBox.Text = employeeData.Issued\_by\_whom;  DateOfIssuePicker.SelectedDate = employeeData.Date\_of\_issue;  // Прочие данные  ChildrenRadioButton.IsChecked = employeeData.Children;  MilitaryServiceRadioButton.IsChecked = employeeData.Military\_service;  // Фото сотрудника  if (employeeData.Photo != null && employeeData.Photo.Length > 0)  {  BitmapImage bitmap = new BitmapImage();  using (MemoryStream stream = new MemoryStream(employeeData.Photo))  {  bitmap.BeginInit();  bitmap.StreamSource = stream;  bitmap.CacheOption = BitmapCacheOption.OnLoad;  bitmap.EndInit();  }  AvatarImage.Source = bitmap;  }  else  {  AvatarImage.Source = null;  }  }  else  {  MessageBox.Show("Сотрудник с таким ID не найден.");  }  }  private void ReportsBtn\_Click(object sender, RoutedEventArgs e)  {  MainReports mainReportswindow = new MainReports(\_currentUser);  mainReportswindow.Show();  Close();  }  private void CancelBtn\_Click(object sender, RoutedEventArgs e)  {  MainStaff mainStaff = new MainStaff(\_currentUser);  mainStaff.Show();  Close();  }  private void SaveEditBtn\_Click(object sender, RoutedEventArgs e)  {    var employeeData = AppConnect.dbModel.Personal\_card.FirstOrDefault(x => x.ID == \_employee.ID);  if (employeeData != null)  {  // Личные данные  employeeData.Surname = SurnameTextBox.Text;  employeeData.Name = FirstNameTextBox.Text;  employeeData.Patronymic = PatronymicTextBox.Text;  employeeData.Date\_of\_birth = DateOfBirthPicker.SelectedDate ?? DateTime.MinValue;  employeeData.Birthplace = BirthplaceTextBox.Text;  employeeData.Registration\_address = RegistrationAddressTextBox.Text;  // Контактные данные  employeeData.Telephone = TelephoneTextBox.Text;  employeeData.Email = EmailTextBox.Text;  // Обновление изображения, если оно было изменено  if (\_updatedPhoto != null)  {  employeeData.Photo = \_updatedPhoto;  }  // Сохранение изменений  AppConnect.dbModel.SaveChanges();  // Трудовые данные: проверка типа SelectedValue и приведение к int  if (PostCmd.SelectedValue != null)  {  if (PostCmd.SelectedValue is int selectedPostId)  {  employeeData.Id\_post = selectedPostId;  }  else  {  MessageBox.Show("Ошибка: SelectedValue не является корректным ID.");  }  }  else  {  MessageBox.Show("Ошибка: SelectedValue является null.");  }  // Образование  if (EducationLevelComboBox.SelectedValue is int selectedEducationId)  {  employeeData.Id\_education = selectedEducationId;  }  employeeData.EducationInstitution = EducationInstitutionTextBox.Text;  // Паспортные данные  employeeData.Series\_and\_number = PassportSeriesTextBox.Text + PassportNumberTextBox.Text;  employeeData.Issued\_by\_whom = PassportIssuedByTextBox.Text;  employeeData.Date\_of\_issue = DateOfIssuePicker.SelectedDate ?? DateTime.MinValue;  // Прочие данные  employeeData.Children = ChildrenRadioButton.IsChecked ?? false;  employeeData.Military\_service = MilitaryServiceRadioButton.IsChecked ?? false;  // Сохраняем изменения  AppConnect.dbModel.SaveChanges();  // Вывод сообщения об успешном сохранении  MessageBox.Show("Данные успешно сохранены.", "Сохранение", MessageBoxButton.OK, MessageBoxImage.Information);  }  else  {  // Сообщение, если данные сотрудника не найдены  MessageBox.Show("Сотрудник с таким ID не найден.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  }  private void LoadEducationLevels()  {  // Загружаем все уровни образования из таблицы Education  var educationLevels = AppConnect.dbModel.Education  .Select(e => new { e.ID, e.Levell }) // Получаем Id и Levell  .ToList();  EducationLevelComboBox.ItemsSource = educationLevels;  EducationLevelComboBox.DisplayMemberPath = "Levell"; // Указываем, что будет отображаться  EducationLevelComboBox.SelectedValuePath = "ID"; // Указываем идентификатор для выбора  }  private void DepartmantCmb\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (DepartmantCmb.SelectedItem != null)  {  string selectedDepartment = DepartmantCmb.SelectedItem.ToString();  LoadPostsForDepartment(selectedDepartment);  }  }  private void LoadPostsForDepartment(string departmentTitle)  {  var department = dbModel.Department.FirstOrDefault(d => d.Title == departmentTitle);  if (department != null)  {  var postTitles = dbModel.Post  .Where(p => p.Id\_department == department.ID) // Фильтруем должности по выбранному отделу  .Select(p => new { p.ID, p.Title })  .ToList();  PostCmd.ItemsSource = postTitles;  PostCmd.DisplayMemberPath = "Title";  PostCmd.SelectedValuePath = "ID";  }  }  private void MainWindowBtn\_Click(object sender, RoutedEventArgs e)  {  MainWindowInspector mainWindowInspector = new MainWindowInspector(\_currentUser);  mainWindowInspector.Show();  Close();  }  private void EmploymentRecordBtn\_Click(object sender, RoutedEventArgs e)  {  MainEmploymentRecord mainEmploymentRecord = new MainEmploymentRecord(\_currentUser);  mainEmploymentRecord.Show();  Close();  }  private void OrdersBtn\_Click(object sender, RoutedEventArgs e)  {  MainOrders mainOrderswindow = new MainOrders(\_currentUser);  mainOrderswindow.Show();  Close();  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  private void DeleteEmployee(int employeeId)  {  using (var transaction = AppConnect.dbModel.Database.BeginTransaction())  {  try  {  // Удаляем записи из таблицы Salary, связанные с сотрудником  var salariesToDelete = AppConnect.dbModel.Salary  .Where(s => s.Id\_personal\_card == employeeId).ToList();  AppConnect.dbModel.Salary.RemoveRange(salariesToDelete);  // Удаляем записи из таблицы Entry\_in\_the\_work\_book, связанные с сотрудником  var entriesToDelete = AppConnect.dbModel.Entry\_in\_the\_work\_book  .Where(e => e.Id\_personal\_card == employeeId).ToList();  AppConnect.dbModel.Entry\_in\_the\_work\_book.RemoveRange(entriesToDelete);  // Удаляем запись о сотруднике из Personal\_card  var employeeToDelete = AppConnect.dbModel.Personal\_card  .FirstOrDefault(e => e.ID == employeeId);  if (employeeToDelete != null)  {  AppConnect.dbModel.Personal\_card.Remove(employeeToDelete);  }  else  {  throw new Exception("Сотрудник с таким ID не найден.");  }  // Сохраняем изменения в базе данных  AppConnect.dbModel.SaveChanges();  transaction.Commit();  // Уведомляем об успешном удалении  MessageBox.Show("Данные сотрудника успешно удалены.", "Удаление", MessageBoxButton.OK, MessageBoxImage.Information);  EmployeeDeleted?.Invoke(this, EventArgs.Empty);  MainStaff mainStaff = new MainStaff(\_currentUser);  Close();  mainStaff.Show();  }  catch (Exception ex)  {  transaction.Rollback();  MessageBox.Show($"Ошибка при удалении данных сотрудника: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  }  }  private void DeleteBtn\_Click(object sender, RoutedEventArgs e)  {  if (\_employee != null)  {  DeleteEmployee(\_employee.ID);  }  else  {  MessageBox.Show("Сотрудник не выбран.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);  }  }  // Поле для хранения измененного изображения  private byte[] \_updatedPhoto;  private void ChangeImage\_Click(object sender, RoutedEventArgs e)  {  // Открытие диалогового окна для выбора изображения  Microsoft.Win32.OpenFileDialog openFileDialog = new Microsoft.Win32.OpenFileDialog  {  Filter = "Image files (\*.png;\*.jpeg;\*.jpg)|\*.png;\*.jpeg;\*.jpg|All files (\*.\*)|\*.\*"  };  if (openFileDialog.ShowDialog() == true)  {  // Загрузка изображения в элемент AvatarImage  BitmapImage bitmap = new BitmapImage();  bitmap.BeginInit();  bitmap.UriSource = new Uri(openFileDialog.FileName);  bitmap.CacheOption = BitmapCacheOption.OnLoad;  bitmap.EndInit();  AvatarImage.Source = bitmap;  // Преобразование изображения в массив байтов для сохранения  using (MemoryStream stream = new MemoryStream())  {  JpegBitmapEncoder encoder = new JpegBitmapEncoder();  encoder.Frames.Add(BitmapFrame.Create(bitmap));  encoder.Save(stream);  // Сохраняем байты в поле \_updatedPhoto  \_updatedPhoto = stream.ToArray();  }  }  }  }  } |

Код ReportsForm.xaml в таблице Б.13.

Таблица Б.13 – Листинг кода ReportsForm.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.RepostsForm.MainReports"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.RepostsForm"  mc:Ignorable="d"  WindowState="Maximized"  Title="Отчёты" Height="720" Width="1280">  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" x:Name="MainWindowBtn" Click="MainWindowBtn\_Click"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" x:Name="StaffBtn" Click="StaffBtn\_Click" />  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}" x:Name="EmploymentRecordBtn" Click="EmploymentRecordBtn\_Click"/>  <Button Content="Приказы" Style="{StaticResource ButtonMain}" x:Name="OrdersBtn" Click="OrdersBtn\_Click"/>  <Button Content="Отчеты" Style="{StaticResource ButtonMain}"/>  <Button Content="Отмена" Grid.Row="0" Width="86" Height="32" HorizontalAlignment="Left" Margin="27,0,0,0" Style="{StaticResource MainButton}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="Namelbl" Content="Имя Фамилия" FontSize="14" Margin="0,0,32,0"/>  <Label x:Name="Rolelbl" Content="Должность" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image Width="32" Height="32"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>  <Grid>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center" HorizontalAlignment="Center">  <Label Content="Добавить отчет" FontSize="18" Margin="0,29,0,15"/>  <Label Content="Выбрать тип отчета" FontSize="16" Margin="0,0,0,15"/>  <ComboBox Width="367" SelectionChanged="ReportsCmd\_SelectionChanged" x:Name="ReportsCmd" Height="32">  <ComboBoxItem Content="Список сотрудников"/>  <ComboBoxItem Content="Список сотрудников по отделам"/>  <ComboBoxItem Content="Список сотрудников с детьми"/>  <ComboBoxItem Content="Список военнообязанных сотрудников"/>  </ComboBox>  <Label Margin="0,15,0,9" FontSize="16" Content="Период отчета"/>  <StackPanel Margin="0,0,0,7" Orientation="Horizontal">  <Label Margin="0,0,82,0" Content="Начало периода"/>  <Label Content="Конец периода"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <DatePicker x:Name="StartDatePicker" Width="173" Height="32" Margin="0,0,14,0"/>  <DatePicker x:Name="EndDatePicker" Width="173" Height="32"/>  </StackPanel>  <StackPanel x:Name="PostSkP" Margin="0,15,0,15" Visibility="Collapsed">  <Label FontSize="16" Margin="0,0,0,15" Content="Выбрать отдел"/>  <ComboBox Width="367" Height="32" x:Name="PostCmd"/>  </StackPanel>  </StackPanel>    </StackPanel>  <StackPanel VerticalAlignment="Bottom">  <Button Width="276" Height="32" Margin="0,300,0,39" Style="{StaticResource MainButton}" x:Name="ReportBtn" Click="ReportBtn\_Click" Content="Просмотр в Excel"/>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# ReportstForm.xaml.cs в таблице Б.14.

Таблица Б.14 – Листинг кода ReportstForm.xaml.cs.

