**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Зайцева Дарья Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Зайцева Дарья Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения;
* Разработка модулей программного обеспечения;
* Тестирование программных модулей и их интеграции;
* Разработка программной документации и стандарты кодирования.

**Индивидуальное задание: ВАРИАНТ 5**

Задание выдал «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Зайцева Д.А (подпись) (Ф.И.О.)

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Зайцева Дарья Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 23.09.2024 | Анализ предметной области. Выявление требований к программе. |  |
| 24.09.2024 | Разработка технического задания. UML. Проектирование диаграммы вариантов использования. |  |
| 25.09.2024 | UML. Проектирование диаграммы последовательности. UML. Проектирование диаграммы активности. |  |
| 26.09.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 27.09.2024 | Проектирование интерфейса пользователя. Создание Wireframe эскизов. Разработка дизайна программы в соответствии с руководством по стилю. |  |
| 28.09.2024 | Разработка базы данных. Разработка словаря данных. |  |
| 30.09.2024 | Создание приложения. Форма авторизации. Создание приложения. Форма заказов. |  |
| 01.10.2024 | Создание приложения. Основные формы приложения. Разработка библиотеки классов. |  |
| 02.10.2024 | Подготовка отчетов и выгрузка документов для печати. Отладка программных модулей. |  |
| 03.10.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 04.10.2024 | Интеграционное тестирование. Разработка самодокументирующегося кода. |  |
| 05.10.2024 | Инспекция кода на соответствие стандартам кодирования. Подготовка отчета и размещение результатов в репозитории контроля версий. |  |

**Содержание**

[Задание 1. Разработка технического задания 6](#_Toc178845158)

[Описание предметной области 6](#_Toc178845159)

[Техническое задание 7](#_Toc178845160)

[Задание 2. Разработка алгоритмов и диаграмм 9](#_Toc178845161)

[Словарь данных 9](#_Toc178845162)

[UML диаграммы 12](#_Toc178845163)

[Задание 3. Разработка макетов программы 14](#_Toc178845164)

[Карта навигации 15](#_Toc178845165)

[Задание 4. Разработка программы 16](#_Toc178845166)

[Задание 5. Тестирование 22](#_Toc178845167)

[Код тестов 25](#_Toc178845168)

[Результаты тестов 27](#_Toc178845169)

[Приложение А 28](#_Toc178845170)

**Задание 1. Разработка технического задания**

**Описание предметной области**

Основной целью данного модуля является автоматизация процессов учета и обработки данных в Государственной инспекции безопасности дорожного движения (ГИБДД). Модуль позволяет сотрудникам ГИБДД прозрачно и эффективно управлять данными, отслеживать статус различных процедур и операций.

Разработка программного модуля для учета данных ДТП — это процесс создания программного продукта, который позволяет ГИБДД эффективно отслеживать и вести учет данных ДТП.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для учета, отслеживания и управления данными о ДТП, что помогает минимизировать время обработки подобных запросов в ГИБДД, обеспечить своевременную осведомлённость сотрудников о случившейся аварии.

Программный модуль позволяет ГИБДД повысить эффективность и точность учета заявок, сократить время обработки заявок, оптимизировать использование ресурсов и повысить скорость прибытия сотрудников на место аварии.

Основные функции и возможности модуля включают:

1. Фиксация ДТП: это информация, предоставленная клиентом о случившемся ДТП. Клиент оставляет новую заявку. Заявка должна содержать данные о гос. номере автомобиля, модели, описании ДТП, личную информацию (ФИО клиента, номер телефона, паспортные данные). Клиент может отредактировать свою заявку.

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки оператором ГИБДД в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке.

3. Обработка заявки: процесс, включающий вызов сотрудника на место ДТП, определение её приоритетности и назначение времени прибытия сотрудника на место ДТП. В процессе обработки может потребоваться дополнительная информация или уточнение деталей ДТП у клиента.

4. Исполнение заявки: фактическое прибытие сотрудника ГИБДД на место ДТП. На этом этапе назначенный сотрудник фиксирует правонарушение, вносит необходимые изменения или заменяет некорректные части заявки. Важно отметить, что на этом этапе может возникать необходимость составления протокола, выдачи штрафа или координации работ с другими специалистами.

5. Отчётность и информирование: важной составляющей учёта заявок на ДПТ является фиксация и отчёт о правонарушении. После завершения фиксации ДТП сотрудник ГИБДД должен предоставить отчёт о правонарушении, включая информацию о владельце ТС, гос. номере ТС, данные об аварии.

6. Мониторинг и анализ: этот этап предполагает контроль и анализ процесса фиксации ДТП. Важно отслеживать и анализировать время обработки заявок, скорость прибытия сотрудника ГИБДД на место ДТП и прочие параметры, которые могут помочь в оптимизации и улучшении процесса.

**Техническое задание**

1. Общие сведения

1.1. Наименование проекта: Разработка программного модуля для фиксации ДТП сотрудниками ГИБДД.

1.2. Заказчик: ГИБДД «Мяу».

1.3. Исполнитель: ГИБДД «UWU».

2. Функциональные требования

2.1. Возможность добавления ДТП в базу данных с указанием следующих параметров:

- Номер ДТП;

- Дата ДТП;

- Адрес места ДТП;

- Описание ДТП;

- Описание аварии;

- Степень серьёзности ДТП;

- Дата создания заявки;

- Дата обновления заявки.

2.2. Возможность редактирования заявок:

- Изменение даты обновления;

- Изменение описания ДТП;

- Изменение степени серьёзности ДТП.

2.3. Возможность отслеживания статуса ДТП через таблицу мероприятия:

- Отображение списка мероприятий;

- Получение уведомлений о смене описания ДТП;

- Поиск заявки по номеру или по параметрам.

2.4. Возможность назначения ответственных за фиксацию ДТП:

- Добавление сотрудника к мероприятию;

- Сотрудника может добавлять комментарии на форме заявки и фиксировать информацию о ДТП.

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

- Информативные уведомления и подсказки.

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

4. Требования к реализации

4.1. Язык программирования: C#;

4.2. СУБД: MS SQL Server;

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

6. Руководство по стилю

6.1. Шрифт: Times New Roman 14.

6.2. Цветовая схема:

Основной: e0ffdd (мятный);

Второстепенный: fdf5e6 (бежевый).

**Задание 2. Разработка алгоритмов и диаграмм**

**Словарь данных**

**Таблица 1: Мероприятия**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Is Null | Описание |
| 1 | Мероприятие\_ID | int |  | Идентификатор мероприятия |
|  | ДТП\_ID | int |  | Идентификатор ДТП |
|  | Дата\_мероприятия | datetime |  | Дата мероприятия |
|  | Описание\_мероприятия | nvarchar(255) |  | Описание мероприятия |
|  | Сотрудник\_ID | int |  | Идентификатор сотрудника |

**Таблица 2: Автомобили**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Is Null | Описание |
| 1 | Авто\_ID | int |  | Идентификатор авто |
|  | Госномер | nvarchar(20) |  | Гос. Номер ТС |
|  | Марка | nvarchar(50) |  | Марка ТС |
|  | Модель | nvarchar(50) |  | Модель ТС |
|  | Год | int |  | Год выпуска ТС |
|  | Цвет | nvarchar(30) | 1 | Цвет ТС |

**Таблица 3: Документы**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Is Null | Описание |
| 1 | Документ\_ID | int |  | Идентификатор документа |
|  | ДТП\_ID | int |  | Идентификатор ДТП |
|  | Тип\_документа | nvarchar(50) |  | Тип документа |
|  | Номер\_документа | nvarchar(50) |  | Номер документа |
|  | Дата\_выдачи\_документа | datetime | 1 | Дата выдачи документа |
|  | Дата\_истечения | datetime | 1 | Дата истечения срока дока |

**Таблица 4: ДТП**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Is Null | Описание |
| 1 | ДТП\_ID | int |  | Идентификатор ДТП |
|  | Дата | datetime |  | Дата ДТП |
|  | Место | nvarchar(255) |  | Место ДТП |
|  | Описание | nvarchar(500) | 1 | Описание ДТП |
|  | Степень\_серьезности | nvarchar(50) | 1 | Степень серьёзности ДТП |
|  | Создано | datetime | 1 | Дата создания ДТП |
|  | Обновлено | datetime | 1 | Дата обновления ДТП |

**Таблица 5: Участники**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Is Null | Описание |
| 1 | Участник\_ID | int |  | Идентификатор участника |
|  | ДТП\_ID | int |  | Идентификатор ДТП |
|  | ФИО | nvarchar(100) |  | ФИО участника ДТП |
|  | Адрес | nvarchar(255) | 1 | Адрес проживания участника |
|  | Телефон | nvarchar(20) | 1 | Номер телефона участника |
|  | ТС | int |  | Гос. Номер ТС участника |
|  | Документ | int | 1 | Идентификатор Документа |

**Таблица 6: Сотрудники**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ключ | Наименование | Тип данных | Is Null | Описание |
| 1 | Сотрудник\_ID | int |  | Идентификатор сотрудника |
|  | ФИО | nvarchar(100) |  | ФИО сотрудника |
|  | Должность | nvarchar(100) |  | Должность сотрудника |
|  | Телефон | nvarchar(20) | 1 | Номер телефона сотрудника |
|  | Email | nvarchar(100) | 1 | Адрес почты сотрудника |