|  |
| --- |
| using ClosedXML.Excel;  using Dapper;  using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.Staff;  using System;  using System.Collections.Generic;  using System.Data;  using System.Data.SqlClient;  using System.Windows;  using System.Windows.Controls;  namespace PersonnelDepartment.RepostsForm  {  /// <summary>  /// Логика взаимодействия для MainReports.xaml  /// </summary>  public partial class MainReports : Window  {  private string connectionString = "data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework";  private Personal\_card \_currentUser; // Хранение данных текущего пользователя  public MainReports(Personal\_card currentUser)  {  \_currentUser = currentUser;  InitializeComponent();  LoadDepartmentsIntoComboBox();  DisplayUserInfo(); // Отображение информации о текущем пользователе  }  private void DisplayUserInfo()  {  if (\_currentUser != null)  {  Namelbl.Content = $"{\_currentUser.Surname} {\_currentUser.Name}";  Rolelbl.Content = \_currentUser.Post.Title;  }  }  public class Department  {  public int ID { get; set; }  public string Title { get; set; }  }  private void LoadDepartmentsIntoComboBox()  {  List<Department> departments = LoadDepartments();  // Привязка данных к ComboBox  PostCmd.ItemsSource = departments;  PostCmd.DisplayMemberPath = "Title"; // Отображаемое название  PostCmd.SelectedValuePath = "ID"; // Значение элемента (ID отдела)  }  public List<Department> LoadDepartments()  {  using (var connection = new SqlConnection(connectionString))  {  connection.Open();  string query = "SELECT ID, Title FROM Department";  return connection.Query<Department>(query).AsList();  }  }  private void StaffBtn\_Click(object sender, RoutedEventArgs e)  {  new MainStaff(\_currentUser).Show();  Close();  }  private void ReportsCmd\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (ReportsCmd.SelectedIndex == 1)  {  PostSkP.Visibility = Visibility.Visible;  }  else  {  PostSkP.Visibility = Visibility.Collapsed;  }  }  private void ReportBtn\_Click(object sender, RoutedEventArgs e)  {  try  {  int reportType = ReportsCmd.SelectedIndex + 1;  DateTime startDate = StartDatePicker.SelectedDate ?? DateTime.Now;  DateTime endDate = EndDatePicker.SelectedDate ?? DateTime.Now;  string department = null;  int? departmentId = null;  if (reportType == 2 && PostCmd.SelectedValue != null)  {  departmentId = (int)PostCmd.SelectedValue;  department = (PostCmd.SelectedItem as Department)?.Title;  }  using (var connection = new SqlConnection(connectionString))  {  connection.Open();  List<Employee> employees = GetEmployeeData(connection, reportType, startDate, endDate, departmentId);  GenerateReport(employees, reportType, startDate, endDate, department);  }  MessageBox.Show("Отчёт успешно создан и сохранён как 'Отчёт.xlsx'.");  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка: {ex.Message}");  }  }  public List<Employee> GetEmployeeData(IDbConnection connection, int reportType, DateTime startDate, DateTime endDate, int? departmentId)  {  string query;  switch (reportType)  {  case 1:  query = @"SELECT Personal\_card.ID,  Personal\_card.Name AS FirstName,  Personal\_card.Surname AS LastName,  Personal\_card.Patronymic,  Post.Title AS Position,  Department.Title AS Department,  Personal\_card.Telephone AS Phone,  Personal\_card.Date\_of\_birth AS BirthDate  FROM Personal\_card  JOIN Post ON Personal\_card.Id\_post = Post.ID  JOIN Department ON Post.Id\_department = Department.ID";  break;  case 2:  query = @"SELECT Personal\_card.ID,  Personal\_card.Name AS FirstName,  Personal\_card.Surname AS LastName,  Personal\_card.Patronymic,  Post.Title AS Position,  Department.Title AS Department,  Personal\_card.Telephone AS Phone,  Personal\_card.Date\_of\_birth AS BirthDate,  Entry\_in\_the\_work\_book.Date AS DateOfHire  FROM Personal\_card  JOIN Post ON Personal\_card.Id\_post = Post.ID  JOIN Department ON Post.Id\_department = Department.ID  JOIN Entry\_in\_the\_work\_book ON Entry\_in\_the\_work\_book.Id\_personal\_card = Personal\_card.ID  JOIN Mixing ON Entry\_in\_the\_work\_book.Id\_mixing = Mixing.ID  WHERE Department.ID = @DepartmentId AND Mixing.Title = 'Приём'";  break;  case 3:  query = @"SELECT Personal\_card.ID,  Personal\_card.Name AS FirstName,  Personal\_card.Surname AS LastName,  Personal\_card.Patronymic,  Post.Title AS Position,  Department.Title AS Department,  Personal\_card.Telephone AS Phone,  Personal\_card.Date\_of\_birth AS BirthDate  FROM Personal\_card  JOIN Post ON Personal\_card.Id\_post = Post.ID  JOIN Department ON Post.Id\_department = Department.ID  WHERE Personal\_card.Children > 0";  break;  case 4:  query = @"SELECT Personal\_card.ID,  Personal\_card.Name AS FirstName,  Personal\_card.Surname AS LastName,  Personal\_card.Patronymic,  Post.Title AS Position,  Department.Title AS Department,  Personal\_card.Telephone AS Phone,  Personal\_card.Date\_of\_birth AS BirthDate  FROM Personal\_card  JOIN Post ON Personal\_card.Id\_post = Post.ID  JOIN Department ON Post.Id\_department = Department.ID  WHERE Personal\_card.Military\_service = 1";  break;  default:  throw new ArgumentException("Неверный тип отчёта");  }  return connection.Query<Employee>(query, new { DepartmentId = departmentId }).AsList();  }  public void GenerateReport(List<Employee> employees, int reportType, DateTime startDate, DateTime endDate, string department)  {  string templatePath = GetTemplatePath(reportType);  var workbook = new XLWorkbook(templatePath);  var worksheet = workbook.Worksheet(1);  // Установка заголовка отчёта с указанием номера кейса  worksheet.Cell("A1").Value = $"Отчёт ({reportType})";  // Установка начала и конца периода  worksheet.Cell("B9").Value = startDate.ToShortDateString();  worksheet.Cell("C9").Value = endDate.ToShortDateString();  // Установка названия отдела, если это отчёт по отделам  if (reportType == 2 && department != null)  {  worksheet.Cell("C6").Value = department;  }  // Начальная строка для записей сотрудников  int currentRow = 15;  foreach (var employee in employees)  {  worksheet.Cell($"A{currentRow}").Value = $"{employee.LastName} {employee.FirstName} {employee.Patronymic}";  if (reportType == 1) // Список сотрудников  {  worksheet.Range($"B{currentRow}:C{currentRow}").Merge().Value = employee.Position;  worksheet.Range($"D{currentRow}:E{currentRow}").Merge().Value = employee.Department;  }  else if (reportType == 2) // Список сотрудников по отделам  {  worksheet.Cell($"B{currentRow}").Value = employee.Position;  worksheet.Cell($"C{currentRow}").Value = employee.BirthDate.ToShortDateString();  // Убираем заполнение столбца D для названия отдела  }  else if (reportType == 3) // Список сотрудников с детьми  {  worksheet.Cell($"B{currentRow}").Value = employee.BirthDate.ToShortDateString();  worksheet.Cell($"C{currentRow}").Value = employee.Position;  worksheet.Cell($"D{currentRow}").Value = employee.Department;  }  else if (reportType == 4) // Список военнообязанных сотрудников  {  worksheet.Cell($"B{currentRow}").Value = employee.BirthDate.ToShortDateString();  worksheet.Cell($"C{currentRow}").Value = employee.Department;  worksheet.Cell($"D{currentRow}").Value = employee.Position;  }  currentRow++;  }  // Добавление строки с общим количеством сотрудников в отделе для отчета типа 2  if (reportType == 2)  {  worksheet.Cell($"C{currentRow}").Value = "Общее количество:";  worksheet.Cell($"D{currentRow}").Value = employees.Count;  }  // Диалоговое окно для сохранения файла  var saveFileDialog = new Microsoft.Win32.SaveFileDialog  {  Filter = "Excel Files|\*.xlsx",  Title = "Сохранить отчёт",  FileName = $"Отчёт\_{reportType}.xlsx"  };  if (saveFileDialog.ShowDialog() == true)  {  workbook.SaveAs(saveFileDialog.FileName);  MessageBox.Show("Отчёт успешно сохранён.", "Сохранение отчёта", MessageBoxButton.OK, MessageBoxImage.Information);  }  else  {  MessageBox.Show("Сохранение отменено.", "Сохранение отчёта", MessageBoxButton.OK, MessageBoxImage.Warning);  }  }  private static string GetTemplatePath(int reportType)  {  string templatePath;  switch (reportType)  {  case 1:  templatePath = "Templates/ОтчётСотрудники.xlsx";  break;  case 2:  templatePath = "Templates/ОтчётОтдел.xlsx";  break;  case 3:  templatePath = "Templates/ОтчётДети.xlsx";  break;  case 4:  templatePath = "Templates/ОтчётВоеннообязанные.xlsx";  break;  default:  throw new ArgumentException("Неверный тип отчёта");  }  return templatePath;  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  private void OrdersBtn\_Click(object sender, RoutedEventArgs e)  {  MainOrders mainOrders = new MainOrders(\_currentUser);  mainOrders.Show();  Close();  }  private void EmploymentRecordBtn\_Click(object sender, RoutedEventArgs e)  {  MainEmploymentRecord mainEmploymentRecord = new MainEmploymentRecord(\_currentUser);  mainEmploymentRecord.Show();  Close();  }  private void MainWindowBtn\_Click(object sender, RoutedEventArgs e)  {  MainWindowInspector mainWindowInspector = new MainWindowInspector(\_currentUser);  mainWindowInspector.Show();  Close();  }  }  } |

Код OrdersForm.xaml в таблице Б.15.

Таблица Б.15 – Листинг кода OrdersForm.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.OrdersForm.MainOrders"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.OrdersForm"  WindowState="Maximized"  mc:Ignorable="d"  Title="Приказы" Height="720" Width="1280">  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" x:Name="MainWindowBtn" Click="MainWindowBtn\_Click"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" x:Name="StaffBtn" Click="StaffBtn\_Click" />  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}" x:Name="EmploymentrecordBtn" Click="EmploymentrecordBtn\_Click"/>  <Button Content="Приказы" Style="{StaticResource ButtonMain}"/>  <Button Content="Отчеты" Style="{StaticResource ButtonMain}" x:Name="ReportsBtn" Click="ReportsBtn\_Click"/>  <Button Content="Отмена" Grid.Row="0" Width="86" Height="32" HorizontalAlignment="Left" Margin="27,0,0,0" Style="{StaticResource MainButton}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="UserName" Content="Имя Фамилия" FontSize="14" Margin="0,0,32,0"/>  <Label x:Name="UserRole" Content="Должность" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image x:Name="UserImage" Width="32" Height="32"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>  <Grid>    <StackPanel Grid.Row="0" Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center" HorizontalAlignment="Center">  <Label Content="Добавить приказ" FontWeight="SemiBold" FontSize="18" Margin="0,29,0,15"/>  <Label Content="Выбрать сотрудника" FontSize="16" Margin="0,0,0,11"/>  <ComboBox x:Name="EmployeeComboBox" SelectionChanged="EmployeeComboBox\_SelectionChanged" Width="367" Height="32" Margin="0,0,0,15"/>  <Label Content="Выбрать тип приказа" FontSize="16" Margin="0,0,0,15"/>  <ComboBox Width="367" SelectionChanged="OrderTypeComboBox\_SelectionChanged" x:Name="OrderTypeComboBox" Height="32">  </ComboBox>  <Grid x:Name="gridOrders" Visibility="Collapsed">  <!-- Для приказа на изменение зарплаты -->  <StackPanel x:Name="OptionSalary" Visibility="Collapsed">  <Label Content="Старая зарплата: " FontSize="16"/>  <Label x:Name="CurrentBaseSarary" FontSize="14" Margin="0,5,0,15"/>  <Label Content="Новая зарплата:" FontSize="16"/>  <TextBox x:Name="NewSalaryTextBox" Style="{StaticResource CenteredTextBox}" Width="367" Height="32" Margin="0,5,0,15"/>  </StackPanel>  <!-- Для приказа на перевод: Отображение текущей должности и выбор новой должности -->  <StackPanel x:Name="OptionPosition" Visibility="Collapsed">  <Label Content="Текущая должность сотрудника:" FontSize="16"/>  <Label x:Name="CurrentPositionLabel" FontSize="14" Margin="0,5,0,15"/>  <Label Content="Выбрать новую должность" FontSize="16"/>  <ComboBox x:Name="NewPositionComboBox" Width="367" Height="32" Margin="0,5,0,15"/>  </StackPanel>  <!-- Для приказа на перевод: Отображение текущего отдела и выбор нового отдела -->  <StackPanel x:Name="OptionDepartment" Visibility="Collapsed">  <Label Content="Текущий отдел сотрудника:" FontSize="16"/>  <Label x:Name="CurrentDepartmentLabel" FontSize="14" Margin="0,5,0,15"/>  <Label Content="Выбрать новый отдел" FontSize="16"/>  <ComboBox x:Name="NewDepartmentComboBox" Width="367" SelectionChanged="NewDepartmentComboBox\_SelectionChanged" Height="32" Margin="0,5,0,15"/>  <Label Content="Новая должность: " FontSize="16"/>  <ComboBox x:Name="NewPostCmd" Width="367" Height="32" Margin="0,5,0,15"/>  </StackPanel>  <!-- Для приказа на премию -->  <StackPanel x:Name="OptionBonus" Visibility="Collapsed">  <Label Content="Тип премии" FontSize="16"/>  <ComboBox x:Name="BonusTypeComboBox" Width="367" Height="32" Margin="0,15,0,0"/>  <Label Content="Сумма премии" FontSize="16" Margin="10,0,0,5"/>  <TextBox x:Name="BonusAmountTextBox" Style="{StaticResource CenteredTextBox}" Width="367" Height="32"/>  </StackPanel>  </Grid>  </StackPanel>    </StackPanel>  <StackPanel VerticalAlignment="Bottom" Orientation="Vertical">  <Button Width="276" Height="32" Margin="0,0,0,39" Style="{StaticResource MainButton}" Content="Создать приказ" x:Name="SaveToExcel" Click="SaveToExcel\_Click"/>  </StackPanel>    </Grid>  </Grid>  </Window> |

Код C# OrdersForm.xaml.cs в таблице Б.16.

Таблица Б.16 – Листинг кода OrdersForm.xaml.cs.

|  |
| --- |
| using ClosedXML.Excel;  using Microsoft.Win32;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.RepostsForm;  using PersonnelDepartment.Staff;  using System;  using System.IO;  using System.Linq;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.OrdersForm  {  public partial class MainOrders : Window  {  private HumanResourcesDepartmentEntities \_context;  private Personal\_card \_currentUser;  private int \_newEmployeeId;  public MainOrders(Personal\_card currentUser)  {  \_currentUser = currentUser;    InitializeComponent();  LoadData();  UserName.Content = $"{currentUser.Surname} {currentUser.Name}";  UserRole.Content = currentUser.Post.Title;    // Проверяем, что ComboBox и данные инициализированы  if (EmployeeComboBox != null && EmployeeComboBox.ItemsSource != null)  {  EmployeeComboBox.SelectedValue = \_newEmployeeId;  }  else  {  Console.WriteLine("Ошибка: EmployeeComboBox или данные не инициализированы.");  }  }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  private void LoadData()  {  \_context = new HumanResourcesDepartmentEntities();  // Загрузка сотрудников из таблицы personal\_card  var employees = \_context.Personal\_card  .Select(e => new  {  e.ID,  FullName = e.Surname + " " + e.Name + " " + e.Patronymic  })  .ToList();  EmployeeComboBox.ItemsSource = employees;  EmployeeComboBox.DisplayMemberPath = "FullName";  EmployeeComboBox.SelectedValuePath = "ID";  // Выбираем нового сотрудника по его ID  EmployeeComboBox.SelectedValue = \_newEmployeeId;  // Загрузка типов приказов из таблицы Mixing  var orderTypes = \_context.Mixing  .Select(o => new { o.ID, o.Title })  .ToList();  OrderTypeComboBox.ItemsSource = orderTypes;  OrderTypeComboBox.DisplayMemberPath = "Title";  OrderTypeComboBox.SelectedValuePath = "ID";  // Выбираем по умолчанию первый тип приказа (например, Прием на работу)  OrderTypeComboBox.SelectedIndex = 0;  }  private void StaffBtn\_Click(object sender, RoutedEventArgs e)  {  MainStaff mainStaffwindow = new MainStaff(\_currentUser);  mainStaffwindow.Show();  Close();  }  private void OrderTypeComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (OrderTypeComboBox.SelectedValue is int selectedOrderTypeId)  {  // Скрываем все дополнительные опции перед настройкой видимости  gridOrders.Visibility = Visibility.Visible;  OptionSalary.Visibility = Visibility.Collapsed;  OptionPosition.Visibility = Visibility.Collapsed;  OptionDepartment.Visibility = Visibility.Collapsed;  OptionBonus.Visibility = Visibility.Collapsed;  // Устанавливаем видимость в зависимости от выбранного типа приказа  switch (selectedOrderTypeId)  {  case 1: // Прием на работу  MessageBox.Show("Прием на работу - дополнительные поля не требуются");  break;  case 2: // Увольнение сотрудника  MessageBox.Show("Увольнение сотрудника - дополнительные поля не требуются");  break;  case 3: // Перевод на другую должность  MessageBox.Show("Отображение опций для перевода на другую должность");  OptionPosition.Visibility = Visibility.Visible;  // Загрузка текущей должности и списка новых должностей  if (EmployeeComboBox.SelectedValue is int selectedEmployeeId)  {  var employeeData = \_context.Department.FirstOrDefault(emp => emp.ID == selectedEmployeeId);  if (employeeData != null)  {  var currentPosition = \_context.Post.FirstOrDefault(pos => pos.ID == selectedEmployeeId);  CurrentPositionLabel.Content = $" {currentPosition?.Title ?? "Неизвестно"}";  }  }  NewPositionComboBox.ItemsSource = \_context.Post.Select(p => p.Title).ToList();  break;  case 4: // Перевод в другое подразделение  MessageBox.Show("Отображение опций для перевода в другое подразделение");  OptionDepartment.Visibility = Visibility.Visible;  // Загрузка текущего отдела и списка новых отделов  if (EmployeeComboBox.SelectedValue is int selectedEmployeeIdDept)  {  var employeeData = \_context.Personal\_card.FirstOrDefault(emp => emp.ID == selectedEmployeeIdDept);  var post = \_context.Post.FirstOrDefault(p => p.ID == employeeData.Id\_post);  var currentDepartment = \_context.Department.FirstOrDefault(d => d.ID == post.Id\_department);  CurrentDepartmentLabel.Content = $"Текущий отдел: {currentDepartment?.Title ?? "Неизвестно"}";  }  // Загрузка списка отделов в NewDepartmentComboBox  NewDepartmentComboBox.ItemsSource = \_context.Department.Select(d => d.Title).ToList();  NewPostCmd.ItemsSource = null; // Очищаем должности, так как отдел еще не выбран  break;  case 5: // Начисление заработной платы  MessageBox.Show("Отображение опций для начисления заработной платы");  OptionSalary.Visibility = Visibility.Visible;  if (EmployeeComboBox.SelectedValue is int selectedEmployeeIdSalary)  {  // Получение данных о текущей зарплате  var currentPostSalary = \_context.Post.FirstOrDefault(p => p.ID == selectedEmployeeIdSalary);  if (currentPostSalary != null)  {  CurrentBaseSarary.Content = $"Текущая зарплата: {currentPostSalary.Base\_salary}";  }  else  {  CurrentBaseSarary.Content = "Текущая зарплата: Неизвестно";  }  }  break;  case 6: // Поощрение  MessageBox.Show("Отображение опций для поощрения");  OptionBonus.Visibility = Visibility.Visible;  BonusTypeComboBox.ItemsSource = \_context.Salary\_type.Select(s => s.Title).ToList();  break;  default:  MessageBox.Show("Неизвестный тип приказа");  break;  }  }  }  private void SaveToExcel\_Click(object sender, RoutedEventArgs e)  {  if (EmployeeComboBox.SelectedValue == null || OrderTypeComboBox.SelectedValue == null)  {  MessageBox.Show("Пожалуйста, выберите сотрудника и тип приказа.");  return;  }  int selectedEmployeeId = (int)EmployeeComboBox.SelectedValue;  int selectedOrderTypeId = (int)OrderTypeComboBox.SelectedValue;  // Выбор шаблона на основе типа приказа  string templatePath = selectedOrderTypeId switch  {  1 => "Templates/Приказоприёме.xlsx",  2 => "Templates/Увольнение.xlsx",  3 => "Templates/Перевод должность.xlsx",  4 => "Templates/Перевод отдел.xlsx",  5 => "Templates/зп.xlsx",  6 => "Templates/приемии.xlsx",  \_ => throw new InvalidOperationException("Неизвестный тип приказа."),  };  var saveFileDialog = new SaveFileDialog  {  Filter = "Excel Files|\*.xlsx",  Title = "Сохранить приказ",  FileName = $"Приказ\_{selectedOrderTypeId}.xlsx"  };  if (saveFileDialog.ShowDialog() == true)  {  try  {  if (!File.Exists(templatePath))  {  MessageBox.Show("Шаблон Excel не найден.");  return;  }  using var workbook = new XLWorkbook(templatePath);  var worksheet = workbook.Worksheet(1);  var employeeData = \_context.Personal\_card.FirstOrDefault(x => x.ID == selectedEmployeeId);  if (employeeData == null)  {  MessageBox.Show("Сотрудник не найден.");  return;  }  var post = \_context.Post.FirstOrDefault(p => p.ID == employeeData.Id\_post);  Department department = null;  if (post != null)  {  department = \_context.Department.FirstOrDefault(d => d.ID == post.Id\_department);  }  switch (selectedOrderTypeId)  {  case 1: // Приказ о приеме на работу  // ФИО сотрудника  worksheet.Cell("A20").Value = $"{employeeData.Surname} {employeeData.Name} {employeeData.Patronymic}";  // Отдел  if (department != null)  {  worksheet.Cell("A22").Value = department.Title;  }  // Должность  if (post != null)  {  worksheet.Cell("A24").Value = post.Title; // Заполняем должность в строке A24  }  else  {  worksheet.Cell("A24").Value = "Должность не указана"; // Если должность не найдена  }  // Зарплата: берется из таблицы Salary с фильтром по salary\_type = 1  var salary = \_context.Salary  .Where(s => s.Id\_personal\_card == selectedEmployeeId && s.Id\_salary\_type == 3)  .OrderByDescending(s => s.Date)  .FirstOrDefault();  if (salary != null)  {  worksheet.Cell("D31").Value = salary.Amount; // Указываем зарплату  }  else  {  worksheet.Cell("D31").Value = "Зарплата не указана";  }  // Дата приема  var currentDate = DateTime.Now;  worksheet.Cell("D39").Value = $"{currentDate:dd.MM}";  worksheet.Cell("G39").Value = currentDate.Year;  break;  case 2: // Приказ об увольнении  // ФИО сотрудника  worksheet.Cell("A18").Value = $"{employeeData.Surname} {employeeData.Name} {employeeData.Patronymic}";  // Отдел  if (department != null)  {  worksheet.Cell("A20").Value = department.Title;  }  // Должность  if (post != null)  {  worksheet.Cell("A22").Value = post.Title;  }  // Дата увольнения  var terminationDate = DateTime.Now;  worksheet.Cell("C14").Value = $"{terminationDate:dd.MM}";  worksheet.Cell("G14").Value = terminationDate.Year;  break;  case 3: // Перевод на другую должность  // ФИО сотрудника  worksheet.Cell("A20").Value = $"{employeeData.Surname} {employeeData.Name} {employeeData.Patronymic}";  // Текущий отдел и должность  if (department != null)  {  worksheet.Cell("C23").Value = department.Title; // Отдел остается текущим  }  if (post != null)  {  worksheet.Cell("C25").Value = post.Title; // Текущая должность  }  // Получаем текущую должность из лейбла  string currentPositionTitle = CurrentPositionLabel.Content?.ToString() ?? post?.Title;  if (string.IsNullOrEmpty(currentPositionTitle))  {  MessageBox.Show("Не удалось получить текущую должность.");  return;  }  // Получаем пост и отдел на основе текущей должности  var currentPost = \_context.Post.FirstOrDefault(p => p.Title == currentPositionTitle);  Department currentDepartment = null;  if (currentPost != null)  {  currentDepartment = \_context.Department.FirstOrDefault(d => d.ID == currentPost.Id\_department);  }  if (currentDepartment != null)  {  // Фильтрация должностей по найденному отделу  var departmentPosts = \_context.Post.Where(p => p.Id\_department == currentDepartment.ID).ToList();  // Очищаем комбобокс  NewPositionComboBox.Items.Clear();  // Добавляем должности, относящиеся к этому отделу  foreach (var departmentPost in departmentPosts)  {  NewPositionComboBox.Items.Add(departmentPost.Title);  }  // При необходимости выбираем первую должность по умолчанию  if (NewPositionComboBox.Items.Count > 0)  {  NewPositionComboBox.SelectedIndex = 0;  }  }  else  {  MessageBox.Show("Не удалось найти отдел для текущей должности.");  }  // Получаем новую должность  string newPositionTitle = NewPositionComboBox.SelectedItem?.ToString() ?? post?.Title ?? "Неизвестно";  worksheet.Cell("B31").Value = newPositionTitle; // Новая должность  // Извлечение новой зарплаты (base\_salary) для выбранной должности  var newPositionData = \_context.Post.FirstOrDefault(p => p.Title == newPositionTitle);  if (newPositionData != null)  {  worksheet.Cell("D34").Value = newPositionData.Base\_salary; // Базовая зарплата новой должности  }  else  {  worksheet.Cell("D34").Value = "Зарплата не указана"; // Если данные отсутствуют  }  // Получаем новый отдел  var newDepartmentTitle = NewDepartmentComboBox.SelectedItem?.ToString() ?? currentDepartment?.Title ?? "Неизвестен";  worksheet.Cell("B29").Value = newDepartmentTitle;  // Дата перевода  var transferDate = DateTime.Now;  worksheet.Cell("D39").Value = $"{transferDate:dd.MM}";  worksheet.Cell("F39").Value = transferDate.Year;  break;  case 4: // Приказ на перевод в другое подразделение  // ФИО сотрудника  worksheet.Cell("A20").Value = $"{employeeData.Surname} {employeeData.Name} {employeeData.Patronymic}";  // Получаем текущий отдел и должность через Post и Department  var currentPost1 = \_context.Post.FirstOrDefault(p => p.ID == employeeData.Id\_post);  var currentDepartment1 = currentPost1 != null ? \_context.Department.FirstOrDefault(d => d.ID == currentPost1.Id\_department) : null;  // Заполнение текущего отдела и должности сотрудника  worksheet.Cell("C23").Value = currentDepartment1?.Title ?? "Неизвестно"; // Текущий отдел  worksheet.Cell("C25").Value = currentPost1?.Title ?? "Неизвестно"; // Текущая должность  // Получаем новый отдел и должность из ComboBox  string newDepartmentTitleForTransfer = NewDepartmentComboBox.SelectedItem?.ToString() ?? currentDepartment1?.Title ?? "Неизвестно";  string newPositionTitleForTransfer = NewPostCmd.SelectedItem?.ToString() ?? currentPost1?.Title ?? "Неизвестно";  worksheet.Cell("B29").Value = newDepartmentTitleForTransfer; // Новый отдел  worksheet.Cell("B31").Value = newPositionTitleForTransfer; // Новая должность  // Получение базовой зарплаты для новой должности  var newDepartment = \_context.Department.FirstOrDefault(d => d.Title == newDepartmentTitleForTransfer);  if (newDepartment == null)  {  MessageBox.Show($"Новый отдел '{newDepartmentTitleForTransfer}' не найден.");  break;  }  var newPost = \_context.Post.FirstOrDefault(p => p.Title == newPositionTitleForTransfer && p.Id\_department == newDepartment.ID);  if (newPost != null)  {  worksheet.Cell("D34").Value = newPost.Base\_salary; // Новая зарплата  }  else  {  worksheet.Cell("D34").Value = "Зарплата не указана"; // Если данные не найдены  }  // Дата перевода  var transferDateForDepartmentChange = DateTime.Now;  worksheet.Cell("D39").Value = $"{transferDateForDepartmentChange:dd.MM}"; // День и месяц  worksheet.Cell("F39").Value = transferDateForDepartmentChange.Year; // Год  break;  case 5: // Приказ о начислении заработной платы  if (EmployeeComboBox.SelectedValue == null)  {  MessageBox.Show("Выберите сотрудника.");  break;  }  // Получение данных сотрудника по выбранному ID  var selectedEmployeeIdSalary = (int)EmployeeComboBox.SelectedValue;  var employeeDataSalary = \_context.Personal\_card.FirstOrDefault(x => x.ID == selectedEmployeeIdSalary);  if (employeeDataSalary == null)  {  MessageBox.Show("Сотрудник не найден.");  break;  }  // Заполнение ФИО сотрудника в ячейку A16  worksheet.Cell("A16").Value = $"{employeeDataSalary.Surname} {employeeDataSalary.Name} {employeeDataSalary.Patronymic}";  // Отдел сотрудника (A18)  var currentPostForSalary = \_context.Post.FirstOrDefault(p => p.ID == employeeDataSalary.Id\_post);  var currentDepartmentForSalary = currentPostForSalary != null  ? \_context.Department.FirstOrDefault(d => d.ID == currentPostForSalary.Id\_department)  : null;  worksheet.Cell("A18").Value = currentDepartmentForSalary?.Title ?? "Отдел не указан";  // Должность сотрудника (A20)  if (currentPostForSalary != null)  {  worksheet.Cell("A20").Value = currentPostForSalary.Title; // Запись должности  }  else  {  worksheet.Cell("A20").Value = "Должность не указана";  }  // Базовая зарплата (D27) из таблицы Post  if (currentPostForSalary != null)  {  worksheet.Cell("D27").Value = currentPostForSalary.Base\_salary; // Запись зарплаты  }  else  {  worksheet.Cell("D27").Value = 0.0f; // Если должность не найдена  }  // Новая зарплата (D29) из текстбокса  if (double.TryParse(NewSalaryTextBox.Text, out double newSalaryAmount))  {  worksheet.Cell("D29").Value = newSalaryAmount;  }  else  {  MessageBox.Show("Пожалуйста, введите корректную новую зарплату.");  worksheet.Cell("D29").Value = 0.0f; // Если введенные данные некорректны  }  // Дата составления приказа (D35) - полный формат (день, месяц, год)  var salaryChangeDate = DateTime.Now;  worksheet.Cell("D35").Value = $"{salaryChangeDate:dd.MM.yyyy}";  break;  case 6: // Поощрение  if (EmployeeComboBox.SelectedValue == null)  {  MessageBox.Show("Выберите сотрудника.");  break;  }  // Получение данных сотрудника по выбранному ID  var selectedEmployeeIdBonus = (int)EmployeeComboBox.SelectedValue;  var employeeDataBonus = \_context.Personal\_card.FirstOrDefault(x => x.ID == selectedEmployeeIdBonus);  if (employeeDataBonus == null)  {  MessageBox.Show("Сотрудник не найден.");  break;  }  // Заполнение ФИО сотрудника в ячейку A20  worksheet.Cell("A20").Value = $"{employeeDataBonus.Surname} {employeeDataBonus.Name} {employeeDataBonus.Patronymic}";  // Отдел сотрудника (A22)  var currentPostForBonus = \_context.Post.FirstOrDefault(p => p.ID == employeeDataBonus.Id\_post);  var currentDepartmentForBonus = currentPostForBonus != null  ? \_context.Department.FirstOrDefault(d => d.ID == currentPostForBonus.Id\_department)  : null;  worksheet.Cell("A22").Value = currentDepartmentForBonus?.Title ?? "Отдел не указан";  // Должность сотрудника (A24)  worksheet.Cell("A24").Value = currentPostForBonus?.Title ?? "Должность не указана";  // Заполнение ComboBox только типами "Премия" (1) и "Командировочные" (2)  BonusTypeComboBox.ItemsSource = \_context.Salary\_type  .Where(t => t.ID == 1 || t.ID == 2)  .Select(t => t.Title)  .ToList();  // Проверка выбранного типа премии  if (BonusTypeComboBox.SelectedItem != null)  {  // Тип премии (D31)  worksheet.Cell("D31").Value = BonusTypeComboBox.SelectedItem.ToString();  }  else  {  MessageBox.Show("Выберите тип премии.");  worksheet.Cell("D31").Value = "Тип не выбран";  }  // Сумма премии из текстбокса (D33)  if (double.TryParse(BonusAmountTextBox.Text, out double bonusAmount))  {  worksheet.Cell("D33").Value = bonusAmount;  }  else  {  MessageBox.Show("Введите корректную сумму премии.");  worksheet.Cell("D33").Value = 0.0f; // Если введенные данные некорректны  }  // Дата премии (D39) - текущая дата (день, месяц, год)  var bonusDate = DateTime.Now;  worksheet.Cell("D39").Value = $"{bonusDate:dd.MM.yyyy}";  break;  }  // Сохранение файла  workbook.SaveAs(saveFileDialog.FileName);  MessageBox.Show("Приказ успешно сохранен.");  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при сохранении файла: {ex.Message}");  }  }  }  private void MainWindowBtn\_Click(object sender, RoutedEventArgs e)  {  MainWindowInspector mainWindowInspector = new MainWindowInspector(\_currentUser);  mainWindowInspector.Show();  Close();  }  private void EmploymentrecordBtn\_Click(object sender, RoutedEventArgs e)  {  MainEmploymentRecord mainEmploymentRecord = new MainEmploymentRecord(\_currentUser);  mainEmploymentRecord.Show();  Close();  }  private void ReportsBtn\_Click(object sender, RoutedEventArgs e)  {  MainReports mainReports = new MainReports(\_currentUser);  mainReports.Show();  Close();  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  private void NewDepartmentComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (NewDepartmentComboBox.SelectedItem is string selectedDepartmentTitle)  {  // Получаем выбранный отдел  var selectedDepartment = \_context.Department.FirstOrDefault(d => d.Title == selectedDepartmentTitle);  if (selectedDepartment != null)  {  // Фильтруем должности по выбранному отделу и заполняем NewPostCmd  NewPostCmd.ItemsSource = \_context.Post  .Where(p => p.Id\_department == selectedDepartment.ID)  .Select(p => p.Title)  .ToList();  }  else  {  // Если отдел не выбран или не найден, очищаем список должностей  NewPostCmd.ItemsSource = null;  }  }  }  private void EmployeeComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {    }  }  } |

Код AddRecordWindow.xaml в таблице Б.17.

Таблица Б.17 – Листинг кода AddRecordWindow.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.EmploymentRecord.AddRecordWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.EmploymentRecord"  mc:Ignorable="d"  WindowStartupLocation="CenterScreen"  WindowState="Normal"  Title="Добавление записи" Height="290" Width="440">  <Grid>  <StackPanel>  <Label Content="Добавление записи" FontSize="18" FontWeight="DemiBold" Margin="32,27,0,21"/>  <StackPanel Orientation="Horizontal">  <Label Content="Сведения" Margin="34,0,134,10" FontSize="12"/>  <Label Content="Дата" FontSize="12"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <ComboBox x:Name="MixingComboBox" Width="172" Height="32" Margin="34,0,24,10"/>  <DatePicker x:Name="DatePicker1" Width="169" Height="32" Margin="0,0,0,10"/>  </StackPanel>  <StackPanel>  <Label Content="Причина" Margin="34,0,0,5"/>  <TextBox x:Name="ReasonTextBox" Width="356" Height="32"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Button Width="107" Height="32" Style="{StaticResource ExitButton}" Content="Отмена" Margin="191,10,0,0" x:Name="CancelBtn" Click="CancelBtn\_Click" />  <Button Width="96" Height="32" Style="{StaticResource MainButton}" Content="Добавить" Margin="8,10,0,0" x:Name="AddButton" Click="AddButton\_Click"/>  </StackPanel>  </StackPanel>  </Grid>  </Window> |

Код C# AddRecordWindow.xaml.cs в таблице Б.18.