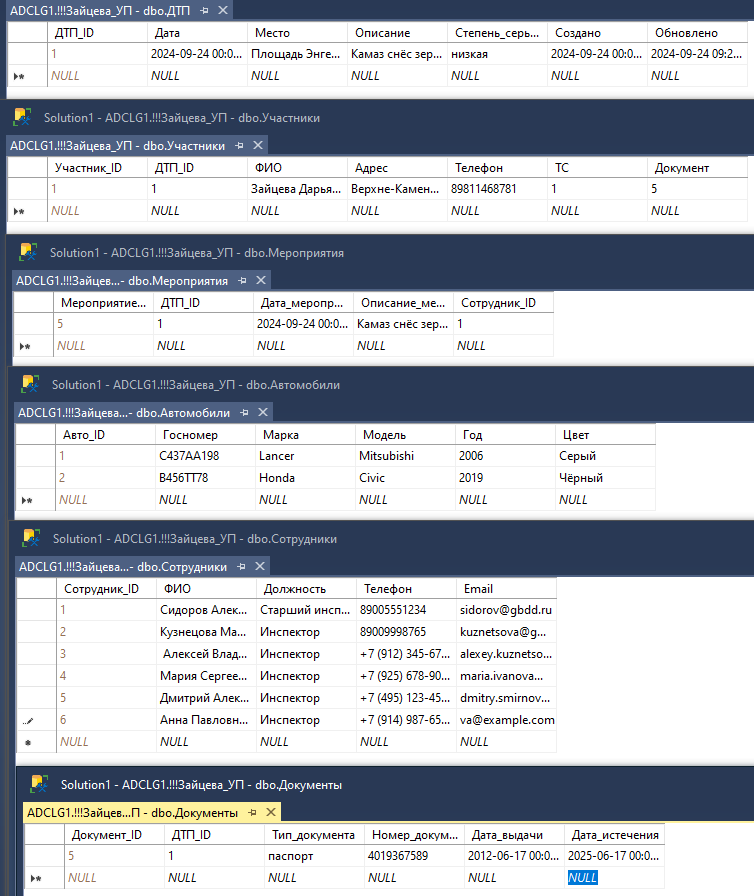
Рисунок 1 – ER модель БД ГИБДД

Рисунок 2 – Заполнение таблиц данными

**UML диаграммы**



Рисунок 3 – Диаграмма вариантов использования



Рисунок 4 – Диаграмма последовательности



Рисунок 5 – Диаграмма активности

**Задание 3. Разработка макетов программы**

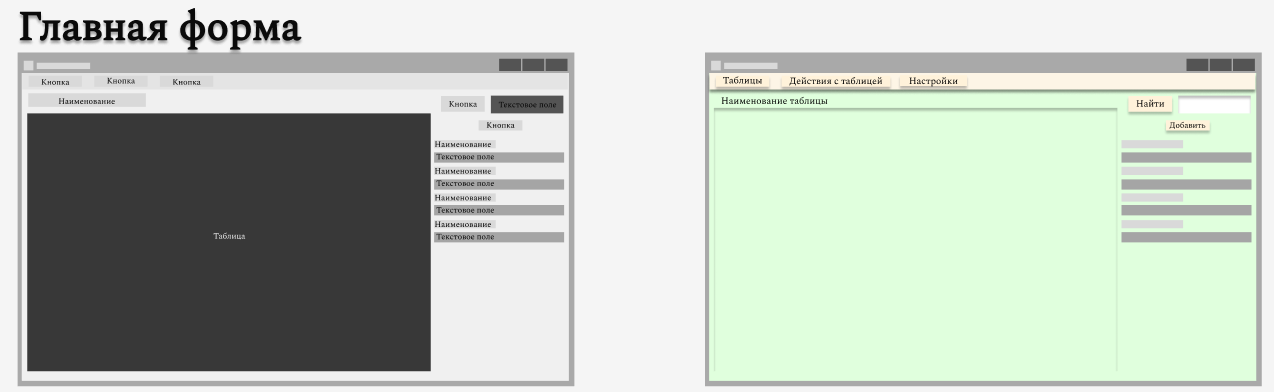


Рисунок 6 – Интерфейс главной формы



Рисунок 7 – Интерфейс главной формы с открытым меню

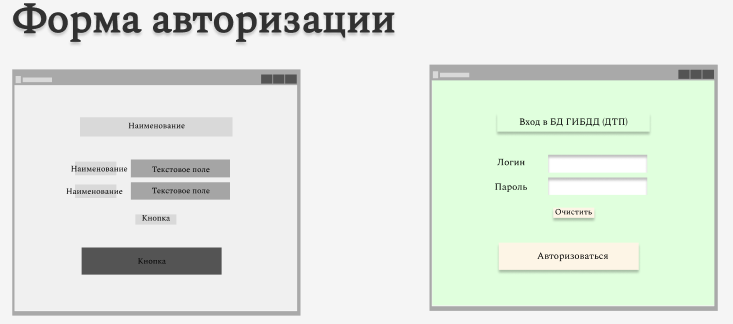


Рисунок 8 – Интерфейс формы авторизации

**Карта навигации**



Рисунок 9 – Карта навигации

**Задание 4. Разработка программы**

Код программы находятся в Приложении А.

При входе в программу появляется окно авторизации. Для дальнейшего перехода в главное меню программы необходимо ввести логин пользователя и его пароль. В качестве логина выступает ФИО, в качестве пароля – номер телефона (Рисунок 10).

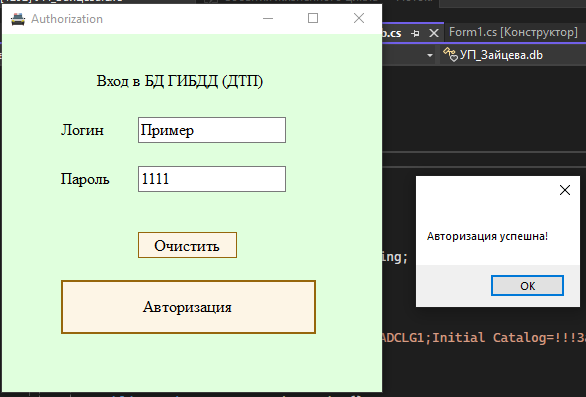


Рисунок 10 – Окно авторизации

После успешной авторизации пользователь попадает в главное меню (Рисунок 11).

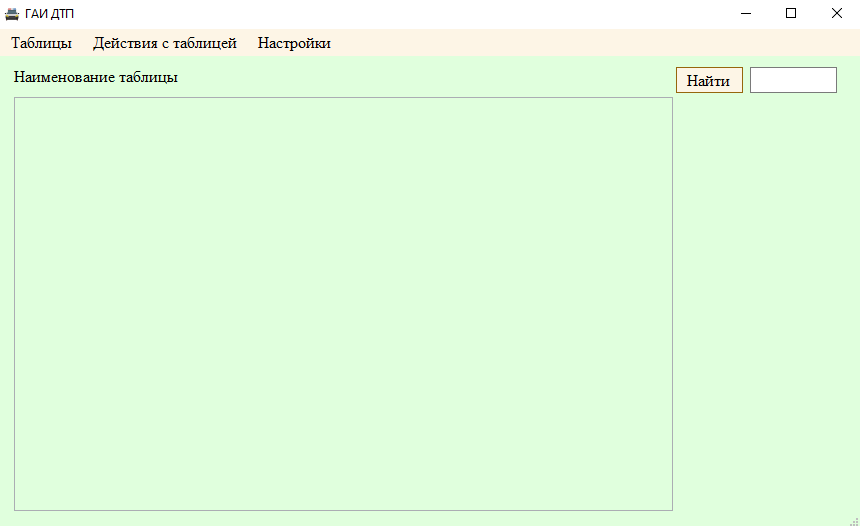


Рисунок 11 – Вход в главное меню

Для открытия любой таблицы и вывода её на экран необходимо в облавти меню нажать на кнопку «Таблицы». В нашем случае мы входим в таблицу «Сотрудники» (Рисунок 12).

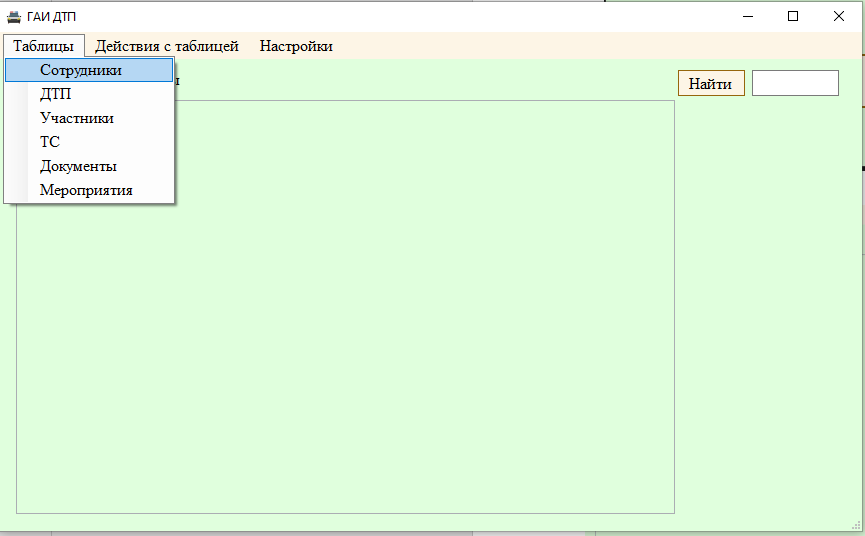


Рисунок 12 – Главное меню с открытым меню «Таблицы»

После нажатия но кнопку «Сотрудники» в области меню «Таблицы», на окне отобража.тся данные таблицы «Сотрудники» (Рисунок 13).