Таблица Б.18 – Листинг кода AddRecordWindow.xaml.cs.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Data.SqlClient;  using System.Windows;  namespace PersonnelDepartment.EmploymentRecord  {  /// <summary>  /// Логика взаимодействия для AddRecordWindow.xaml  /// </summary>  public partial class AddRecordWindow : Window  {  public event EventHandler RecordAdded;  private int \_employeeId;  public AddRecordWindow(int employeeId)  {  InitializeComponent();  \_employeeId = employeeId;  LoadMixingData();  }  public AddRecordWindow()  {  InitializeComponent();  LoadMixingData();  }  private void LoadMixingData()  {  List<string> mixingTitles = new List<string>();  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT TOP 4 Title FROM Mixing", connection);  using (var reader = command.ExecuteReader())  {  while (reader.Read())  {  mixingTitles.Add(reader.GetString(0));  }  }  }  MixingComboBox.ItemsSource = mixingTitles;  }  private void AddButton\_Click(object sender, RoutedEventArgs e)  {  string selectedMixing = MixingComboBox.SelectedItem as string;  DateTime? selectedDate = DatePicker1.SelectedDate;  string reason = ReasonTextBox.Text;  if (string.IsNullOrEmpty(selectedMixing) || selectedDate == null)  {  MessageBox.Show("Пожалуйста, заполните все поля.");  return;  }  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("INSERT INTO Entry\_in\_the\_work\_book (Id\_personal\_card, Date, Reason, Id\_mixing) VALUES (@Id\_personal\_card, @Date, @Reason, (SELECT ID FROM Mixing WHERE Title = @MixingTitle))", connection);  command.Parameters.AddWithValue("@Id\_personal\_card", \_employeeId);  command.Parameters.AddWithValue("@Date", selectedDate);  command.Parameters.AddWithValue("@Reason", reason);  command.Parameters.AddWithValue("@MixingTitle", selectedMixing);  command.ExecuteNonQuery();  }  MessageBox.Show("Запись добавлена успешно!");  // Вызываем событие RecordAdded, чтобы основное окно узнало об успешном добавлении  RecordAdded?.Invoke(this, EventArgs.Empty);  this.Close();  }  private void CancelBtn\_Click(object sender, RoutedEventArgs e)  {  this.Close();  }  }  } |

Код MainEmploymentRecord.xaml в таблице Б.19.

Таблица Б.19 – Листинг кода MainEmploymentRecord.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.EmploymentRecord.MainEmploymentRecord"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.EmploymentRecord"  mc:Ignorable="d"  WindowState="Maximized"  xmlns:conv="clr-namespace:PersonnelDepartment.Converter"  Title="Трудовые книжки" Height="720" Width="1280">    <Window.Resources>  <conv:ByteArrayToImageConverter x:Key="ByteArrayToImageConverter"/>  </Window.Resources>    <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" x:Name="MainWindow" Command="{Binding NavigateToMainWindowCommand}"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" x:Name="StaffBtn" Command="{Binding NavigateToStaffCommand}"/>  <Button Content="Трудовые книжки" Style="{StaticResource ButtonMain}"/>  <Button Content="Приказы" Style="{StaticResource ButtonMain}" x:Name="OrdersBtn" Command="{Binding NavigateToOrdersCommand}"/>  <Button Content="Отчеты" Style="{StaticResource ButtonMain}" x:Name="ReportsBtn" Command="{Binding NavigateToReportsCommand}"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label Content="{Binding UserName}" FontSize="14" Margin="0,0,32,0" />  <Label Content="{Binding UserRole}" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image Width="32" Height="32" Source="{Binding UserImage}"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" Command="{Binding ExitApplicationCommand}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>    <Grid>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0" Grid.ColumnSpan="7">  <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">  <TextBox x:Name="SearchTextBox" Width="320" Height="32" Style="{StaticResource CenteredTextBox}" Margin="0,52,16,0"/>  <Button Width="86" Height="32" Margin="0, 53, 0,0" Content="Найти" Command="{Binding FilterEmployeesCommand}" CommandParameter="{Binding Text, ElementName=SearchTextBox}" Style="{StaticResource MainButton}"/>  </StackPanel>    <StackPanel Orientation="Horizontal" >  <StackPanel Width="882" HorizontalAlignment="Left">  <Label Content="Трудовые книжки сотрудников" FontWeight="DemiBold" FontSize="18" Margin="40,0,0,15"/>  <ListView Width="842" Height="450" ScrollViewer.VerticalScrollBarVisibility="Auto" Style="{StaticResource ListViewSt}" ItemsSource="{Binding FilteredEmployees}" SelectedItem="{Binding SelectedEmployee}">  <ListView.View>  <GridView>  <GridViewColumn Header="Имя сотрудника" DisplayMemberBinding="{Binding FullName}" Width="200"/>  <GridViewColumn Header="Должность" DisplayMemberBinding="{Binding Position}" Width="150"/>  <GridViewColumn Header="Отдел" DisplayMemberBinding="{Binding Department}" Width="150"/>  <GridViewColumn Header="Телефон" DisplayMemberBinding="{Binding Phone}" Width="150"/>  <GridViewColumn Header="Дата рождения" DisplayMemberBinding="{Binding BirthDate, StringFormat=dd.MM.yyyy}" Width="150"/>  </GridView>  </ListView.View>  </ListView>  </StackPanel>  <StackPanel Margin="10,0,0,0">  <Label Content="Личные данные" FontWeight="DemiBold" Margin="0,47,0,0"/>  <StackPanel Orientation="Horizontal" Margin="0, 0, 0, 15">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Width="120">  <Image Width="80" Height="80" Source="{Binding SelectedEmployee.PhotoData, Converter={StaticResource ByteArrayToImageConverter}}" HorizontalAlignment="Left" Margin="10, 0, 0, 0"/>  </StackPanel>  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <!-- Имя -->  <Label Content="Имя" Grid.Row="0" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium" FontSize="12"/>  <Label Content="{Binding SelectedEmployee.FirstName}" Grid.Row="1" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <!-- Фамилия -->  <Label Content="Фамилия" Grid.Row="0" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium" FontSize="12"/>  <Label Content="{Binding SelectedEmployee.LastName}" Grid.Row="1" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  <!-- Отчество -->  <Label Content="Отчество" Grid.Row="2" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium" FontSize="12"/>  <Label Content="{Binding SelectedEmployee.Patronymic}" Grid.Row="3" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <!-- Дата рождения -->  <Label Content="Дата рождения" Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium" FontSize="12"/>  <Label Content="{Binding SelectedEmployee.BirthDate, StringFormat=dd.MM.yyyy}" Grid.Row="3" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  <!-- Должность -->  <Label Content="Должность" Grid.Row="4" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium" FontSize="12"/>  <Label Content="{Binding SelectedEmployee.Position}" Grid.Row="5" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <!-- Отдел -->  <Label Content="Отдел" Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium" FontSize="12"/>  <Label Content="{Binding SelectedEmployee.Department}" Grid.Row="5" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  </Grid>  </StackPanel>  <StackPanel>  <Label Content="Записи в трудовой книжке" FontWeight="DemiBold" FontSize="16" Margin="0,0,0,10"/>  <ListView Width="351" Height="200" ScrollViewer.VerticalScrollBarVisibility="Auto" Style="{StaticResource ListViewSt}" ItemsSource="{Binding EmploymentRecords}">  <ListView.View>  <GridView>  <GridViewColumn Header="Сведения" Width="100" DisplayMemberBinding="{Binding MixingTitle}"/>  <GridViewColumn Header="Дата" Width="100" DisplayMemberBinding="{Binding Date, StringFormat=dd.MM.yyyy}"/>  <GridViewColumn Header="Причина" DisplayMemberBinding="{Binding Reason}"/>  </GridView>  </ListView.View>  </ListView>  <Button Width="210" Height="32" Style="{StaticResource MainButton}" Command="{Binding AddRecordCommand}" Content="Добавить запись" Margin="0,10,0,0"/>  </StackPanel>  </StackPanel>  </StackPanel>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# MainEmploymentRecord.xaml.cs в таблице Б.20.

Таблица Б.20 – Листинг кода MainEmploymentRecord.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.ViewModel;  using System.Windows;  namespace PersonnelDepartment.EmploymentRecord  {  public partial class MainEmploymentRecord : Window  {  private Personal\_card \_currentUser;  public MainEmploymentRecord(Personal\_card currentUser)  {  InitializeComponent();  \_currentUser = currentUser;  var viewModel = new MainEmploymentRecordViewModel(\_currentUser, this);  DataContext = viewModel;  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код MainWindowAdmin.xaml в таблице Б.21.

Таблица Б.21 – Листинг кода MainWindowAdmin.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.MainWindowAdmin"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment"  mc:Ignorable="d"  WindowState="Maximized"  Title="Главное окно" Height="720" Width="1280">  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" x:Name="StaffBtn" Click="StaffBtn\_Click" />  <Button Content="База данных" Style="{StaticResource ButtonMain}" x:Name="DataSourceBtn" Click="DataSourceBtn\_Click"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="Namelbl" Content="Имя Фамилия" FontSize="14" Margin="0,0,32,0"/>  <Label x:Name="Rolelbl" Content="Должность" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image x:Name="UserImage" Width="32" Height="32"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>  <Grid>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <Label Content="Выберите подсистему" FontSize="18" VerticalAlignment="Center" HorizontalAlignment="Center" Margin="0,287"/>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# MainWindowAdmin.xaml.cs в таблице Б.22.

Таблица Б.22 – Листинг кода MainWindowAdmin.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.AdminDataSource;  using PersonnelDepartment.SraffAdmin;  using System.Windows;  namespace PersonnelDepartment  {  public partial class MainWindowAdmin : Window  {  private Personal\_card \_currentUser;  public MainWindowAdmin(Personal\_card currentUser)  {  InitializeComponent();  \_currentUser = currentUser;  DisplayUserInfo();  }  public MainWindowAdmin()  {  InitializeComponent();  }  private void DisplayUserInfo()  {  if (\_currentUser != null)  {  Namelbl.Content = $"{\_currentUser.Surname} {\_currentUser.Name}";  Rolelbl.Content = \_currentUser.Post.Title;  }  else  {  MessageBox.Show("Сотрудник не найден");  }  }  private void StaffBtn\_Click(object sender, RoutedEventArgs e)  {  StaffAdminForm staffAdminForm = new StaffAdminForm(\_currentUser);  staffAdminForm.Show();  Close();  }  private void DataSourceBtn\_Click(object sender, RoutedEventArgs e)  {  DataSourceForm dataSourceForm = new DataSourceForm(\_currentUser);  dataSourceForm.Show();  Close();  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код StaffAdminForm.xaml в таблице Б.23.

Таблица Б.23 – Листинг кода StaffAdminForm.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.SraffAdmin.StaffAdminForm"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.SraffAdmin"  mc:Ignorable="d"  WindowState="Maximized"  xmlns:conv="clr-namespace:PersonnelDepartment.Converter"  Title="Сотрудники" Height="720" Width="1280">  <Window.Resources>  <conv:ByteArrayToImageConverter x:Key="ByteArrayToImageConverter"/>  </Window.Resources>  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" x:Name="MainWindowBtn" Click="MainWindowBtn\_Click"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}"/>  <Button Content="База данных" Style="{StaticResource ButtonMain}" x:Name="EmploymentRecordBtn" Click="EmploymentRecordBtn\_Click"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="Namelbl" Content="Имя Фамилия" FontSize="14"/>  <Label x:Name="Rolelbl" Content="Должность" FontSize="12"/>  </StackPanel>  <Image Width="32" Height="32"/>  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>  <Grid>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0" Grid.ColumnSpan="7">  <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">  </StackPanel>  <StackPanel Orientation="Horizontal" >  <StackPanel>  <Label/>  <StackPanel Orientation="Horizontal" Margin="0, 0, 0, 15">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Width="174">  <Label Content="Аватар" FontSize="12" HorizontalAlignment="Left" Margin="0, 0, 0, 10"/>  <Image x:Name="AvatarImage" Width="80" Height="80" Source="../Media/template.png" HorizontalAlignment="Left" Margin="10, 0, 0, 0"/>  </StackPanel>  <!--Имя, Фамилия, Отчество-->  <StackPanel>  <StackPanel Orientation="Horizontal">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Имя" FontSize="12"/>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left">  <Label Content="Фамилия" FontSize="12"/>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Margin="0, 0, 19, 0">  <Label Content="Отчество" FontSize="12"/>  </StackPanel>  </StackPanel>  </StackPanel>  </StackPanel>  </StackPanel>  </StackPanel>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <StackPanel Orientation="Horizontal" VerticalAlignment="Center" HorizontalAlignment="Center">  <TextBox x:Name="SearchTextBox" Style="{StaticResource CenteredTextBox}" Width="320" Height="32" Margin="0,52,16,27"/>  <Button Width="86" Height="32" Margin="0, 53, 0,27" Content="Найти" x:Name="SearchBtn" Style="{StaticResource MainButton}" Click="SearchBtn\_Click" />  </StackPanel>  <StackPanel Orientation="Horizontal" >  <StackPanel Width="882" HorizontalAlignment="Left">  <Label Content="Личные карточки сотрудников" FontWeight="DemiBold" FontSize="18" Margin="40,0,0,15"/>  <ListView Width="842" Height="450" x:Name="EmployeeListView1" Style="{StaticResource ListViewSt}" Margin="40,0,0,0" SelectionChanged="EmployeeListView1\_SelectionChanged" ScrollViewer.VerticalScrollBarVisibility="Auto">  <ListView.View>  <GridView>  <GridViewColumn Header="Имя сотрудника" DisplayMemberBinding="{Binding FullName}" Width="200"/>  <GridViewColumn Header="Должность" DisplayMemberBinding="{Binding Position}" Width="150"/>  <GridViewColumn Header="Отдел" DisplayMemberBinding="{Binding Department}" Width="150"/>  <GridViewColumn Header="Телефон" DisplayMemberBinding="{Binding Phone}" Width="150"/>  <GridViewColumn Header="Дата рождения" DisplayMemberBinding="{Binding BirthDate, StringFormat=dd.MM.yyyy}" Width="150"/>  </GridView>  </ListView.View>  </ListView>  </StackPanel>  <StackPanel Margin="10,0,0,0">  <Label Content="Личные данные" FontWeight="DemiBold" Margin="0,47,0,0"/>  <StackPanel Orientation="Horizontal" Margin="0, 0, 0, 15">  <StackPanel Orientation="Vertical" HorizontalAlignment="Left" Width="120">  <Image Source="{Binding SelectedEmployee.PhotoData, Converter={StaticResource ByteArrayToImageConverter}}" Width="80" Height="80" HorizontalAlignment="Left" Margin="10, 0, 0, 0"/>  </StackPanel>  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*" />  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  <RowDefinition Height="Auto" />  </Grid.RowDefinitions>  <!--Имя, Фамилия, Отчество-->  <Label Content="Имя" FontSize="12" Grid.Row="0" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.FirstName}" Grid.Row="1" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Фамилия" FontSize="12" Grid.Row="0" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.LastName}" FontSize="12" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left"/>  <Label Content="Отчество" FontSize="12" Grid.Row="2" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.Patronymic}" Grid.Row="3" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Дата рождения" FontSize="12" Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.BirthDate, StringFormat=dd.MM.yyyy}" Grid.Row="3" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Должность" FontSize="12" Grid.Row="4" Grid.Column="0" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.Position}" Grid.Row="5" Grid.Column="0" FontSize="12" HorizontalAlignment="Left"/>  <Label Content="Отдел" FontSize="12" Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left" FontWeight="Medium"/>  <Label Content="{Binding SelectedEmployee.Department}" Grid.Row="5" Grid.Column="1" FontSize="12" HorizontalAlignment="Left"/>  </Grid>  </StackPanel>    <Label Content="Назначить учётные данные" FontSize="16" FontWeight="DemiBold"/>  <StackPanel Orientation="Horizontal">  <Label Content="Текущий логин сотрудника: "/>  <Label Content="{Binding SelectedEmployee.Login}" FontSize="12"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content="Текущий пароль сотрудника: "/>  <Label Content="{Binding SelectedEmployee.Password}" FontSize="12"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <Label Content="Логин" Margin="0,15,110,0"/>  <Label Content="Пароль" Margin="0,15,0,0"/>  </StackPanel>  <StackPanel Orientation="Horizontal">  <TextBox Width="140" Height="32" Style="{StaticResource CenteredTextBox}" Margin="0,0,15,0" x:Name="LoginTextBox" Text="{Binding SelectedEmployee.Login, Mode=TwoWay}"/>  <TextBox Width="174" Height="32" Style="{StaticResource CenteredTextBox}" x:Name="PasswordTextBox" Text="{Binding SelectedEmployee.Password, Mode=TwoWay}"/>  </StackPanel>  <Label Content="Назначить роль в системе" FontSize="16" FontWeight="DemiBold" />  <Label Content="Роль сотрудника"/>  <ComboBox x:Name="PositionComboBox" SelectedItem="{Binding SelectedEmployee.Position, Mode=TwoWay}" ItemsSource="{Binding Roles}" />  <Button Width="276" Height="32" Style="{StaticResource MainButton}" Content="Подтвердить изменения" VerticalAlignment="Bottom" Margin="0,20,0,0" x:Name="SaveChanges" Click="SaveChanges\_Click"/>  </StackPanel>  </StackPanel>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# StaffAdminForm.xaml.cs в таблице Б.24.

Таблица Б.24 – Листинг кода StaffAdminForm.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.AdminDataSource;  using PersonnelDepartment.ClassHelper;  using System.Collections.ObjectModel;  using System.ComponentModel;  using System.Data.SqlClient;  using System.Linq;  using System.Windows;  using System.Windows.Controls;  namespace PersonnelDepartment.SraffAdmin  {  /// <summary>  /// Логика взаимодействия для StaffAdminForm.xaml  /// </summary>  public partial class StaffAdminForm : Window, INotifyPropertyChanged  {  private Employee \_selectedEmployee;  // Реализация интерфейса INotifyPropertyChanged  public event PropertyChangedEventHandler PropertyChanged;  protected virtual void OnPropertyChanged(string propertyName)  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  private Personal\_card \_currentUser; // Хранение данных текущего пользователя  public ObservableCollection<Employee> Employees { get; set; }  public ObservableCollection<Employee> FilteredEmployees { get; set; }  public ObservableCollection<string> Roles { get; set; }  public Employee SelectedEmployee  {  get => \_selectedEmployee;  set  {  \_selectedEmployee = value;  OnPropertyChanged(nameof(SelectedEmployee)); // Обновляем привязки, когда меняется выбранный сотрудник  }  }  public StaffAdminForm(Personal\_card currentUser)  {  Roles = LoadRoles();  Employees = LoadEmployeesFromDatabase();  InitializeComponent();      FilteredEmployees = new ObservableCollection<Employee>(Employees); // Копия для фильтрации  EmployeeListView1.ItemsSource = FilteredEmployees;// Загружаем данные при инициализации  \_currentUser = currentUser;  DisplayUserInfo(); // Отображение информации о текущем пользователе  DataContext = this;  }  private void DisplayUserInfo()  {  if (\_currentUser != null)  {  Namelbl.Content = $"{\_currentUser.Surname} {\_currentUser.Name}";  Rolelbl.Content = \_currentUser.Post.Title;  }  else  {  MessageBox.Show("Сотрудник не найден");  }  }  private ObservableCollection<Employee> LoadEmployeesFromDatabase()  {  ObservableCollection<Employee> employees = new ObservableCollection<Employee>();  // Здесь ваш код для подключения к базе данных и выполнения SQL-запроса  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT p.ID, p.Name AS FirstName, p.Surname AS LastName, p.Patronymic, po.Title AS Position, d.Title AS Department, p.Telephone AS Phone, p.Date\_of\_birth AS BirthDate, p.Photo, p.Login, p.Password FROM Personal\_card p JOIN Post po ON p.Id\_post = po.ID JOIN Department d ON po.Id\_department = d.ID;", connection);  using (var reader = command.ExecuteReader())  {  while (reader.Read())  {  employees.Add(new Employee  {  ID = reader.GetInt32(0),  FirstName = reader.GetString(1),  LastName = reader.GetString(2),  Patronymic = reader.GetString(3),  Position = reader.GetString(4),  Department = reader.GetString(5),  Phone = reader.GetString(6),  BirthDate = reader.GetDateTime(7),  PhotoData = reader.IsDBNull(8) ? null : (byte[])reader[8],  Login = reader.IsDBNull(9) ? string.Empty : reader.GetString(9),  Password = reader.IsDBNull(10) ? string.Empty : reader.GetString(10)  });  }  }  }  return employees;  }  //private ObservableCollection<string> LoadPositions()  //{  // ObservableCollection<string> positions = new ObservableCollection<string>();  // using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  // {  // connection.Open();  // var command = new SqlCommand("SELECT TOP 2 Title FROM Post ORDER BY ID ASC;", connection);  // using (var reader = command.ExecuteReader())  // {  // while (reader.Read())  // {  // positions.Add(reader.GetString(0));  // }  // }  // }  // return positions;  //}  private void EmployeeListView1\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (EmployeeListView1.SelectedItem is Employee selectedEmployee)  {  SelectedEmployee = selectedEmployee; // Устанавливаем выбранного сотрудника для отображения его данных  }  }  private void FilterEmployees()  {  string searchText = SearchTextBox.Text.Trim().ToLower();  if (string.IsNullOrEmpty(searchText))  {  // Если поле поиска пустое, показываем всех сотрудников  FilteredEmployees.Clear();  foreach (var emp in Employees)  {  FilteredEmployees.Add(emp);  }  }  else  {  // Фильтруем список сотрудников по имени, фамилии или их комбинации  var filteredList = Employees.Where(emp =>  (!string.IsNullOrEmpty(emp.FirstName) && emp.FirstName.ToLower().Contains(searchText)) ||  (!string.IsNullOrEmpty(emp.LastName) && emp.LastName.ToLower().Contains(searchText)) ||  ($"{emp.FirstName} {emp.LastName}".ToLower().Contains(searchText))  ).ToList();  // Обновляем коллекцию для отображения  FilteredEmployees.Clear();  foreach (var emp in filteredList)  {  FilteredEmployees.Add(emp);  }  }  }  private void SaveChanges\_Click(object sender, RoutedEventArgs e)  {  if (SelectedEmployee == null) return;  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("UPDATE Personal\_card SET Login = @Login, Password = @Password, Id\_post = (SELECT ID FROM Post WHERE Title = @Position) WHERE ID = @ID", connection);  command.Parameters.AddWithValue("@Login", SelectedEmployee.Login);  command.Parameters.AddWithValue("@Password", SelectedEmployee.Password);  command.Parameters.AddWithValue("@Position", SelectedEmployee.Position);  command.Parameters.AddWithValue("@ID", SelectedEmployee.ID);  command.ExecuteNonQuery();  }  MessageBox.Show("Изменения сохранены");  }  private ObservableCollection<string> LoadRoles()  {  ObservableCollection<string> roles = new ObservableCollection<string>();  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT Title FROM Post ORDER BY ID ASC;", connection);  using (var reader = command.ExecuteReader())  {  while (reader.Read())  {  roles.Add(reader.GetString(0));  }  }  }  return roles;  }  private void SearchBtn\_Click(object sender, RoutedEventArgs e)  {  FilterEmployees();  }  private void EmploymentRecordBtn\_Click(object sender, RoutedEventArgs e)  {  DataSourceForm dataSourceForm = new DataSourceForm(\_currentUser);  dataSourceForm.Show();  Close();  }  private void MainWindowBtn\_Click(object sender, RoutedEventArgs e)  {  MainWindowAdmin mainWindowAdmin = new MainWindowAdmin(\_currentUser);  mainWindowAdmin.Show();  Close();  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код DataSourceForm.xaml в таблице Б.25.

Таблица Б.25 – Листинг кода DataSourceForm.xaml.

|  |
| --- |
| <Window x:Class="PersonnelDepartment.AdminDataSource.DataSourceForm"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:PersonnelDepartment.AdminDataSource"  mc:Ignorable="d"  WindowState="Maximized"  Title="DataSourceForm" Height="720" Width="1280">  <Grid>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="62"/>  <RowDefinition Height="\*"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="Auto"/>  </Grid.ColumnDefinitions>  <StackPanel Grid.ColumnSpan="2" Background="#E4EDF3"/>  <StackPanel Orientation="Horizontal" HorizontalAlignment="Left" Margin="20 0 0 0">  <Button FontSize="20" VerticalAlignment="Center" Content="Отдел кадров" Style="{StaticResource ButtonMain}" Click="MainWindowBtn\_Click" x:Name="MainWindowBtn"/>  <Button Content="Сотрудники" Style="{StaticResource ButtonMain}" x:Name="StaffAdminBtn" Click="StaffAdminBtn\_Click"/>  <Button Content="База данных" Style="{StaticResource ButtonMain}" x:Name="EmploymentRecordBtn"/>  </StackPanel>  <StackPanel Orientation="Horizontal" MinWidth="130" Grid.Column="1" Margin="0 0 20 0">  <StackPanel Orientation="Vertical" VerticalAlignment="Center">  <Label x:Name="Namelbl" Content="Имя Фамилия" FontSize="14" Margin="0,0,32,0"/>  <Label x:Name="Rolelbl" Content="Должность" FontSize="12" Margin="0,0,32,0"/>  </StackPanel>  <!--<Image x:Name="UserPhoto" Width="32" Height="32"/>-->  <Button Content="Выйти" Width="66" Height="32" Style="{StaticResource ExitButton}" x:Name="ExitBtn" Click="ExitBtn\_Click" Grid.Column="1"/>  </StackPanel>  </Grid>  <Grid>  <StackPanel Orientation="Vertical" Background="#FFFF" Height="642" Width="1280" Margin="0,62,0,0">  <Label Content="Добавить новые данные" FontSize="18" FontWeight="DemiBold" Margin="450,29,0,14" />  <Label Content="Выбрать категорию" FontSize="16" Margin="450,0,0,14" />  <ComboBox Width="367" Height="32" x:Name="ComboBox" SelectionChanged="ComboBox\_SelectionChanged" />  <Label Content="Добавить новую запись" FontSize="16" Margin="450,14,0,14"/>  <StackPanel Orientation="Horizontal">  <TextBox Width="367" Height="32" Style="{StaticResource CenteredTextBox}" Margin="450,0,10,14" x:Name="NewDataInput"/>  <Button Width="100" Height="32" Margin="0,0,0,14" Style="{StaticResource MainButton}" Content="Добавить" x:Name="AddButton" Click="AddButton\_Click"/>  </StackPanel>  <Label Content="Записи в базе данных" FontSize="16" Margin="450,14,0,14"/>  <ListView Width="664" x:Name="ListView" Height="280" Style="{StaticResource ListViewSt}" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,0,0,14" ScrollViewer.VerticalScrollBarVisibility="Auto">  <ListView.View>  <GridView x:Name="GridView">  <!-- Заголовки столбцов будут добавляться динамически -->  </GridView>  </ListView.View>  <ListView.ItemTemplate>  <DataTemplate>  <StackPanel Orientation="Horizontal" Margin="5">  <TextBlock Text="{Binding}" VerticalAlignment="Center" Margin="0,0,10,0"/>  <Button Content="Удалить" Width="75" Height="25" Style="{StaticResource ExitButton}" x:Name="DeleteRow" Click="DeleteRow\_Click"/>  </StackPanel>  </DataTemplate>  </ListView.ItemTemplate>  </ListView>  </StackPanel>  </Grid>  </Grid>  </Window> |

Код C# DataSourceForm.xaml.cs в таблице Б.26.

Таблица Б.26 – Листинг кода DataSourceForm.xaml.cs.