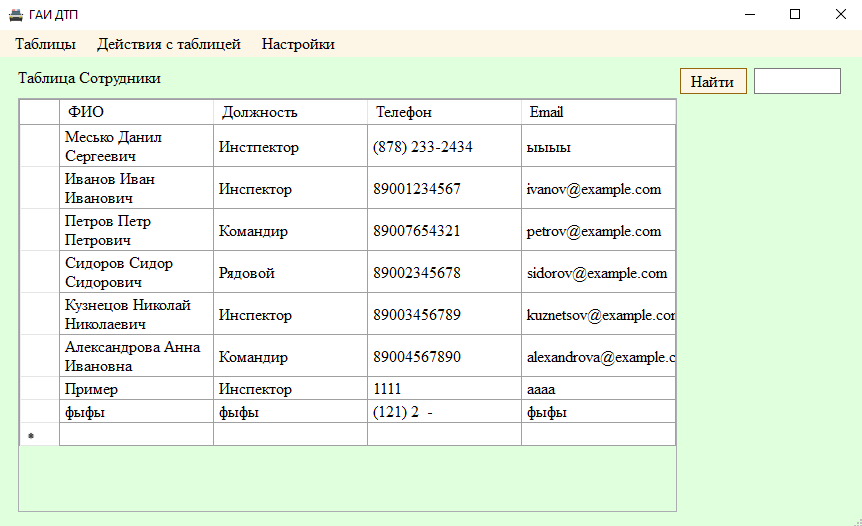


Рисунок 13 – Таблица «Сотрудники»

Для проведения поиска по таблице необходимо ввести данные поиска в поле ввода рядом с кнопкой «Найти». В нашем случае мы ищем все данные о сотрудниках с должностью «Командир» (Рисунок 14).

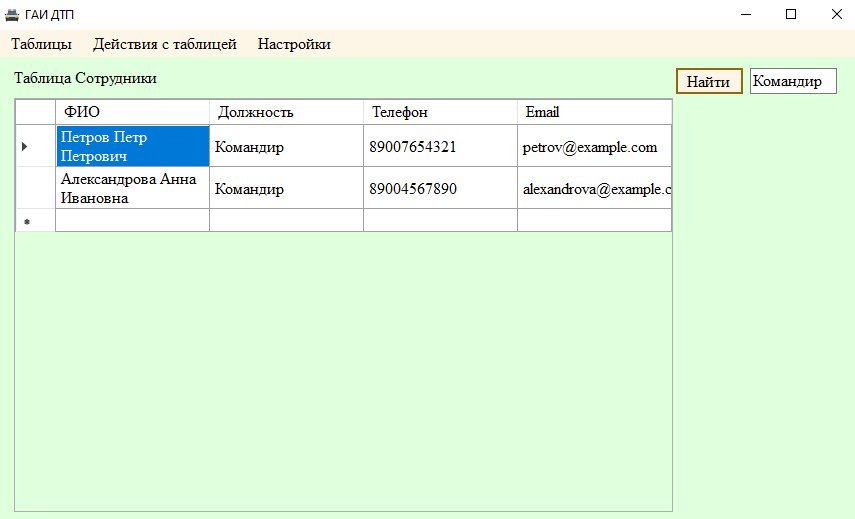


Рисунок 14 – Поиск по таблице «Сотрудники» с входящими данными «Командир»

Для добавления данных в таблицу нужно в блоке меню нажать на кнопку «Действия с таблицей», выбираем «Добавить». После появится поле для ввода данных. Вводим необходимые данные (Рисунок 15).

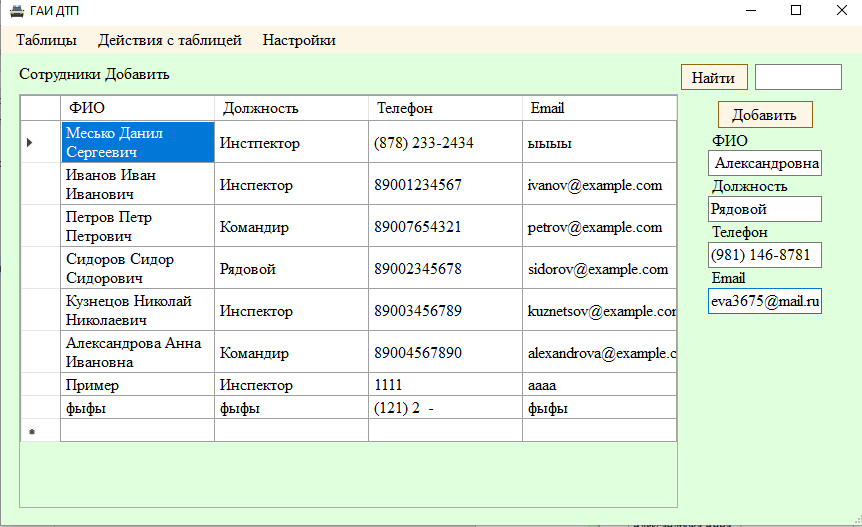


Рисунок 15 – Добавление данных в таблицу «Сотрудники»

После ввода данных нажимаем на кнопку «Добавить», после чего таблица автоматически обнавляется (Рисунок 16).

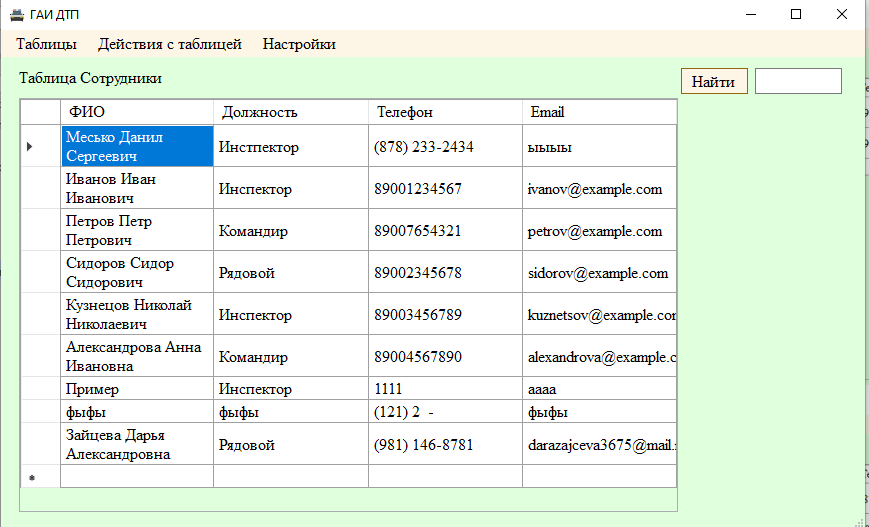


Рисунок 16 – Обновленная таблица «Сотрудники»

Для изменения данных в таблицу нужно в блоке меню нажать на кнопку «Действия с таблицей», выбираем «Изменить». После появится поле для изменения данных. Изменяем необходимые данные (Рисунок 17).

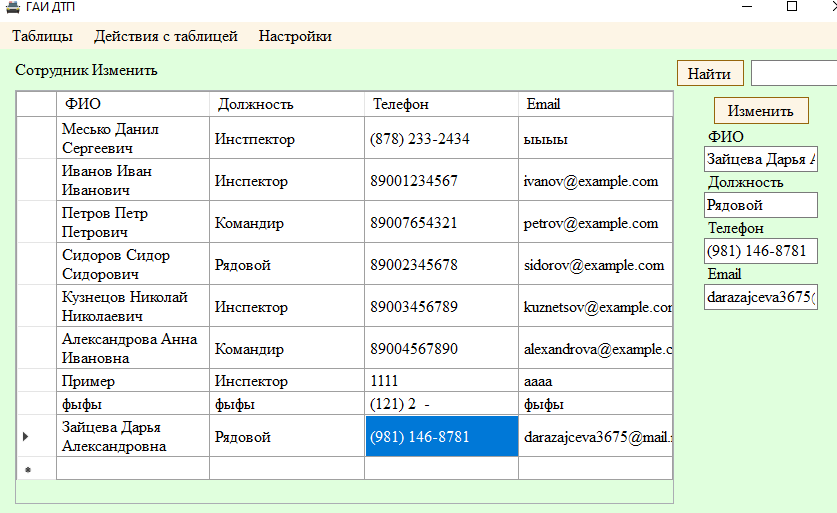


Рисунок 17 – Изменение данных в таблице «Сотрудники»

После изменения данных нажимаем на кнопку «Изменить», после чего таблица автоматически обнавляется (Рисунок 18).