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.SraffAdmin;  using System;  using System.Collections.Generic;  using System.Data; // Для работы с базой данных  using System.Data.SqlClient; // Подключение к SQL Server  using System.Linq;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.AdminDataSource  {  /// <summary>  /// Логика взаимодействия для DataSourceForm.xaml  /// </summary>  public partial class DataSourceForm : Window  {    private Employee currentUser;  private Personal\_card \_currentUser; // Хранение данных текущего пользователя  // Список таблиц с переводами  private Dictionary<string, string> tableNames = new Dictionary<string, string>()  {  { "Citizenship", "Гражданство" },  { "Department", "Отдел" },  { "Post", "Должность" },  { "Education", "Образование" },  { "Mixing", "Причины" },  { "Salary\_type", "Тип зарплаты" }  };  private readonly Dictionary<string, string> columnNameMap = new Dictionary<string, string>  {  //Гражданство  { "ID", "Номер" },  { "Country", "Страна" },  //Отдел  { "Title", "Наименование" },  { "Responsibilities", "Описание" },  { "Base\_salary", "Базовая зарплата" },  { "Id\_department", "Номер отдела" },  { "Levell", "Уровень" },  };  private string connectionString = "data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework";  public DataSourceForm(Personal\_card currentUser)  {  InitializeComponent();  \_currentUser = currentUser;  DisplayUserInfo(); // Отображение информации о текущем пользователе    // Инициализация комбобокса  foreach (var table in tableNames.Values)  {  ComboBox.Items.Add(table);  }  ComboBox.SelectionChanged += ComboBox\_SelectionChanged;;  }  private void DisplayUserInfo()  {  if (\_currentUser != null)  {  Namelbl.Content = $"{\_currentUser.Surname} {\_currentUser.Name}";  Rolelbl.Content = \_currentUser.Post.Title;  }  }  private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)  {  if (ComboBox.SelectedItem != null)  {  string selectedTableRus = ComboBox.SelectedItem.ToString();  string selectedTableEng = tableNames.FirstOrDefault(t => t.Value == selectedTableRus).Key;  LoadTableData(selectedTableEng);  }  }  // Загрузка данных в ListView  private void LoadTableData(string tableName)  {  try  {  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  string query = $"SELECT \* FROM {tableName}";  SqlDataAdapter adapter = new SqlDataAdapter(query, connection);  DataTable dataTable = new DataTable();  adapter.Fill(dataTable);  // Очистка предыдущих данных  ListView.Items.Clear();  GridView.Columns.Clear();  // Динамическое создание столбцов из DataTable  foreach (DataColumn column in dataTable.Columns)  {  // Формируем ключ для перевода столбца в формате "TableName\_ColumnName"  string columnKey = column.ColumnName;  // Получаем переведённое название столбца из словаря, если оно существует  string columnHeader = columnNameMap.ContainsKey(columnKey) ? columnNameMap[columnKey] : column.ColumnName;  // Добавляем столбец в GridView  GridView.Columns.Add(new GridViewColumn  {  Header = columnHeader, // Переведённый или оригинальный заголовок  DisplayMemberBinding = new System.Windows.Data.Binding($"[{column.ColumnName}]") // Привязка к данным  });  }  // Добавляем колонку для кнопки "Удалить"  GridView.Columns.Add(new GridViewColumn  {  Header = "Действия",  CellTemplate = CreateDeleteButtonTemplate()  });  // Добавляем строки в ListView  foreach (DataRow row in dataTable.Rows)  {  var item = new Dictionary<string, object>();  foreach (DataColumn column in dataTable.Columns)  {  item[column.ColumnName] = row[column];  }  item["ID"] = row["ID"]; // Сохраняем ID для удаления  ListView.Items.Add(item);  }  }  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка загрузки данных: {ex.Message}");  }  }  // Шаблон кнопки "Удалить"  private DataTemplate CreateDeleteButtonTemplate()  {  var template = new DataTemplate();  // Стек для размещения кнопки  var stackPanel = new FrameworkElementFactory(typeof(StackPanel));  stackPanel.SetValue(StackPanel.OrientationProperty, Orientation.Horizontal);  // Создание кнопки "Удалить"  var button = new FrameworkElementFactory(typeof(Button));  button.SetValue(Button.ContentProperty, "Удалить");  button.SetValue(Button.WidthProperty, 75.0);  button.SetValue(Button.HeightProperty, 25.0);  button.SetValue(Button.MarginProperty, new Thickness(5));  // Назначаем стиль кнопке  button.SetValue(Button.StyleProperty, Application.Current.Resources["ExitButton"]);  // Привязываем обработчик события  button.AddHandler(Button.ClickEvent, new RoutedEventHandler(DeleteRow\_Click));  // Добавляем кнопку в стек  stackPanel.AppendChild(button);  // Устанавливаем стек как шаблон  template.VisualTree = stackPanel;  return template;  }  private void DeleteRow\_Click(object sender, RoutedEventArgs e)  {  // Получаем ID строки из контекста кнопки  if (sender is Button button)  {  var row = button.DataContext as Dictionary<string, object>;  if (row != null && row.ContainsKey("ID"))  {  int id = Convert.ToInt32(row["ID"]);  string tableName = tableNames.FirstOrDefault(t => t.Value == ComboBox.SelectedItem.ToString()).Key;  try  {  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  string query = $"DELETE FROM {tableName} WHERE ID = @ID";  SqlCommand command = new SqlCommand(query, connection);  command.Parameters.AddWithValue("@ID", id);  command.ExecuteNonQuery();  MessageBox.Show("Запись успешно удалена.");  LoadTableData(tableName); // Обновляем данные  }  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при удалении записи: {ex.Message}");  }  }  }  }  private void AddNewRow(string tableName, string[] inputData)  {  try  {  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Генерация SQL-запроса для вставки данных  string query = GenerateInsertQuery(tableName, inputData.Length);  using (SqlCommand command = new SqlCommand(query, connection))  {  // Подстановка значений параметров  for (int i = 0; i < inputData.Length; i++)  {  command.Parameters.AddWithValue($"@param{i + 1}", inputData[i].Trim());  }  command.ExecuteNonQuery();  MessageBox.Show("Данные успешно добавлены!");  // Обновление отображаемых данных  LoadTableData(tableName);  }  }  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при добавлении данных: {ex.Message}");  }  }  private string GenerateInsertQuery(string tableName, int paramCount)  {  using (SqlConnection connection = new SqlConnection(connectionString))  {  connection.Open();  // Получение имен столбцов из таблицы  string columnQuery = $"SELECT COLUMN\_NAME FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = '{tableName}'";  SqlDataAdapter adapter = new SqlDataAdapter(columnQuery, connection);  DataTable columns = new DataTable();  adapter.Fill(columns);  // Исключаем столбец ID, если он автоинкрементный  var columnNames = columns.AsEnumerable()  .Select(row => row.Field<string>("COLUMN\_NAME"))  .Skip(1) // Пропускаем первый столбец (ID)  .Take(paramCount)  .ToArray();  if (columnNames.Length != paramCount)  {  throw new Exception("Количество введенных данных не совпадает с количеством столбцов.");  }  string columnList = string.Join(", ", columnNames);  string valueList = string.Join(", ", Enumerable.Range(1, paramCount).Select(i => $"@param{i}"));  return $"INSERT INTO {tableName} ({columnList}) VALUES ({valueList})";  }  }  private void AddButton\_Click(object sender, RoutedEventArgs e)  {  // Проверяем, выбрана ли таблица  if (ComboBox.SelectedItem == null)  {  MessageBox.Show("Выберите таблицу для добавления данных.");  return;  }  // Проверяем, введены ли данные  if (string.IsNullOrWhiteSpace(NewDataInput.Text))  {  MessageBox.Show("Введите данные для добавления.");  return;  }  // Получаем выбранное имя таблицы  string selectedTableRus = ComboBox.SelectedItem.ToString();  string selectedTableEng = tableNames.FirstOrDefault(t => t.Value == selectedTableRus).Key;  // Парсим данные из TextBox  string[] inputData = NewDataInput.Text.Split(',');  // Вызываем метод добавления  AddNewRow(selectedTableEng, inputData);  // Очищаем поле ввода  NewDataInput.Clear();  }  private void StaffAdminBtn\_Click(object sender, RoutedEventArgs e)  {  StaffAdminForm staffAdminForm = new StaffAdminForm(\_currentUser);  staffAdminForm.Show();  Close();  }  private void MainWindowBtn\_Click(object sender, RoutedEventArgs e)  {  MainWindowAdmin mainWindowAdmin = new MainWindowAdmin(\_currentUser);  mainWindowAdmin.Show();  Close();  }  private void ExitBtn\_Click(object sender, RoutedEventArgs e)  {  // Показать окно подтверждения  MessageBoxResult result = MessageBox.Show(  "Вы действительно хотите выйти?",  "Подтверждение выхода",  MessageBoxButton.YesNo,  MessageBoxImage.Question  );  // Проверка ответа пользователя  if (result == MessageBoxResult.Yes)  {  Application.Current.Shutdown(); // Закрытие приложения  }  // Если пользователь нажал "Нет", ничего не делаем  }  }  } |

Код C# Employee.cs в таблице Б.27.

Таблица Б.27 – Листинг кода Employee.cs.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace PersonnelDepartment.ClassHelper  {  public class Employee  {  public int ID { get; set; }  public string FirstName { get; set; } // Name из Personal\_card  public string LastName { get; set; } // Surname из Personal\_card  public string Patronymic { get; set; }  public string Position { get; set; } // Title из Post  public string Department { get; set; } // Title из Department  public string Phone { get; set; } // Telephone из Personal\_card  public DateTime BirthDate { get; set; } // Date\_of\_birth из Personal\_card  public byte[] PhotoData { get; set; } // Путь к изображению    // Свойство для отображения полного имени  public string FullName => $"{FirstName} {LastName}";  // Новые свойства для логина и пароля  public string Login { get; set; }  public string Password { get; set; }  }  } |

Код C# EntryWorkBook.cs в таблице Б.28.

Таблица Б.28 – Листинг кода EntryWorkBook.cs.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace PersonnelDepartment.ClassHelper  {  public class EntryInWorkBook  {    public int ID { get; set; }  public DateTime Date { get; set; }  public string Reason { get; set; }  public string MixingTitle { get; set; } // Название из таблицы Mixing  }  } |

Код C# PersonalCard.cs в таблице Б.29.

Таблица Б.29 – Листинг кода PersonalCard.cs.

|  |
| --- |
| using System;  namespace PersonnelDepartment.ClassHelper  {  public class PersonalCard  {  public int Id { get; set; }  public string Surname { get; set; }  public string Name { get; set; }  public string Patronymic { get; set; }  public DateTime DateOfBirth { get; set; }  public string SeriesAndNumber { get; set; }  public string IssuedByWhom { get; set; }  public DateTime DateOfIssue { get; set; }  public string RegistrationAddress { get; set; }  public string Telephone { get; set; }  public bool Children { get; set; }  public bool MilitaryService { get; set; }  public int IdCitizenship { get; set; }  public int IdPost { get; set; }  public int IdDepartment { get; set; }  public int IdEducation { get; set; }  public string Birthplace { get; set; }  public string Email { get; set; }  public string EducationInstitution { get; set; }  public byte[] Photo { get; set; }  public string GetSeries  {  get => !string.IsNullOrEmpty(SeriesAndNumber) && SeriesAndNumber.Length >= 4  ? SeriesAndNumber.Substring(0, 4)  : string.Empty;  set =>  // Устанавливаем новую серию, объединяя её с номером, если он уже есть  SeriesAndNumber = value + (SeriesAndNumber?.Substring(4) ?? "");  }  public string GetNumber  {  get => !string.IsNullOrEmpty(SeriesAndNumber) && SeriesAndNumber.Length > 4  ? SeriesAndNumber.Substring(4)  : string.Empty;  set =>  // Устанавливаем новый номер, объединяя его с серией, если она уже есть  SeriesAndNumber = (SeriesAndNumber?.Substring(0, 4) ?? "") + value;  }  public void SetSeriesAndNumber(string series, string number)  {  SeriesAndNumber = series + number;  }  }  } |

Код C# UserService.cs в таблице Б.30.

Таблица Б.30 – Листинг кода UserService.cs.

|  |
| --- |
| using System.Data.SqlClient;  namespace PersonnelDepartment.ClassHelper  {  public class UserService  {  private string \_connectionString = "data source=DESKTOP-F4MJF3P;initial catalog=TestDB;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework";  public UserService(string connectionString)  {  this.\_connectionString = connectionString;  }  // Метод для авторизации пользователя по логину и паролю  public UserInfo Authenticate(string login, string password)  {  string query = @"  SELECT p.FullName, ps.Position  FROM personal\_card p  JOIN post ps ON p.PostId = ps.Id  WHERE p.Login = @Login AND p.Password = @Password";  using (SqlConnection connection = new SqlConnection(\_connectionString))  {  connection.Open();  using (SqlCommand command = new SqlCommand(query, connection))  {  command.Parameters.AddWithValue("@Login", login);  command.Parameters.AddWithValue("@Password", password);  SqlDataReader reader = command.ExecuteReader();  if (reader.Read())  {  return new UserInfo  {  FullName = reader["FullName"].ToString(),  Position = reader["Position"].ToString()  };  }  else  {  return null; // Если пользователь не найден  }  }  }  }  }  // Класс для хранения информации о пользователе  public class UserInfo  {  public string FullName { get; set; }  public string Position { get; set; }  }  } |

Код C# Validator.cs в таблице Б.31.

Таблица Б.31 – Листинг кода Validator.cs.

|  |
| --- |
| using System.Text.RegularExpressions;  using System;  using System.Windows.Controls;  namespace PersonnelDepartment.ClassHelper  {  public class Validator  {  // Регулярные выражения для проверки полей  private static readonly string NamePattern = "^[А-ЯЁ][а-яё]+$"; // Имя, Фамилия, Отчество  //private static readonly string BirthDatePattern = "^(0[1-9]|[12][0-9]|3[01])\\.(0[1-9]|1[0-2])\\.(19|20)\\d{2}$"; // Дата рождения, Дата выдачи  private static readonly string PlacePattern = "^[А-ЯЁа-яё\\s.,]+$"; //Кем выдан  private static readonly string PlaceBirthPattern = "^[гпсд]\\. ?[А-ЯЁ][а-яё\\s.,]+$|^[А-ЯЁ][а-яё\\s.,]+$"; //Место рождения  private static readonly string PlaceAddressPattern = "^(?=.\*[а-яА-ЯЁё])[А-ЯЁа-яё0-9\\s.,]+$"; //Адрес регистрации  private static readonly string PhonePattern = "^\\+?[0-9]{10,15}$"; // Телефон  private static readonly string EmailPattern = "^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$"; // Электронная почта  private static readonly string NonEmptyPattern = "^(?!\\s\*$).+"; // Отдел, Должность, Уровень образования, Гражданство  private static readonly string PassportSeriesPattern = "^\\d{4}$"; // Серия паспорта  private static readonly string PassportNumberPattern = "^\\d{6}$"; // Номер паспорта  private static readonly string EducationInstitutionPattern = "^[А-ЯЁ][А-ЯЁа-яё0-9 №]+(?: [А-ЯЁа-яё0-9 №]+)\*$"; // Учебное заведение  private static readonly string datePattern = "^(0[1-9]|[12][0-9]|3[01])\\.(0[1-9]|1[0-2])\\.(19|20)\\d{2}$";  // Валидация имени (Фамилия, Имя, Отчество)  public static bool IsValidName(string name, out string errorMessage)  {  return IsValid(name, NamePattern, out errorMessage, "Некорректный ввод в поле «Имя»");  }  public static bool IsValidSurName(string name, out string errorMessage)  {  return IsValid(name, NamePattern, out errorMessage, "Некорректный ввод в поле «Фамилия»");  }  public static bool IsValidPatronymic(string name, out string errorMessage)  {  return IsValid(name, NamePattern, out errorMessage, "Некорректный ввод в поле «Отчество»");  }  public static bool IsValidBirthDate(string birthDateString, out string errorMessage)  {  // Проверка строки на соответствие формату даты  if (string.IsNullOrWhiteSpace(birthDateString) || !Regex.IsMatch(birthDateString, datePattern))  {  errorMessage = "Некорректный ввод в поле «Дата рождения»";  return false;  }  // Попытка преобразовать строку в DateTime  DateTime birthDate;  if (DateTime.TryParseExact(birthDateString, "dd.MM.yyyy", null, System.Globalization.DateTimeStyles.None, out birthDate))  {  // Проверка на дату больше текущей  if (birthDate > DateTime.Now.Date)  {  errorMessage = "Некорректный ввод в поле «Дата рождения»";  return false;  }  errorMessage = null;  return true;  }  errorMessage = "Некорректный ввод в поле «Дата рождения»";  return false;  }  // Валидация места рождения  public static bool IsValidPlace(string place, out string errorMessage)  {  return IsValid(place, PlaceBirthPattern, out errorMessage, "Некорректный ввод в поле «Место рождения»");  }  // Валидация адреса регистрации  public static bool IsValidAddress(string address, out string errorMessage)  {  return IsValid(address, PlaceAddressPattern, out errorMessage, "Некорректный ввод в поле «Адрес регистрации»");  }  // Валидация телефона  public static bool IsValidPhone(string phone, out string errorMessage)  {  return IsValid(phone, PhonePattern, out errorMessage, "Некорректный ввод в поле «Телефонный номер»");  }  // Валидация электронной почты  public static bool IsValidEmail(string email, out string errorMessage)  {  return IsValid(email, EmailPattern, out errorMessage, "Некорректный ввод в поле «Электронная почта»");  }  // Валидация отдела  public static bool IsValidDepartment(string department, out string errorMessage)  {  return IsValid(department, NonEmptyPattern, out errorMessage, "Поле со списком «Отдел» пустое");  }  // Валидация должности  public static bool IsValidPost(string post, out string errorMessage)  {  return IsValid(post, NonEmptyPattern, out errorMessage, "Поле со списком «Должность» пустое");  }  // Валидация серии паспорта  public static bool IsValidPassportSeries(string series, out string errorMessage)  {  return IsValid(series, PassportSeriesPattern, out errorMessage, "Некорректный ввод в поле «Серия паспорта»");  }  // Валидация номера паспорта  public static bool IsValidPassportNumber(string number, out string errorMessage)  {  return IsValid(number, PassportNumberPattern, out errorMessage, "Некорректный ввод в поле «Номер паспорта»");  }  // Валидация даты выдачи паспорта  public static bool IsValidDateOfIssue(string dateOfIssueString, out string errorMessage)  {  // Проверка строки на соответствие формату даты  if (string.IsNullOrWhiteSpace(dateOfIssueString) || !Regex.IsMatch(dateOfIssueString, datePattern))  {  errorMessage = "Некорректный ввод в поле «Дата выдачи»";  return false;  }  // Попытка преобразовать строку в DateTime  DateTime dateOfIssue;  if (DateTime.TryParseExact(dateOfIssueString, "dd.MM.yyyy", null, System.Globalization.DateTimeStyles.None, out dateOfIssue))  {  // Проверка на дату больше текущей  if (dateOfIssue > DateTime.Now.Date)  {  errorMessage = "Некорректный ввод в поле «Дата выдачи»";  return false;  }  errorMessage = null;  return true;  }  errorMessage = "Некорректный ввод в поле «Дата выдачи»";  return false;  }  // Валидация уровня образования  public static bool IsValidEducationLevel(string educationLevel, out string errorMessage)  {  return IsValid(educationLevel, NonEmptyPattern, out errorMessage, "Поле со списком «Уровень» пустое");  }  // Валидация поля «Кем выдан»  public static bool IsValidIssuedBy(string issuedBy, out string errorMessage)  {  return IsValid(issuedBy, PlacePattern, out errorMessage, "Некорректный ввод в поле «Кем выдан»");  }  //Валидация поля «Учебное заведение»  public static bool IsValidEducationInstitution(string institution, out string errorMessage)  {  return IsValid(institution, EducationInstitutionPattern, out errorMessage, "Некорректный ввод в поле «Учебное заведение»");  }  // Валидация гражданства  public static bool IsValidCitizenship(string citizenship, out string errorMessage)  {  return IsValid(citizenship, NonEmptyPattern, out errorMessage, "Поле со списком «Гражданство» пустое");  }  // Валидация поля с фото  public static bool IsValidPhoto(string photoPath, out string errorMessage)  {  // Проверяем, что поле не пустое и путь к файлу корректный  if (string.IsNullOrWhiteSpace(photoPath))  {  errorMessage = "Выберите значение в поле «Аватар»";  return false;  }  // Проверяем, что файл существует  if (!System.IO.File.Exists(photoPath))  {  errorMessage = "Выберите значение в поле «Аватар»";  return false;  }  errorMessage = null;  return true;  }  // Валидация поля с обязательным заполнением  public static bool IsValidField(string field, out string errorMessage, string errorText)  {  if (string.IsNullOrWhiteSpace(field))  {  errorMessage = errorText;  return false;  }  errorMessage = null;  return true;  }  // Основной метод для проверки строк с регулярными выражениями  private static bool IsValid(string value, string pattern, out string errorMessage, string errorText)  {  if (string.IsNullOrWhiteSpace(value))  {  errorMessage = errorText;  return false;  }  if (!Regex.IsMatch(value, pattern))  {  errorMessage = errorText;  return false;  }  errorMessage = null;  return true;  }  }  } |

Код C# RelayCommand.cs в таблице Б.32.