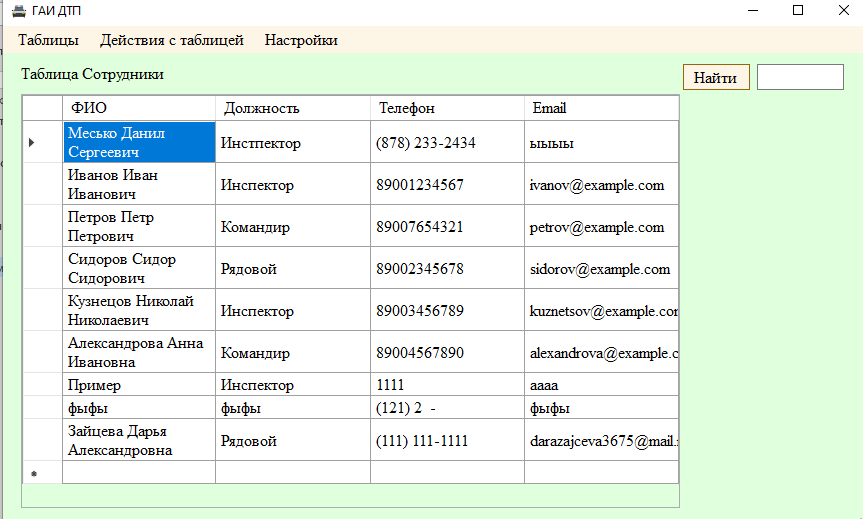


Рисунок 18 – Обновленная таблица «Сотрудники»

Для удаления данных в таблицу нужно в блоке меню нажать на кнопку «Действия с таблицей», выбираем «Удалить». Для корректного удаления необходимо выделить всю строку, которую необходимо удалить (Рисунок 19).

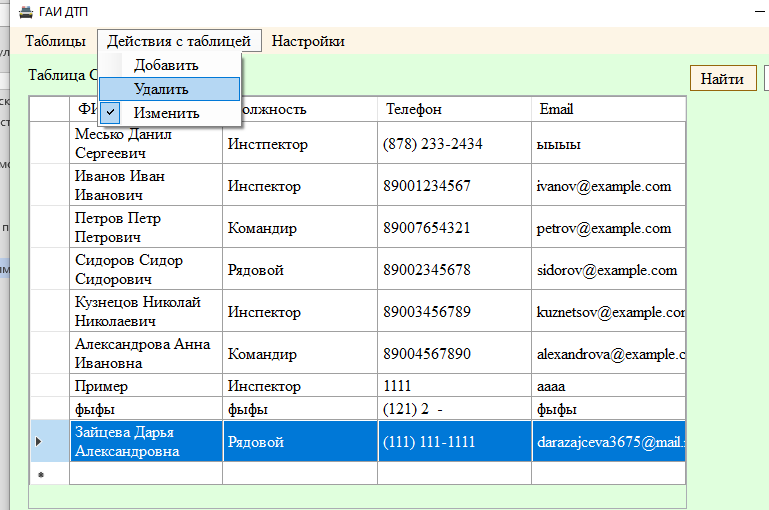


Рисунок 19 – Удаление данных из таблицы «Сотрудники»

После нажатия на кнопку удаления, появляется окно для подтверждения удаления (Рисунок 20).

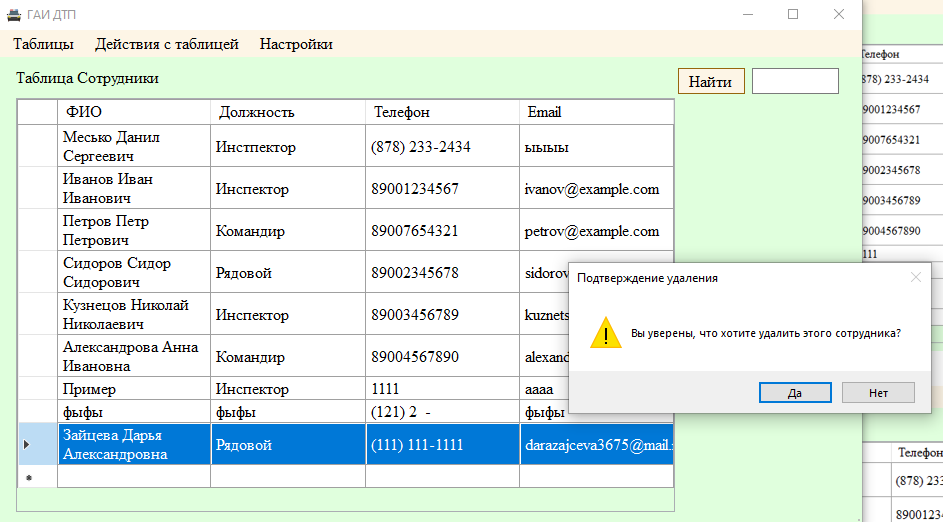


Рисунок 20 – Подтверждение удаления

После успешного удаления появляется уведомление о том, что сотрудник успешно удалён (Рисунок 21).

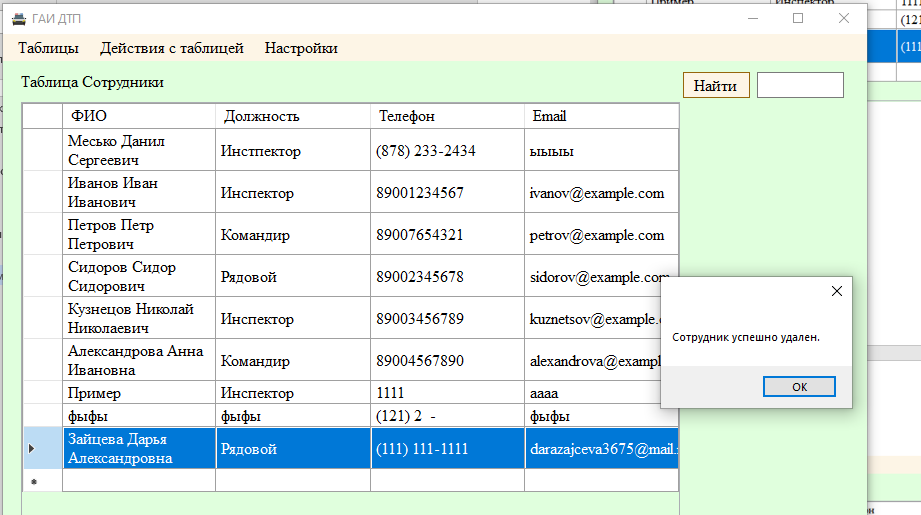


Рисунок 21 – Успешное удаление

После успешного удаления таблица автоматически обновляется (Рисунок 22).

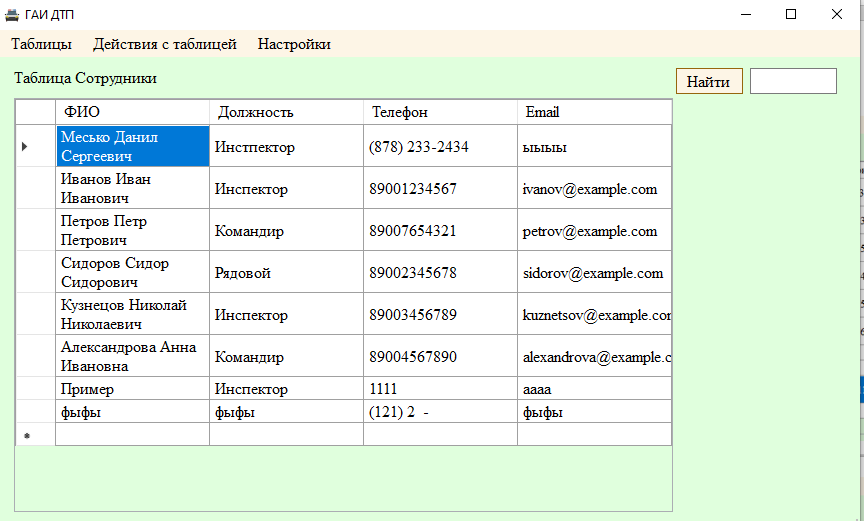


Рисунок 22 – Обновленная таблица «Сотрудники»

Для проверки работоспособности таблицы «ДТП» проведём всё те же действия, что и на рисунках выше (Рисунок 12-22). Итог проведённых действий преведён ниже (Рисунок 23-27)

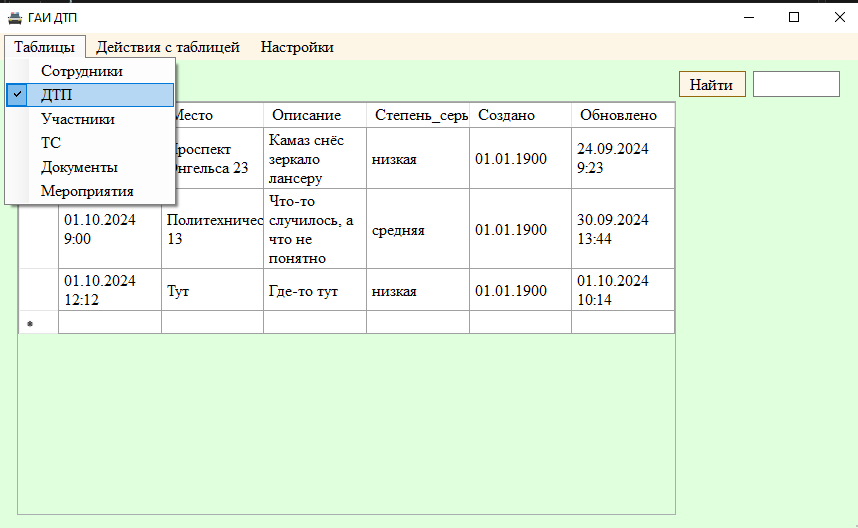


Рисунок 23 – Вывод такблицы «ДТП»

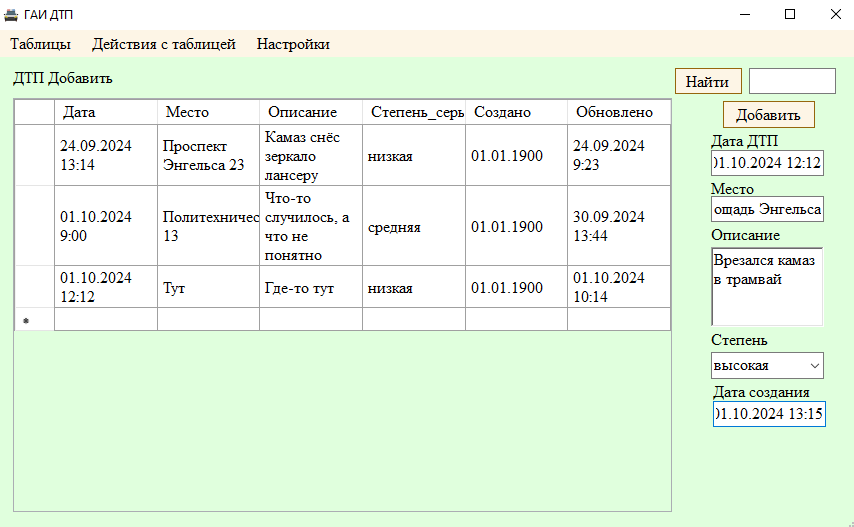


Рисунок 24 – Добавление данных в таблицу «ДТП»



Рисунок 25 – Обновленная таблица «ДТП»

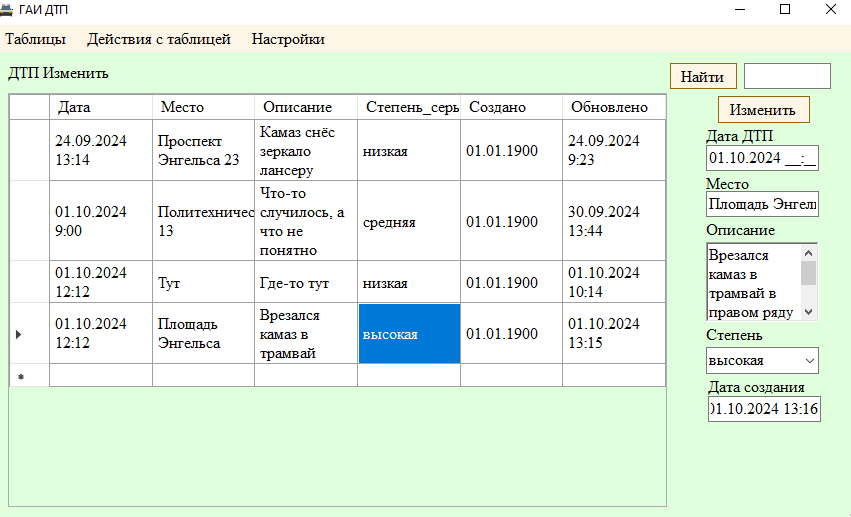


Рисунок 26 – Изменения данных в таблице ДТП

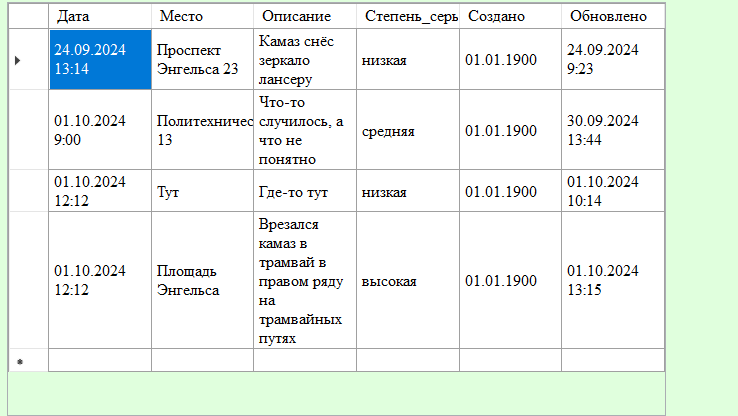


Рисунок 27 – Обновленная таблица «ДТП»

**Задание 5. Тестирование**

**Таблица 7: Аннотация теста**

|  |  |
| --- | --- |
| **Название проекта** | УП\_Зайцева |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Зайцева Дарья |
| **Дата(ы) теста** | 03.10.2024 |

**Таблица 8: Тестовый пример №1**

|  |  |
| --- | --- |
| **Тестовый пример** | ТС\_ПИ\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка обеспечения безопасности программы путём ввода не правильного логина и пароля. |
| **Краткое изложение теста** | Тест проверяет невозможность авторизации пользователя |
| **Этапы теста** | 1. Написать логин в текстовое поле формы авторизация. 2. Написать пароль в текстовое поле формы авторизация. 3. Проверить, что записи с таким логином и паролем нет в БД 4. Выводится ошибка |
| **Тестовые данные** | Логин: «1111», пароль: «1111» |
| **Ожидаемый результат** | Появляется сообщение об ошибке: «Неправильное имя пользователя или пароль» |
| **Фактический результат** | Появляется сообщение об ошибке: «Неправильное имя пользователя или пароль» |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь не авторизован, введён неправильный логин или/и пароль |

**Таблица 9: Тестовый пример №2**

|  |  |
| --- | --- |
| **Тестовый пример** | ТС\_ПИ\_2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка обеспечения доступа к таблице «Сотрудники», сотрудника с должностью «Инспектор» |
| **Краткое изложение теста** | Тест проверяет возможность «Инспектора» просматривать данные таблицы «Сотрудники» |
| **Этапы теста** | 1. Написать логин «Инспектора» в текстовое поле 2. Написать пароль «Инспектора» в текстовое поле 3. Проверить, является ли сотрудник «Инспектором» по БД 4. Успешная авторизация 5. В пункте меню «Таблицы» выбрать кнопку «Сотрудники» 6. Успешное отображение таблицы «Сотрудники» |
| **Тестовые данные** | Логин: «Зайцева Дарья Александровна», пароль: «(878) 233-2434» |
| **Ожидаемый результат** | Пользователь остаётся на главной форме, с отображённой таблицей «Сотрудники» |
| **Фактический результат** | Пользователь остаётся на главной форме, с отображённой таблицей «Сотрудники» |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь ввёл правильный логин и пароль для «Инспектора» |

**Таблица 10: Тестовый пример №3**

|  |  |
| --- | --- |
| **Тестовый пример** | ТС\_ПИ\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Попытка входа уволенного сотрудника |
| **Краткое изложение теста** | Тест проверяет возможность удалённых из БД сотрудников входить в программу |
| **Этапы теста** | 1. Написать логин удалённого сотрудника в текстовое поле 2. Написать пароль удалённого сотрудника в текстовое поле 3. Проверить, что записи с таким же логином и паролем нет в БД 4. Выводится ошибка |
| **Тестовые данные** | Логин: «фыфы», пароль: «1212» |
| **Ожидаемый результат** | Появляется сообщение об ошибке: «Неправильное имя пользователя или пароль» |
| **Фактический результат** | Появляется сообщение об ошибке: «Неправильное имя пользователя или пароль» |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователя предварительно удалили из таблицы «Сотрудники» |

**Таблица 11: Тестовый пример №4**

|  |  |
| --- | --- |
| **Тестовый пример** | ТС\_ПИ\_4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка обеспечения доступа к таблице «Документы», сотрудника с должностью «Командир» |
| **Краткое изложение теста** | Тест проверяет возможность «Командира» просматривать данные таблицы «Документы» |
| **Этапы теста** | 1. Написать логин «Командира» в текстовое поле 2. Написать пароль «Командира» в текстовое поле 3. Проверить, является ли сотрудник «Командира» по БД 4. Успешная авторизация 5. В пункте меню «Таблицы» выбрать кнопку «Документы» 6. Успешное отображение таблицы «Документы» |
| **Тестовые данные** | Логин: «Петров Петр Петрович», пароль: «89007654321» |
| **Ожидаемый результат** | Пользователь остаётся на главной форме, с отображённой таблицей «Документы» |
| **Фактический результат** | Пользователь остаётся на главной форме, с отображённой таблицей «Документы» |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь ввёл правильный логин и пароль для «Командира» |