Таблица Б.32 – Листинг кода RelayCommand.cs.

|  |
| --- |
| using System;  using System.Windows.Input;  namespace PersonnelDepartment.Commands  {  public class RelayCommand : ICommand  {  private readonly Action \_execute;  private readonly Func<bool> \_canExecute;  public RelayCommand(Action execute, Func<bool> canExecute = null)  {  \_execute = execute ?? throw new ArgumentNullException(nameof(execute));  \_canExecute = canExecute;  }  public bool CanExecute(object parameter) => \_canExecute == null || \_canExecute();  public void Execute(object parameter) => \_execute();  public event EventHandler CanExecuteChanged  {  add => CommandManager.RequerySuggested += value;  remove => CommandManager.RequerySuggested -= value;  }  }  public class RelayCommand<T> : ICommand  {  private readonly Action<T> \_execute;  private readonly Func<T, bool> \_canExecute;  public RelayCommand(Action<T> execute) : this(execute, null) { }  public RelayCommand(Action<T> execute, Func<T, bool> canExecute)  {  \_execute = execute ?? throw new ArgumentNullException(nameof(execute));  \_canExecute = canExecute;  }  public bool CanExecute(object parameter) => \_canExecute == null || \_canExecute((T)parameter);  public void Execute(object parameter) => \_execute((T)parameter);  public event EventHandler CanExecuteChanged  {  add => CommandManager.RequerySuggested += value;  remove => CommandManager.RequerySuggested -= value;  }  }  } |

Код C# AppConnect.cs в таблице Б.33.

Таблица Б.33 – Листинг кода AppConnect.cs.

|  |
| --- |
| namespace PersonnelDepartment.ControlClasses  {  class AppConnect  {  public static HumanResourcesDepartmentEntities dbModel = new HumanResourcesDepartmentEntities();  }  } |

Код C# AuthorizationViewModel в таблице Б.34.

Таблица Б.34 – Листинг кода AuthorizationViewModel.

|  |
| --- |
| using PersonnelDepartment.Commands;  using PersonnelDepartment.ControlClasses;  using System.Linq;  using System.Windows;  using System.Windows.Input;  namespace PersonnelDepartment.ViewModel  {  public class AuthorizationViewModel : BaseViewModel  {  private string \_login;  private string \_password;  private ICommand \_loginCommand;  public string Login  {  get => \_login;  set  {  \_login = value;  OnPropertyChanged(nameof(Login));  }  }  public string Password  {  get => \_password;  set  {  \_password = value;  OnPropertyChanged(nameof(Password));  }  }  public ICommand LoginCommand => \_loginCommand ??= new RelayCommand(ExecuteLogin);  private void ExecuteLogin()  {  var user = AppConnect.dbModel.Personal\_card  .FirstOrDefault(p => p.Login == Login && p.Password == Password);  if (user == null)  {  MessageBox.Show("Пользователь не найден!");  return;  }  if (user.Post.ID == 1)  {  var mainWindow = new MainWindowAdmin(user);  mainWindow.Show();  Application.Current.MainWindow.Close();  }  else if (user.Post.ID == 2)  {  var mainWindow = new MainWindowInspector(user);  mainWindow.Show();  Application.Current.MainWindow.Close();  }  }  }  } |

Код C# BaseViewModel.cs в таблице Б.35.

Таблица Б.35 – Листинг кода BaseViewModel.cs.

|  |
| --- |
| using System.ComponentModel;  namespace PersonnelDepartment.ViewModel  {  public class BaseViewModel : INotifyPropertyChanged  {  public event PropertyChangedEventHandler PropertyChanged;  protected void OnPropertyChanged(string propertyName)  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));  }  }  } |

Код C# MainEmploymentRecordViewModel.cs в таблице Б.36.

Таблица Б.36 – Листинг кода MainEmploymentRecordViewModel.cs.

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.Commands;  using PersonnelDepartment.EmploymentRecord;  using System.Collections.ObjectModel;  using System.Data.SqlClient;  using System.Linq;  using System.Windows.Input;  using System.Windows;  using System.Windows.Media.Imaging;  using PersonnelDepartment.Staff;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  namespace PersonnelDepartment.ViewModel  {  public class MainEmploymentRecordViewModel : BaseViewModel  {  private Personal\_card \_currentUser;  private Window \_currentWindow;  private string \_userName;  private string \_userRole;  private BitmapImage \_userImage;  private Employee \_selectedEmployee;  private ObservableCollection<Employee> \_employees;  private ObservableCollection<Employee> \_filteredEmployees;  private ObservableCollection<EntryInWorkBook> \_employmentRecords;  public ObservableCollection<Employee> Employees  {  get => \_employees;  set  {  \_employees = value;  OnPropertyChanged(nameof(Employees));  }  }  public ObservableCollection<Employee> FilteredEmployees  {  get => \_filteredEmployees;  set  {  \_filteredEmployees = value;  OnPropertyChanged(nameof(FilteredEmployees));  }  }  public ObservableCollection<EntryInWorkBook> EmploymentRecords  {  get => \_employmentRecords;  set  {  \_employmentRecords = value;  OnPropertyChanged(nameof(EmploymentRecords));  }  }  public Employee SelectedEmployee  {  get => \_selectedEmployee;  set  {  \_selectedEmployee = value;  OnPropertyChanged(nameof(SelectedEmployee));  LoadEmploymentRecordsForSelectedEmployee();  }  }  public ICommand FilterEmployeesCommand { get; set; }  public ICommand AddRecordCommand { get; set; }  public ICommand NavigateToMainWindowCommand { get; set; }  public ICommand NavigateToOrdersCommand { get; set; }  public ICommand NavigateToReportsCommand { get; set; }  public ICommand NavigateToStaffCommand { get; set; }  // public ICommand ExitApplicationCommand { get; set; }  public MainEmploymentRecordViewModel(Personal\_card currentUser, Window currentWindow)  {  \_currentUser = currentUser;  \_currentWindow = currentWindow;  UserName = $"{currentUser.Surname} {currentUser.Name}";  UserRole = currentUser.Post.Title;  if (currentUser.Photo != null)  {  UserImage = ByteArrayToBitmapImage(currentUser.Photo);  }  Employees = LoadEmployeesFromDatabase();  FilteredEmployees = new ObservableCollection<Employee>(Employees);  EmploymentRecords = new ObservableCollection<EntryInWorkBook>();  FilterEmployeesCommand = new RelayCommand<string>(FilterEmployees);  AddRecordCommand = new RelayCommand(AddRecord);  NavigateToMainWindowCommand = new RelayCommand(NavigateToMainWindow);  NavigateToOrdersCommand = new RelayCommand(NavigateToOrders);  NavigateToReportsCommand = new RelayCommand(NavigateToReports);  NavigateToStaffCommand = new RelayCommand(NavigateToStaff);  // ExitApplicationCommand = new RelayCommand(ExitApplication);    }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  public void FilterEmployees(string searchText)  {  var filteredList = Employees.Where(emp =>  (!string.IsNullOrEmpty(emp.FirstName) && emp.FirstName.ToLower().Contains(searchText)) ||  (!string.IsNullOrEmpty(emp.LastName) && emp.LastName.ToLower().Contains(searchText)) ||  ($"{emp.FirstName} {emp.LastName}".ToLower().Contains(searchText))  ).ToList();  FilteredEmployees.Clear();  foreach (var emp in filteredList)  {  FilteredEmployees.Add(emp);  }  }  private ObservableCollection<Employee> LoadEmployeesFromDatabase()  {  var employees = new ObservableCollection<Employee>();  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT p.ID, p.Name AS FirstName, p.Surname AS LastName, p.Patronymic, po.Title AS Position, d.Title AS Department, p.Telephone AS Phone, p.Date\_of\_birth AS BirthDate, p.Photo FROM Personal\_card p JOIN Post po ON p.Id\_post = po.ID JOIN Department d ON po.Id\_department = d.ID;", connection);  using var reader = command.ExecuteReader();  while (reader.Read())  {  employees.Add(new Employee  {  ID = reader.GetInt32(0),  FirstName = reader.GetString(1),  LastName = reader.GetString(2),  Patronymic = reader.GetString(3),  Position = reader.GetString(4),  Department = reader.GetString(5),  Phone = reader.GetString(6),  BirthDate = reader.GetDateTime(7),  PhotoData = reader.IsDBNull(8) ? null : (byte[])reader[8]  });  }  }  return employees;  }  public string UserName  {  get => \_userName;  set  {  \_userName = value;  OnPropertyChanged(nameof(UserName));  }  }  public string UserRole  {  get => \_userRole;  set  {  \_userRole = value;  OnPropertyChanged(nameof(UserRole));  }  }  public BitmapImage UserImage  {  get => \_userImage;  set  {  \_userImage = value;  OnPropertyChanged(nameof(UserImage));  }  }  private void LoadEmploymentRecordsForSelectedEmployee()  {  if (SelectedEmployee == null) return;  EmploymentRecords.Clear();  using var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework");  connection.Open();  var command = new SqlCommand(@"SELECT e.ID, e.Date, e.Reason, m.Title AS MixingTitle FROM Entry\_in\_the\_work\_book e LEFT JOIN Mixing m ON e.Id\_mixing = m.ID WHERE e.Id\_personal\_card = @EmployeeId", connection);  command.Parameters.AddWithValue("@EmployeeId", SelectedEmployee.ID);  using var reader = command.ExecuteReader();  while (reader.Read())  {  EmploymentRecords.Add(new EntryInWorkBook  {  ID = reader.GetInt32(0),  Date = reader.GetDateTime(1),  Reason = reader.IsDBNull(2) ? string.Empty : reader.GetString(2),  MixingTitle = reader.IsDBNull(3) ? string.Empty : reader.GetString(3)  });  }  }  private void AddRecord()  {  if (SelectedEmployee == null)  {  MessageBox.Show("Пожалуйста, выберите сотрудника.");  return;  }  AddRecordWindow addRecordWindow = new AddRecordWindow(SelectedEmployee.ID);  addRecordWindow.RecordAdded += (s, args) =>  {  LoadEmploymentRecordsForSelectedEmployee();  };  addRecordWindow.Show();  }  private void NavigateToMainWindow()  {  MainStaff mainStaffWindow = new MainStaff(\_currentUser);  mainStaffWindow.Show();  \_currentWindow.Close();  }  private void NavigateToOrders()  {  MainOrders mainOrdersWindow = new MainOrders(\_currentUser);  mainOrdersWindow.Show();  \_currentWindow.Close();  }  private void NavigateToReports()  {  MainReports mainReportsWindow = new MainReports(\_currentUser);  mainReportsWindow.Show();  \_currentWindow.Close();  }  private void NavigateToStaff()  {  MainStaff mainStaffWindow = new MainStaff(\_currentUser);  mainStaffWindow.Show();  \_currentWindow.Close();  }  }  } |

Код C# MainStaffViewModel.cs в таблице Б.37.

Таблица Б.37 – Листинг кода MainStaffViewModel.cs.