**Таблица 12: Тестовый пример №5**

|  |  |
| --- | --- |
| **Тестовый пример** | ТС\_ПИ\_5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка обеспечения доступа к таблице «ДТП», сотрудника с должностью «Рядовой» |
| **Краткое изложение теста** | Тест проверяет возможность «Рядового» просматривать данные таблицы «ДТП» |
| **Этапы теста** | 1. Написать логин «Рядового» в текстовое поле 2. Написать пароль «Рядового» в текстовое поле 3. Проверить, является ли сотрудник «Рядового» по БД 4. Успешная авторизация 5. В пункте меню «Рядового» выбрать кнопку «ДТП» 6. Успешное отображение таблицы «ДТП» |
| **Тестовые данные** | Логин: «Сидоров Сидор Сидорович», пароль: «89002345678» |
| **Ожидаемый результат** | Пользователь остаётся на главной форме, с отображённой таблицей «Документы» |
| **Фактический результат** | Пользователь остаётся на главной форме, с отображённой таблицей «ДТП» |
| **Статус** | Успешно |
| **Предварительное условие** | Пользователь ввёл правильный логин и пароль для «Рядового» |

**Результаты тестов**

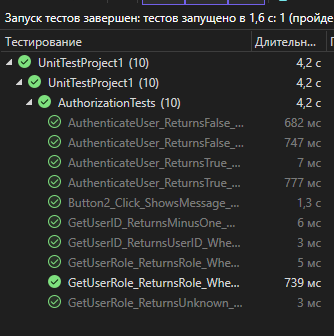


Рисунок 28 – Результат проведения UnitTest

**Приложение А**

Код программы

Form 1:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Reflection.Emit;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ToolBar;

namespace УП\_Зайцева

{

public partial class Form1 : Form

{

private int userid;

private readonly db DataBase;

private readonly string connectionString;

public Form1(int userid)

{

InitializeComponent();

this.userid = userid;

DataBase = new db();

connectionString = DataBase.GetConnectionString();

string userRole = GetUserRole(userid, DataBase.GetConnectionString());

dataGridView1.DefaultCellStyle.WrapMode = DataGridViewTriState.True;

dataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;

}

private string GetUserRole(int Сотрудник\_ID, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Должность FROM Сотрудники WHERE Сотрудник\_ID = @Сотрудник\_ID";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Сотрудник\_ID", Сотрудник\_ID);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

}

}

return "Unknown";

}

private void настройкиToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void сменитьПользователяToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Restart();

}

private void сотрудникиToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Сотрудников();

}

private void Отобразить\_Сотрудников()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_дтп.Visible = false;

panel\_МЕРОПРИЯТИЯ.Visible = false;

panel\_Сотрудники.Visible = false;

label1.Text = "Таблица Сотрудники";

connection.Open();

string query = @"SELECT s.Сотрудник\_ID, s.ФИО, s.Должность, s.Телефон, s.Email

FROM Сотрудники s ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Сотрудники = new DataTable();

adapter.Fill(Сотрудники);

dataGridView1.DataSource = Сотрудники;

dataGridView1.Columns["Сотрудник\_ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о сотрудники: " + ex.Message);

}

}

private void Найти\_Сотрудника(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT s.Сотрудник\_ID, s.ФИО, s.Должность, s.Телефон, s.Email

FROM Сотрудники s

WHERE s.Сотрудник\_ID LIKE @searchValue OR s.ФИО LIKE @searchValue OR s.Должность LIKE @searchValue

OR s.Телефон LIKE @searchValue OR s.Email LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Сотрудники = new DataTable();

adapter.Fill(Сотрудники);

dataGridView1.DataSource = Сотрудники;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_Сотрудника(int Сотрудник\_ID)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Сотрудники WHERE Сотрудник\_ID = @Сотрудник\_ID";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Сотрудник\_ID", Сотрудник\_ID);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Сотрудник успешно удален.");

}

else

{

MessageBox.Show("Произошла ошибка при удалении сотрудника.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении сотрудника: " + ex.Message);

}

}

private void Добавить\_Сотрудника()

{

Выполнить\_Запрос\_Сотрудники("Insert into Сотрудники(ФИО, Должность, Телефон, Email) " +

"VALUES(@ФИО, @Должность, @Телефон, @Email);");

}

private void Изменить\_Сотрудника()

{

Выполнить\_Запрос\_Сотрудники("Update Сотрудники set ФИО=@ФИО, Должность=@Должность, Телефон=@Телефон, Email=@Email Where Сотрудник\_ID=@Сотрудник\_ID;");

}

private void Выполнить\_Запрос\_Сотрудники(string query)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand cmd = new SqlCommand(query, connection))

{

string Сотрудник\_ФИО = textBox2.Text;

string Сотрудник\_Должность = textBox3.Text;

string Сотрудник\_Телефон = maskedTextBox1.Text;

string Сотрудник\_Почта = textBox4.Text;

string ID\_Сотр = textBox6.Text;

cmd.Parameters.AddWithValue("@ФИО", Сотрудник\_ФИО);

cmd.Parameters.AddWithValue("@Должность", Сотрудник\_Должность);

cmd.Parameters.AddWithValue("@Телефон", Сотрудник\_Телефон);

cmd.Parameters.AddWithValue("@Email", Сотрудник\_Почта);

cmd.Parameters.AddWithValue("@Сотрудник\_ID", ID\_Сотр);

cmd.ExecuteNonQuery();

}

textBox2.Clear();

textBox3.Clear();

maskedTextBox1.Clear();

textBox4.Clear();

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

}

private void дТПToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_ДТП();

}

private void Отобразить\_ДТП()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_дтп.Visible = false;

panel\_МЕРОПРИЯТИЯ.Visible = false;

panel\_Сотрудники.Visible = false;

label1.Text = "Таблица ДТП";

connection.Open();

string query = @"SELECT d.ДТП\_ID, d.Дата, d.Место, d.Описание, d.Степень\_серьезности, d.Создано, d.Обновлено

FROM ДТП d ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable ДТП = new DataTable();

adapter.Fill(ДТП);

dataGridView1.DataSource = ДТП;

dataGridView1.Columns["ДТП\_ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о ДТП: " + ex.Message);

}

}

private void Найти\_ДТП(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT d.ДТП\_ID, d.Дата, d.Место, d.Описание, d.Степень\_серьезности, d.Создано, d.Обновлено

FROM ДТП d

WHERE d.ДТП\_ID LIKE @searchValue OR d.Дата LIKE @searchValue OR d.Место LIKE @searchValue OR d.Описание LIKE @searchValue

OR d.Степень\_серьезности LIKE @searchValue OR d.Создано LIKE @searchValue OR d.Обновлено LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable ДТП = new DataTable();

adapter.Fill(ДТП);

dataGridView1.DataSource = ДТП;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_ДТП(int ДТП\_ID)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM ДТП WHERE ДТП\_ID = @ДТП\_ID";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ДТП\_ID", ДТП\_ID);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("ДТП успешно удалено.");

}

else

{

MessageBox.Show("Произошла ошибка при удалении ДТП.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении ДТП: " + ex.Message);

}

}

private void Добавить\_ДТП()

{

Выполнить\_Запрос\_ДТП("Insert into ДТП(ДТП.Дата, Место, Описание, Степень\_серьезности, Создано) " +

"VALUES(@Дата, @Место, @Описание, @Степень\_серьезности, @Создано);");

}

private void Изменить\_ДТП()

{

Выполнить\_Запрос\_ДТП("Update ДТП set Дата=@Дата, Место=@Место, Описание=@Описание, Степень\_серьезности=@Степень\_серьезности, Создано=@Создано Where ДТП\_ID=@ДТП\_ID;");

}

private void Выполнить\_Запрос\_ДТП(string query)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand cmd = new SqlCommand(query, connection))

{

DateTime ДТП\_Дата;

if (!DateTime.TryParse(maskedTextBox2.Text, out ДТП\_Дата))

{

MessageBox.Show("Некорректный формат даты.");

return;

}

string ДТП\_Место = textBox5.Text;

string ДТП\_Описание = richTextBox1.Text;

string ДТП\_Степень\_серьезности = comboBox1.Text;

string ДТП\_Создано = textBox6.Text;

string ID\_ДТП = textBox7.Text;

cmd.Parameters.AddWithValue("@Дата", ДТП\_Дата);

cmd.Parameters.AddWithValue("@Место", ДТП\_Место);

cmd.Parameters.AddWithValue("@Описание", ДТП\_Описание);

cmd.Parameters.AddWithValue("@Степень\_серьезности", ДТП\_Степень\_серьезности);

cmd.Parameters.AddWithValue("@Создано", ДТП\_Создано);

cmd.Parameters.AddWithValue("@ДТП\_ID", ID\_ДТП);

cmd.ExecuteNonQuery();

}

maskedTextBox2.Clear();

textBox5.Clear();

richTextBox1.Text = "";

comboBox1.Text = "";

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

}

private void участникиToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach(ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Участников();

}

private void Отобразить\_Участников()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_дтп.Visible = false;

panel\_МЕРОПРИЯТИЯ.Visible = false;

panel\_Сотрудники.Visible = false;

label1.Text = "Таблица Участники";

connection.Open();

string query = @"SELECT u.Участник\_ID, u.ДТП\_ID, u.ФИО, u.Адрес, u.Телефон, u.ТС, u.Документ

FROM Участники u ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Участники = new DataTable();

adapter.Fill(Участники);

dataGridView1.DataSource = Участники;

dataGridView1.Columns["Участник\_ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о участниках: " + ex.Message);

}

}

private void тСToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_ТС();

}

private void Отобразить\_ТС()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_дтп.Visible = false;

panel\_МЕРОПРИЯТИЯ.Visible = false;

panel\_Сотрудники.Visible = false;

label1.Text = "Таблица ТС";

connection.Open();

string query = @"SELECT a.Авто\_ID, a.Госномер, a.Марка, a.Модель, a.Год, a.Цвет

FROM Автомобили a ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable ТС = new DataTable();

adapter.Fill(ТС);

dataGridView1.DataSource = ТС;

dataGridView1.Columns["Авто\_ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о транспортных средствах: " + ex.Message);

}

}

private void документыToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Документы();

}

private void Отобразить\_Документы()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_дтп.Visible = false;

panel\_МЕРОПРИЯТИЯ.Visible = false;

panel\_Сотрудники.Visible = false;

label1.Text = "Таблица Документы";

connection.Open();

string query = @"SELECT doc.Документ\_ID, doc.ДТП\_ID, doc.Тип\_документа, doc.Номер\_документа, doc.Дата\_выдачи, doc.Дата\_истечения

FROM Документы doc ";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Документы = new DataTable();

adapter.Fill(Документы);

dataGridView1.DataSource = Документы;

dataGridView1.Columns["Документ\_ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о документах: " + ex.Message);

}

}

private void мероприятияToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

foreach (ToolStripMenuItem a in таблицыToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Отобразить\_Мероприятия();

}

private void Отобразить\_Мероприятия()

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

panel\_дтп.Visible = false;

panel\_МЕРОПРИЯТИЯ.Visible = false;

panel\_Сотрудники.Visible = false;

label1.Text = "Таблица Мероприятия";

connection.Open();

string query = @"SELECT m.Мероприятие\_ID, m.ДТП\_ID, m.Дата\_мероприятия, m.Описание\_мероприятия, m.Сотрудник\_ID

FROM Мероприятия m";

using (SqlCommand command = new SqlCommand(query, connection))

{

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Мероприятия = new DataTable();

adapter.Fill(Мероприятия);

dataGridView1.DataSource = Мероприятия;

dataGridView1.Columns["Мероприятие\_ID"].Visible = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при загрузке данных о мероприятиях: " + ex.Message);

}

}

private void Найти\_Мероприятие(string searchValue)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = @"SELECT m.Мероприятие\_ID, m.ДТП\_ID, m.Дата\_мероприятия, m.Описание\_мероприятия, m.Сотрудник\_ID

FROM Мероприятия m

WHERE m.Мероприятие\_ID LIKE @searchValue OR m.ДТП\_ID LIKE @searchValue OR m.Дата\_мероприятия LIKE @searchValue

OR m.Описание\_мероприятия LIKE @searchValue OR m.Сотрудник\_ID LIKE @searchValue";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@searchValue", "%" + searchValue + "%");

SqlDataAdapter adapter = new SqlDataAdapter(command);

DataTable Мероприятия = new DataTable();

adapter.Fill(Мероприятия);

dataGridView1.DataSource = Мероприятия;

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при выполнении поиска: " + ex.Message);

}

}

private void Удалить\_Мероприятие(int Мероприятие\_ID)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "DELETE FROM Мероприятия WHERE Мероприятие\_ID = @Мероприятие\_ID";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Мероприятие\_ID", Мероприятие\_ID);

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Мероприятие успешно удалено.");

}

else

{

MessageBox.Show("Произошла ошибка при удалении мероприятия.");

}

}

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при удалении мероприятия: " + ex.Message);

}

}

private void Добавить\_Меро()

{

Выполнить\_Запрос\_Меро("Insert into Мероприятия(ДТП\_ID, Дата\_мероприятия, Описание\_мероприятия, Сотрудник\_ID) " +

"VALUES(@ДТП\_ID, @Дата\_мероприятия, @Описание\_мероприятия, @Сотрудник\_ID);");

}

private void Изменить\_Меро()

{

Выполнить\_Запрос\_Меро("Update Мероприятия set ДТП\_ID=@ДТП\_ID, Дата\_мероприятия=@Дата\_мероприятия, Описание\_мероприятия=@Описание\_мероприятия, Сотрудник\_ID=@Сотрудник\_ID Where Мероприятие\_ID=@Мероприятие\_ID;");

}

private void Выполнить\_Запрос\_Меро(string query)

{

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand cmd = new SqlCommand(query, connection))

{

string Меро\_ID\_ДТП = domainUpDown1.Text;

string Меро\_Дата = maskedTextBox4.Text;

string Меро\_Описание = richTextBox2.Text;

string Меро\_ID\_Сотрудника = domainUpDown2.Text;

string ID\_Меро = textBox8.Text;

cmd.Parameters.AddWithValue("@ДТП\_ID", Меро\_ID\_ДТП);

cmd.Parameters.AddWithValue("@Дата\_мероприятия", Меро\_Дата);

cmd.Parameters.AddWithValue("@Описание\_мероприятия", Меро\_Описание);

cmd.Parameters.AddWithValue("@Сотрудник\_ID", Меро\_ID\_Сотрудника);

cmd.Parameters.AddWithValue("@Мероприятие\_ID", ID\_Меро);

cmd.ExecuteNonQuery();

}

domainUpDown1.Text = "";

maskedTextBox4.Clear();

richTextBox2.Text = "";

domainUpDown2.Text = "";

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка: " + ex.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (дТПToolStripMenuItem.Checked)

{

string searchValue = textBox1.Text;

Найти\_ДТП(searchValue);

}

else if (мероприятияToolStripMenuItem.Checked)

{

string searchValue = textBox1.Text;

Найти\_Мероприятие(searchValue);

}

else if (сотрудникиToolStripMenuItem.Checked)

{

string searchValue = textBox1.Text;

Найти\_Сотрудника(searchValue);

}

}

private void удалитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

if (сотрудникиToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int Сотрудник\_ID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["Сотрудник\_ID"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить этого сотрудника?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_Сотрудника(Сотрудник\_ID);

Отобразить\_Сотрудников();

}

}

else

{

MessageBox.Show("Выберите сотрудника для удаления.");

}

}

else if (дТПToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int ДТП\_ID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ДТП\_ID"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить это ДТП?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_ДТП(ДТП\_ID);

Отобразить\_ДТП();

}

}

else

{

MessageBox.Show("Выберите ДТП для удаления.");

}

}

else if (мероприятияToolStripMenuItem.Checked)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int Мероприятие\_ID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["Мероприятие\_ID"].Value);

DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить это мероприятие?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (result == DialogResult.Yes)

{

Удалить\_Мероприятие(Мероприятие\_ID);

Отобразить\_Мероприятия();

}

}

else

{

MessageBox.Show("Выберите мероприятие для удаления.");

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_Сотрудника();

Отобразить\_Сотрудников();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_Сотрудника();

Отобразить\_Сотрудников();

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_ДТП();

Отобразить\_ДТП();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_ДТП();

Отобразить\_ДТП();

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (добавитьToolStripMenuItem.Checked)

{

Добавить\_Меро();

Отобразить\_Мероприятия();

}

else if (изменитьToolStripMenuItem.Checked)

{

Изменить\_Меро();

Отобразить\_Мероприятия();

}

}

private void Изменить()

{

try

{

if (сотрудникиToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("Сотрудники");

}

else if(дТПToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("ДТП");

}

else if (мероприятияToolStripMenuItem.Checked)

{

UpdatePanelForTableИзменить("Мероприятия");

}

else

{

// Если ни один пункт не выбран, скрываем все панели

UpdatePanelForTableИзменить("");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при реализации: " + ex.Message);

}

}

private void Добавить()

{

try

{

// Определяем, какую таблицу выбрал пользователь и вызываем метод обновления панели

if (сотрудникиToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("Сотрудники");

}

else if (дТПToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("ДТП");

}

else if (мероприятияToolStripMenuItem.Checked)

{

UpdatePanelForTableДобавить("Мероприятия");

}

else

{

// Если ни один пункт не выбран, скрываем все панели

UpdatePanelForTableДобавить("");

}

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при реализации: " + ex.Message);

}

}

private void добавитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Добавить();

if (сотрудникиToolStripMenuItem.Checked)

{

textBox2.Clear();

textBox3.Clear();

maskedTextBox1.Clear();

textBox4.Clear();

}

else if (дТПToolStripMenuItem.Checked)

{

maskedTextBox2.Clear();

textBox5.Clear();

richTextBox1.Text = "";

comboBox1.Text = "";

maskedTextBox3.Clear();

}

else if (мероприятияToolStripMenuItem.Checked)

{

domainUpDown1.Text = "";

maskedTextBox4.Clear();

richTextBox2.Text = "";

domainUpDown2.Text = "";

}

}

private void UpdatePanelForTableИзменить(string tableName)

{

switch (tableName)