|  |
| --- |
| using PersonnelDepartment.ClassHelper;  using PersonnelDepartment.Commands;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  using PersonnelDepartment.Staff;  using System.Collections.ObjectModel;  using System.Data.SqlClient;  using System.Linq;  using System.Windows;  using System.Windows.Input;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.ViewModel  {  public class MainStaffViewModel : BaseViewModel  {  private ObservableCollection<Employee> \_employees;  private ObservableCollection<Employee> \_filteredEmployees;  private string \_searchText;  private Employee \_selectedEmployee;  private Personal\_card \_currentUser;  private Window \_currentWindow;  private string \_userName;  private string \_userRole;  private BitmapImage \_userImage;  public ObservableCollection<Employee> Employees  {  get => \_employees;  set  {  \_employees = value;  OnPropertyChanged(nameof(Employees));  FilterEmployees();  }  }  public ObservableCollection<Employee> FilteredEmployees  {  get => \_filteredEmployees;  set  {  \_filteredEmployees = value;  OnPropertyChanged(nameof(FilteredEmployees));  }  }  public string SearchText  {  get => \_searchText;  set  {  \_searchText = value;  OnPropertyChanged(nameof(SearchText));  FilterEmployees();  }  }  public Employee SelectedEmployee  {  get { return \_selectedEmployee; }  set  {  \_selectedEmployee = value;  OnPropertyChanged(nameof(SelectedEmployee));  // Когда выбран сотрудник, активируем команду редактирования  //if (\_selectedEmployee != null)  //{  // EditEmployeeCommand.Execute(null);  //}  }  }  public ICommand SearchCommand { get; }  public ICommand AddStaffCommand { get; }  public ICommand ExitCommand { get; }  public ICommand AssignRewardCommand { get; }  public ICommand NavigateToMainWindowCommand { get; }  public ICommand NavigateToEmploymentRecordCommand { get; }  public ICommand NavigateToOrdersCommand { get; }  public ICommand NavigateToReportsCommand { get; }  private ICommand \_editEmployeeCommand;  public ICommand EditEmployeeCommand => \_editEmployeeCommand ??= new RelayCommand<Employee>(OpenEditEmployeeWindow);  public MainStaffViewModel(Personal\_card currentUser, Window currentWindow)  {  \_currentUser = currentUser;  \_currentWindow = currentWindow;  \_employees = LoadEmployeesFromDatabase();  \_filteredEmployees = new ObservableCollection<Employee>(\_employees);  UserName = $"{currentUser.Surname} {currentUser.Name}";  UserRole = currentUser.Post.Title;  if (currentUser.Photo != null)  {  UserImage = ByteArrayToBitmapImage(currentUser.Photo);  }  SearchCommand = new RelayCommand(Search);  AddStaffCommand = new RelayCommand(AddStaff);  // ExitCommand = new RelayCommand(Exit);  AssignRewardCommand = new RelayCommand(AssignReward);  NavigateToMainWindowCommand = new RelayCommand(NavigateToMainWindow);  NavigateToEmploymentRecordCommand = new RelayCommand(NavigateToEmploymentRecord);  NavigateToOrdersCommand = new RelayCommand(NavigateToOrders);  NavigateToReportsCommand = new RelayCommand(NavigateToReports);  }  public void OpenEditEmployeeWindow(Employee employee)  {  if (employee != null)  {  // Открытие окна редактирования сотрудника  EditStaff editWindow = new EditStaff(employee, \_currentUser);  editWindow.Show();  \_currentWindow.Close();  }  }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  public string UserName  {  get => \_userName;  set  {  \_userName = value;  OnPropertyChanged(nameof(UserName));  }  }  public string UserRole  {  get => \_userRole;  set  {  \_userRole = value;  OnPropertyChanged(nameof(UserRole));  }  }  public BitmapImage UserImage  {  get => \_userImage;  set  {  \_userImage = value;  OnPropertyChanged(nameof(UserImage));  }  }  private void Search()  {  FilterEmployees();  }  private void FilterEmployees()  {  string searchText = SearchText?.Trim().ToLower();  if (string.IsNullOrEmpty(searchText))  {  FilteredEmployees.Clear();  foreach (var emp in Employees)  {  FilteredEmployees.Add(emp);  }  }  else  {  var filteredList = Employees.Where(emp =>  (!string.IsNullOrEmpty(emp.FirstName) && emp.FirstName.ToLower().Contains(searchText)) ||  (!string.IsNullOrEmpty(emp.LastName) && emp.LastName.ToLower().Contains(searchText)) ||  ($"{emp.FirstName} {emp.LastName}".ToLower().Contains(searchText))  ).ToList();  FilteredEmployees.Clear();  foreach (var emp in filteredList)  {  FilteredEmployees.Add(emp);  }  }  }  private ObservableCollection<Employee> LoadEmployeesFromDatabase()  {  ObservableCollection<Employee> employees = new ObservableCollection<Employee>();  using (var connection = new SqlConnection("data source=KSESHA;initial catalog=HumanResourcesDepartment;integrated security=True;encrypt=False;MultipleActiveResultSets=True;App=EntityFramework"))  {  connection.Open();  var command = new SqlCommand("SELECT p.ID, p.Name AS FirstName, p.Surname AS LastName, p.Patronymic, po.Title AS Position, d.Title AS Department, p.Telephone AS Phone, p.Date\_of\_birth AS BirthDate, p.Photo FROM Personal\_card p JOIN Post po ON p.Id\_post = po.ID JOIN Department d ON po.Id\_department = d.ID;", connection);  using (var reader = command.ExecuteReader())  {  while (reader.Read())  {  employees.Add(new Employee  {  ID = reader.GetInt32(0),  FirstName = reader.GetString(1),  LastName = reader.GetString(2),  Patronymic = reader.GetString(3),  Position = reader.GetString(4),  Department = reader.GetString(5),  Phone = reader.GetString(6),  BirthDate = reader.GetDateTime(7),  PhotoData = reader.IsDBNull(8) ? null : (byte[])reader[8]  });  }  }  }  return employees;  }  private void AddStaff()  {  var addStaff = new AddStaff(\_currentUser);  addStaff.Show();  \_currentWindow.Close();  }  //private void Exit()  //{  // var result = MessageBox.Show("Вы уверены, что хотите выйти из приложения?", "Подтверждение о закрытии", MessageBoxButton.YesNo, MessageBoxImage.Warning);  // if (result == MessageBoxResult.Yes)  // {  // Application.Current.Shutdown();  // }  //}  private void AssignReward()  {  if (SelectedEmployee == null)  {  MessageBox.Show("Выберите сотрудника из списка.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);  return;  }  var assignRewardForm = new AssignRewardForm(SelectedEmployee.ID, $"{SelectedEmployee.LastName} {SelectedEmployee.FirstName} {SelectedEmployee.Patronymic}");  assignRewardForm.ShowDialog();  }  private void NavigateToMainWindow()  {  new MainWindowInspector(\_currentUser).Show();  Application.Current.Windows.Cast<Window>().FirstOrDefault(w => w is MainStaff)?.Close();  }  private void NavigateToEmploymentRecord()  {  new MainEmploymentRecord(\_currentUser).Show();  \_currentWindow.Close();  }  private void NavigateToOrders()  {  new MainOrders(\_currentUser).Show();  \_currentWindow.Close();  }  private void NavigateToReports()  {  new MainReports(\_currentUser).Show();  \_currentWindow.Close();  }  }  } |

Код C# MainWindowInspectorViewModel.cs в таблице Б.38.

Таблица Б.38 – Листинг кода MainWindowInspectorViewModel.cs.

|  |
| --- |
| using PersonnelDepartment.Commands;  using PersonnelDepartment.EmploymentRecord;  using PersonnelDepartment.OrdersForm;  using PersonnelDepartment.RepostsForm;  using PersonnelDepartment.Staff;  using System.Windows;  using System.Windows.Input;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.ViewModel  {  public class MainWindowInspectorViewModel : BaseViewModel  {  private Personal\_card \_currentUser;  private string \_userName;  private string \_userRole;  private BitmapImage \_userImage;  private Window \_currentWindow;  public MainWindowInspectorViewModel(Personal\_card currentUser, Window currentWindow)  {  \_currentUser = currentUser;  \_currentWindow = currentWindow;  UserName = $"{currentUser.Surname} {currentUser.Name}";  UserRole = currentUser.Post.Title;  if (currentUser.Photo != null)  {  UserImage = ByteArrayToBitmapImage(currentUser.Photo);  }  StaffCommand = new RelayCommand(ExecuteStaff);  EmploymentRecordCommand = new RelayCommand(ExecuteEmploymentRecord);  OrdersCommand = new RelayCommand(ExecuteOrders);  ReportCommand = new RelayCommand(ExecuteReports);  //ExitCommand = new RelayCommand(ExecuteExit);  }  public string UserName  {  get => \_userName;  set  {  \_userName = value;  OnPropertyChanged(nameof(UserName));  }  }  public string UserRole  {  get => \_userRole;  set  {  \_userRole = value;  OnPropertyChanged(nameof(UserRole));  }  }  public BitmapImage UserImage  {  get => \_userImage;  set  {  \_userImage = value;  OnPropertyChanged(nameof(UserImage));  }  }  public ICommand StaffCommand { get; }  public ICommand EmploymentRecordCommand { get; }  public ICommand OrdersCommand { get; }  public ICommand ReportCommand { get; }  //public ICommand ExitCommand { get; }  private BitmapImage ByteArrayToBitmapImage(byte[] byteArray)  {  if (byteArray == null) return null;  using var stream = new System.IO.MemoryStream(byteArray);  var image = new BitmapImage();  image.BeginInit();  image.StreamSource = stream;  image.CacheOption = BitmapCacheOption.OnLoad;  image.EndInit();  return image;  }  private void ExecuteStaff()  {  MainStaff mainStaffWindow = new MainStaff(\_currentUser);  mainStaffWindow.Show();  \_currentWindow.Close();  }  private void ExecuteEmploymentRecord()  {  MainEmploymentRecord mainEmploymentRecordWindow = new MainEmploymentRecord(\_currentUser);  mainEmploymentRecordWindow.Show();  \_currentWindow.Close();  }  private void ExecuteOrders()  {  MainOrders mainOrdersWindow = new MainOrders(\_currentUser);  mainOrdersWindow.Show();  \_currentWindow.Close();  }  private void ExecuteReports()  {  MainReports mainReportsWindow = new MainReports(\_currentUser);  mainReportsWindow.Show();  \_currentWindow.Close();  }  }  } |

Код C# ValidatorTests.cs в таблице Б.39.

Таблица Б.39 – Листинг кода ValidatorTests.cs.

|  |
| --- |
| using Microsoft.VisualStudio.TestTools.UnitTesting;  using PersonnelDepartment;  using PersonnelDepartment.ClassHelper;  using System;  using System.IO;  using System.Linq;  namespace HumanDepartmaentTests  {  [TestClass]  public class ValidatorTests  {  private HumanResourcesDepartmentEntities dbModel;  [TestInitialize]  public void Setup()  {  dbModel = new HumanResourcesDepartmentEntities();  }  // Тест 1 - Ввод с заглавной буквы значения в поле «Имя»  [TestMethod]  public void Test\_ValidData()  {  string errorMessage;  // Проверяем ввод правильных данных для каждого поля  bool isValid = Validator.IsValidName("Иван", out errorMessage);  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidName("Иванов", out errorMessage); // Фамилия  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidName("Иванович", out errorMessage); // Отчество  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPhone("78965431234", out errorMessage); // Телефон  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidEmail("abcd@mail.ru", out errorMessage); // Электронная почта  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  string validDate = "15.03.2020";  isValid = Validator.IsValidBirthDate(validDate, out errorMessage);  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPlace("г. Тула", out errorMessage); // Место рождения  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidAddress("г. Тула, ул. Ленина, д. 15, кв. 15", out errorMessage); // Адрес регистрации  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPassportSeries("1234", out errorMessage); // Серия паспорта  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidPassportNumber("123456", out errorMessage); // Номер паспорта  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  validDate = "14.10.2022";  isValid = Validator.IsValidDateOfIssue(validDate, out errorMessage);  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidIssuedBy("МВД", out errorMessage); // Кем выдан  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidEducationInstitution("МГУ", out errorMessage); // Учебное заведение  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  // Списки  isValid = Validator.IsValidDepartment("Технический", out errorMessage); // Отдел  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidEducationLevel("Высшее образование (бакалавриат)", out errorMessage); // Уровень  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  isValid = Validator.IsValidCitizenship("Россия", out errorMessage); // Гражданство  Assert.IsTrue(isValid);  Assert.IsNull(errorMessage);  byte[] photoBytes = File.ReadAllBytes(@"C:\PDepartment\Media\face.jpg");  // Сохраняем данные в БД  var personalCard = new Personal\_card  {  Name = "Иван",  Surname = "Иванов",  Patronymic = "Иванович",  Date\_of\_birth = DateTime.Parse("15.03.2020"),  Email = "abcd@mail.ru",  Telephone = "78965431234",  Birthplace = "г. Тула",  Registration\_address = "г. Тула, ул. Ленина, д. 15, кв. 15",  Series\_and\_number = "123456",  Date\_of\_issue = DateTime.Parse("14.10.2022"),  Issued\_by\_whom = "МВД",  EducationInstitution = "МГУ",  Id\_post = 4,  Id\_education = 1,  Id\_citizenship = 1,  Children = true,  Military\_service = true,  Photo = photoBytes  };  dbModel.Personal\_card.Add(personalCard);  dbModel.SaveChanges();  // Проверка, что запись была добавлена  var addedRecord = dbModel.Personal\_card.FirstOrDefault(x => x.Name == "Иван" && x.Surname == "Иванов");  Assert.IsNotNull(addedRecord);  Assert.AreEqual("Иван", addedRecord.Name);  Assert.AreEqual("Иванов", addedRecord.Surname);  }  // Тест 2 - Ввод строчной буквы в поле «Имя»  [TestMethod]  public void Test\_InvalidName\_Lowercase()  {  string errorMessage;  bool isValid = Validator.IsValidName("иван", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Имя»", errorMessage);  }  // Тест 3 - Ввод в поле «Фамилия» чисел  [TestMethod]  public void Test\_InvalidSurname\_WithNumbers()  {  string errorMessage;  bool isValid = Validator.IsValidSurName("Иванов123", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Фамилия»", errorMessage);  }  // Тест 4 - Ввод специального символа в поле «Отчество»  [TestMethod]  public void Test\_InvalidPatronymic\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidPatronymic("Иванович@", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Отчество»", errorMessage);  }  // Тест 5 - Ввод номера с буквой в поле «Телефон»  [TestMethod]  public void Test\_InvalidPhone\_WithLetter()  {  string errorMessage;  bool isValid = Validator.IsValidPhone("78965431234a", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Телефонный номер»", errorMessage);  }  // Тест 6 - Ввод значения поля «Электронная почта» без знака @  [TestMethod]  public void Test\_InvalidEmail\_WithoutAtSign()  {  string errorMessage;  bool isValid = Validator.IsValidEmail("abcdmail.ru", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Электронная почта»", errorMessage);  }  // Тест 7 - Выбор даты в поле «Дата рождения» больше системной  [TestMethod]  public void Test\_InvalidBirthDate\_FutureDate()  {  string errorMessage;  DateTime invalidDate = DateTime.Now.AddYears(1);  string invalidDateString = invalidDate.ToString("dd.MM.yyyy"); // Преобразуем в строку  bool isValid = Validator.IsValidBirthDate(invalidDateString, out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Дата рождения»", errorMessage);  }  // Тест 8 - Ввод значения в поле «Место рождения» с прописной буквы  [TestMethod]  public void Test\_InvalidPlace\_WithLowercase()  {  string errorMessage;  bool isValid = Validator.IsValidPlace("тула", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Место рождения»", errorMessage);  }  // Тест 9 - Ввод чисел в поле «Адрес регистрации»  [TestMethod]  public void Test\_InvalidAddress\_WithNumbers()  {  string errorMessage;  bool isValid = Validator.IsValidAddress("12345", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Адрес регистрации»", errorMessage);  }  // Тест 10 - Ввод буквы в поле «Серия»  [TestMethod]  public void Test\_InvalidPassportSeries\_WithLetter()  {  string errorMessage;  bool isValid = Validator.IsValidPassportSeries("123a", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Серия паспорта»", errorMessage);  }  // Тест 11 - Ввод специального символа в поле «Номер»  [TestMethod]  public void Test\_InvalidPassportNumber\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidPassportNumber("12345@", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Номер паспорта»", errorMessage);  }  // Тест 12 - Ввод даты с специальным символом в поле «Дата выдачи»  [TestMethod]  public void Test\_InvalidDateOfIssue\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidDateOfIssue("14,10.2024", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Дата выдачи»", errorMessage);  }  // Тест 13 - Ввод чисел в поле «Кем выдан»  [TestMethod]  public void Test\_InvalidIssuedBy\_WithNumbers()  {  string errorMessage;  bool isValid = Validator.IsValidIssuedBy("12345", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Кем выдан»", errorMessage);  }  // Тест 14 - Ввод значения со специальным символом в поле «Учебное заведение»  [TestMethod]  public void Test\_InvalidEducationInstitution\_WithSpecialChar()  {  string errorMessage;  bool isValid = Validator.IsValidEducationInstitution("МГУ@", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Некорректный ввод в поле «Учебное заведение»", errorMessage);  }  // Тест 15 - Поле со списком «Отдел» пустое  [TestMethod]  public void Test\_EmptyDepartment()  {  string errorMessage;  bool isValid = Validator.IsValidDepartment("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Отдел» пустое", errorMessage);  }  // Тест 16 - Поле со списком «Должность» пустое  [TestMethod]  public void Test\_EmptyPost()  {  string errorMessage;  bool isValid = Validator.IsValidPost("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Должность» пустое", errorMessage);  }  // Тест 17 - Поле со списком «Уровень» пустое  [TestMethod]  public void Test\_EmptyEducationLevel()  {  string errorMessage;  bool isValid = Validator.IsValidEducationLevel("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Уровень» пустое", errorMessage);  }  // Тест 18 - Поле со списком «Гражданство» пустое  [TestMethod]  public void Test\_EmptyCitizenship()  {  string errorMessage;  bool isValid = Validator.IsValidCitizenship("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Поле со списком «Гражданство» пустое", errorMessage);  }  // Тест 19 - Поле "Аватар" пустое  [TestMethod]  public void Test\_InvalidPhoto\_FileNotFound()  {  string errorMessage;  bool isValid = Validator.IsValidPhoto("", out errorMessage);  Assert.IsFalse(isValid);  Assert.AreEqual("Выберите значение в поле «Аватар»", errorMessage);  }  }  } |

Код C# ByteImageToImageConverter.cs в таблице Б.40.

Таблица Б.40 – Листинг кода ByteImageToImageConverter.cs.

|  |
| --- |
| using System;  using System.Globalization;  using System.IO;  using System.Windows.Data;  using System.Windows.Media.Imaging;  namespace PersonnelDepartment.Converter // Убедитесь, что это совпадает с XAML  {  public class ByteArrayToImageConverter : IValueConverter  {  public object Convert(object value, Type targetType, object parameter, CultureInfo culture)  {  if (value is byte[] bytes && bytes.Length > 0)  {  using (var stream = new MemoryStream(bytes))  {  var image = new BitmapImage();  image.BeginInit();  image.CacheOption = BitmapCacheOption.OnLoad;  image.StreamSource = stream;  image.EndInit();  return image;  }  }  return null; // Возвращаем null, если данных нет  }  public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)  {  throw new NotImplementedException();  }  }  } |