{

case "Сотрудники":

panel\_Сотрудники.Visible = true;

label1.Text = "Сотрудник Изменить";

button2.Text = "Изменить";

break;

case "ДТП":

panel\_дтп.Visible = true;

label1.Text = "ДТП Изменить";

button3.Text = "Изменить";

break;

case "Мероприятия":

panel\_МЕРОПРИЯТИЯ.Visible = true;

label1.Text = "Мероприятия Изменить";

button4.Text = "Изменить";

break;

default:

break;

}

}

private void UpdatePanelForTableДобавить(string tableName)

{

switch (tableName)

{

case "Сотрудники":

panel\_Сотрудники.Visible = true;

label1.Text = "Сотрудники Добавить";

button2.Text = "Добавить";

break;

case "ДТП":

panel\_дтп.Visible = true;

label1.Text = "ДТП Добавить";

button3.Text = "Добавить";

break;

case "Мероприятия":

panel\_МЕРОПРИЯТИЯ.Visible = true;

label1.Text = "Мероприятия Добавить";

button4.Text = "Добавить";

break;

default:

break;

}

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (!добавитьToolStripMenuItem.Checked)

{

if (сотрудникиToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

try

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

textBox2.Text = row.Cells["ФИО"].Value.ToString();

textBox3.Text = row.Cells["Должность"].Value.ToString();

maskedTextBox1.Text = row.Cells["Телефон"].Value.ToString();

textBox4.Text = row.Cells["Email"].Value.ToString();

textBox6.Text = row.Cells["Сотрудник\_ID"].Value.ToString();

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при реализации: " + ex.Message);

}

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

else if (дТПToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

// Заполняем поля текущими значениями выбранной строки

DateTime дата;

if (DateTime.TryParse(row.Cells["Дата"].Value.ToString(), out дата))

{

maskedTextBox2.Text = дата.ToString("d"); // Формат даты по желанию; "d" - короткий формат

}

else

{

MessageBox.Show("Некорректный формат даты.");

return;

}

textBox5.Text = row.Cells["Место"].Value.ToString();

richTextBox1.Text = row.Cells["Описание"].Value.ToString();

comboBox1.Text = row.Cells["Степень\_серьезности"].Value.ToString();

maskedTextBox3.Text = row.Cells["Создано"].Value.ToString();

textBox7.Text = row.Cells["ДТП\_ID"].Value.ToString();

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

else if (мероприятияToolStripMenuItem.Checked)

{

try

{

if (e.RowIndex >= 0 && изменитьToolStripMenuItem.Checked == true) // проверяем, что выбрана строка, а не заголовок или пустое пространство

{

DataGridViewRow row = dataGridView1.Rows[e.RowIndex];

domainUpDown1.Text = row.Cells["ДТП\_ID"].Value.ToString();

maskedTextBox4.Text = row.Cells["Дата\_мероприятия"].Value.ToString();

richTextBox2.Text = row.Cells["Описание\_мероприятия"].Value.ToString();

domainUpDown2.Text = row.Cells["Сотрудник\_ID"].Value.ToString();

textBox8.Text = row.Cells["Мероприятие\_ID"].Value.ToString();

}

}

catch (Exception ex)

{

MessageBox.Show("Вы находитесь вне границ таблицы: " + ex.Message);

}

}

}

}

private void изменитьToolStripMenuItem\_Click(object sender, EventArgs e)

{

foreach (ToolStripMenuItem a in действияСТаблицейToolStripMenuItem.DropDownItems)

{

a.Checked = false;

}

((ToolStripMenuItem)sender).Checked = true;

Изменить();

try

{

dataGridView1\_CellClick(dataGridView1, new DataGridViewCellEventArgs(0, dataGridView1.SelectedRows[0].Index));

}

catch

{

}

}

}

}Authorization:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace УП\_Зайцева

{

public partial class Authorization : Form

{

private readonly db DataBase;

public Authorization()

{

InitializeComponent();

DataBase = new db();

}

private void button1\_Click(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

}

private string GetUserRole(string username, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Должность FROM Сотрудники WHERE ФИО = @ФИО";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

object result = command.ExecuteScalar();

if (result != null)

{

return result.ToString();

}

}

}

return "Unknown";

}

private void button2\_Click(object sender, EventArgs e)

{

string username = textBox1.Text;

string password = textBox2.Text;

if (string.IsNullOrWhiteSpace(username) || string.IsNullOrWhiteSpace(password))

{

MessageBox.Show("Введите имя пользователя и пароль.");

return;

}

string connectionString = DataBase.GetConnectionString();

int userid = GetUserID(username, connectionString); // Получаем userid из базы данных

if (AuthenticateUser(username, password, connectionString))

{

string userRole = GetUserRole(username, connectionString); // Получаем роль пользователя

MessageBox.Show("Авторизация успешна!");

this.Hide();

Form1 maimMuni = new Form1(userid); // Передаем userid в конструктор формы MaimMuni

maimMuni.ShowDialog();

}

else

{

MessageBox.Show("Неверное имя пользователя или пароль.");

}

}

private bool AuthenticateUser(string username, string password, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT COUNT(\*) FROM Сотрудники WHERE ФИО = @ФИО AND Телефон = @Пароль";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

command.Parameters.AddWithValue("@Пароль", password);

int count = (int)command.ExecuteScalar();

return count > 0;

}

}

}

private int GetUserID(string username, string connectionString)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string query = "SELECT Сотрудник\_ID FROM Сотрудники WHERE ФИО = @ФИО";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@ФИО", username);

object result = command.ExecuteScalar();

if (result != null)

{

return (int)result;

}

}

}

return -1;

}

}

}

db:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace УП\_Зайцева

{

internal class db

{

private readonly string connectionString;

public db()

{

connectionString = @"Data Source=ADCLG1;Initial Catalog=!!!Зайцева\_УП;Integrated Security=true";

}

public string GetConnectionString()

{

return connectionString;

}

}

}

**Приложение Б**

**UnitTest:**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

using УП\_Зайцева;

namespace UnitTestProject1

{

[TestClass]

public class AuthorizationTests

{

private Authorization \_authorization;

private string \_connectionString = "Data Source=ADCLG1;Initial Catalog=!!!Зайцева\_УП;Integrated Security=true";

[TestInitialize]

public void Setup()

{

\_authorization = new Authorization();

}

[TestMethod]

public void GetUserRole\_ReturnsRole\_WhenUserExists()

{

string username = "Тестовый Пользователь";

string expectedRole = "Командир";

string result = \_authorization.GetUserRole(username, \_connectionString);

Assert.AreEqual(expectedRole, result);

}

[TestMethod]

public void GetUserRole\_ReturnsRole\_WhenUserExists2()

{

string username = "Зайцева Дарья Александровна";

string expectedRole = "Инстпектор";

string result = \_authorization.GetUserRole(username, \_connectionString);

Assert.AreEqual(expectedRole, result);

}

[TestMethod]

public void GetUserRole\_ReturnsUnknown\_WhenUserDoesNotExist()

{

string result = \_authorization.GetUserRole("НеСуществующийПользователь", \_connectionString);

Assert.AreEqual("Unknown", result);

}

[TestMethod]

public void AuthenticateUser\_ReturnsTrue\_WhenValidCredentials()

{

string username = "Тестовый Пользователь";

string password = "2222";

bool result = \_authorization.AuthenticateUser(username, password, \_connectionString);

Assert.IsTrue(result);

}

[TestMethod]

public void AuthenticateUser\_ReturnsTrue\_WhenValidCredentials2()

{

string username = "Кузнецов Николай Николаевич";

string password = "89003456789";

bool result = \_authorization.AuthenticateUser(username, password, \_connectionString);

Assert.IsTrue(result);

}

[TestMethod]

public void AuthenticateUser\_ReturnsFalse\_WhenInvalidCredentials()

{

string username = "invalidUser";

string password = "invalidPassword";

bool result = \_authorization.AuthenticateUser(username, password, \_connectionString);

Assert.IsFalse(result);

}

[TestMethod]

public void AuthenticateUser\_ReturnsFalse\_WhenInvalidCredentials2()

{

string username = "Петров Петр Петрович";

string password = "1111";

bool result = \_authorization.AuthenticateUser(username, password, \_connectionString);

Assert.IsFalse(result);

}

[TestMethod]

public void GetUserID\_ReturnsUserID\_WhenUserExists()

{

string username = "Тестовый Пользователь";

int result = \_authorization.GetUserID(username, \_connectionString);

Assert.IsTrue(result > 0);

}

[TestMethod]

public void GetUserID\_ReturnsMinusOne\_WhenUserDoesNotExist()

{

int result = \_authorization.GetUserID("НеСуществующийПользователь", \_connectionString);

Assert.AreEqual(-1, result);

}

[TestMethod]

public void Button2\_Click\_ShowsMessage\_OnEmptyFields()

{

var textBox1 = new TextBox();

var textBox2 = new TextBox();

\_authorization.Controls.Add(textBox1);

\_authorization.Controls.Add(textBox2);

\_authorization.button2\_Click(null, null);

}

}

}

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/2 группы

Зайцева Дарья Александровна

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «23» сентября 2024 г. по «05» октября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «05» октября 2024 г.