**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

Измайлов Ярослав Денисович

(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

Измайлов Ярослав Денисович

(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. Проектирование UML-диаграмм;
* Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;
* Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 10**

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

(подпись) (Ф.И.О.)

Задание получил «23» сентября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Измайлов Я.Д. (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

Измайлов Ярослав Денисович

(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

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 23.09.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 24.09.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 25.09.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 26.09.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 27.09.2024 | Проектирование интерфейса пользователя. |  |
| 28.09.2024 | Разработка схемы алгоритма программного продукта. |  |
| 30.09.2024 | Разработка модели данных. |  |
| 01.10.2024 | Разработка программного продукта. Создание функциональности ПО. |  |
| 02.10.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 03.10.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 04.10.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. |  |
| 05.10.2024 | Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. Подготовка отчета. |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc178927544)

[Задание № 1 «Разработка технического задания» 7](#_Toc178927545)

[Задание № 2 «Разработка алгоритмов и диаграмм» 10](#_Toc178927546)

[Задание № 3 «Разработка макетов программы» 18](#_Toc178927547)

[Задание № 4 «Разработка программы» 21](#_Toc178927548)

[Задание № 5 «Тестирование» 24](#_Toc178927549)

[Задание № 6 «GitHub. Оценка проекта» 30](#_Toc178927550)

[ЗАКЛЮЧЕНИЕ 32](#_Toc178927551)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 33](#_Toc178927552)

[ПРИЛОЖЕНИЕ А 34](#_Toc178927553)

[ПРИЛОЖЕНИЕ Б 96](#_Toc178927554)

ВВЕДЕНИЕ

За время, отведённое на практику, мне необходимо будет создать программный продукт, который должен полностью удовлетворить требованиям заказчика. Я планирую научиться составлять техническое задание, составлять диаграммы активности, последовательности, вариантов использования, создавать макет интерфейса программы, а также написание программного кода и тестирование программы.

Задание № 1 «Разработка технического задания»

Описание предметной области

Основной целью данного модуля является автоматизация процесса управления авиакомпанией, включая управление рейсами, бронированием билетов, управление персоналом и отчетностью.

Модуль позволит сотрудникам авиакомпании эффективно управлять рейсами, отслеживать статус бронирования билетов, управлять персоналом и получать необходимую отчетность.

Разработка программного модуля для управления авиакомпанией - это процесс создания программного продукта, который позволит авиакомпаниям эффективно управлять своими операциями, минимизировать время простоя и повысить удовлетворенность клиентов.

Основные функции и возможности модуля включают:

1. Управление рейсами:
   1. Добавление/редактирование/удаление рейсов.
   2. Управление статусом рейсов (запланирован, отменен, выполнен.)
   3. Отображение списка рейсов.
2. Бронирование билетов:
   1. Добавление/редактирование/удаление бронирований.
   2. Управление статусом бронирований (зарезервирован, оплачен, отменен).
   3. Отображение списка бронирований.
3. Управление персоналом:
   1. Добавление/редактирование/удаление сотрудников.
   2. Управление ролью сотрудников (администратор, диспетчер, билетный агент).
   3. Отображение списка сотрудников
4. Отчетность:
   1. Генерация отчетов о рейсах, бронированиях и персонале.

Техническое задание

1. Общие сведения

1.1. Наименование проекта: Разработка программного модуля для управления авиакомпанией.

1.2. Заказчик: Авиакомпания "Аэрофлот".

1.3. Исполнитель: Компания "Аэро «ИзДен» Софт».

2. Функциональные требования

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

Номер рейса;

Дата и время вылета;

Дата и время прибытия;

Аэропорт вылета;

Аэропорт прибытия;

Тип самолета;

2.2. Возможность редактирования рейсов:

Изменение даты и времени вылета/прибытия;

Изменение аэропорта вылета/прибытия;

2.3. Возможность бронирования билетов:

Добавление/редактирование/удаление бронирований;

Управление статусом бронирований (зарезервирован, оплачен, отменен);

Отображение списка бронирований

2.4. Возможность управления персоналом:

Добавление/редактирование/удаление сотрудников;

Управление ролью сотрудников (администратор, диспетчер, билетный агент);

Отображение списка сотрудников;

2.5. Авторизация:

Возможность авторизации пользователей;

Ограничение доступа к функциям в зависимости от роли пользователя.

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: SQL Server.

5. Требования к документации

Техническое задание на разработку программного модуля.

6. Руководство по стилю

Цветовая схема:

Цвет фона: белый, светло-серый.

Цвет элементов (белый цвет/синие оттенки).

Цвет текста – белый, серый, черный.

Задание № 2 «Разработка алгоритмов и диаграмм»

В данном задании я создал диаграмму активности (Рисунок 1), диаграмму вариантов использования (Рисунок 2), диаграмму последовательности (Рисунок 3), а также ER-Модель (Рисунок 4).

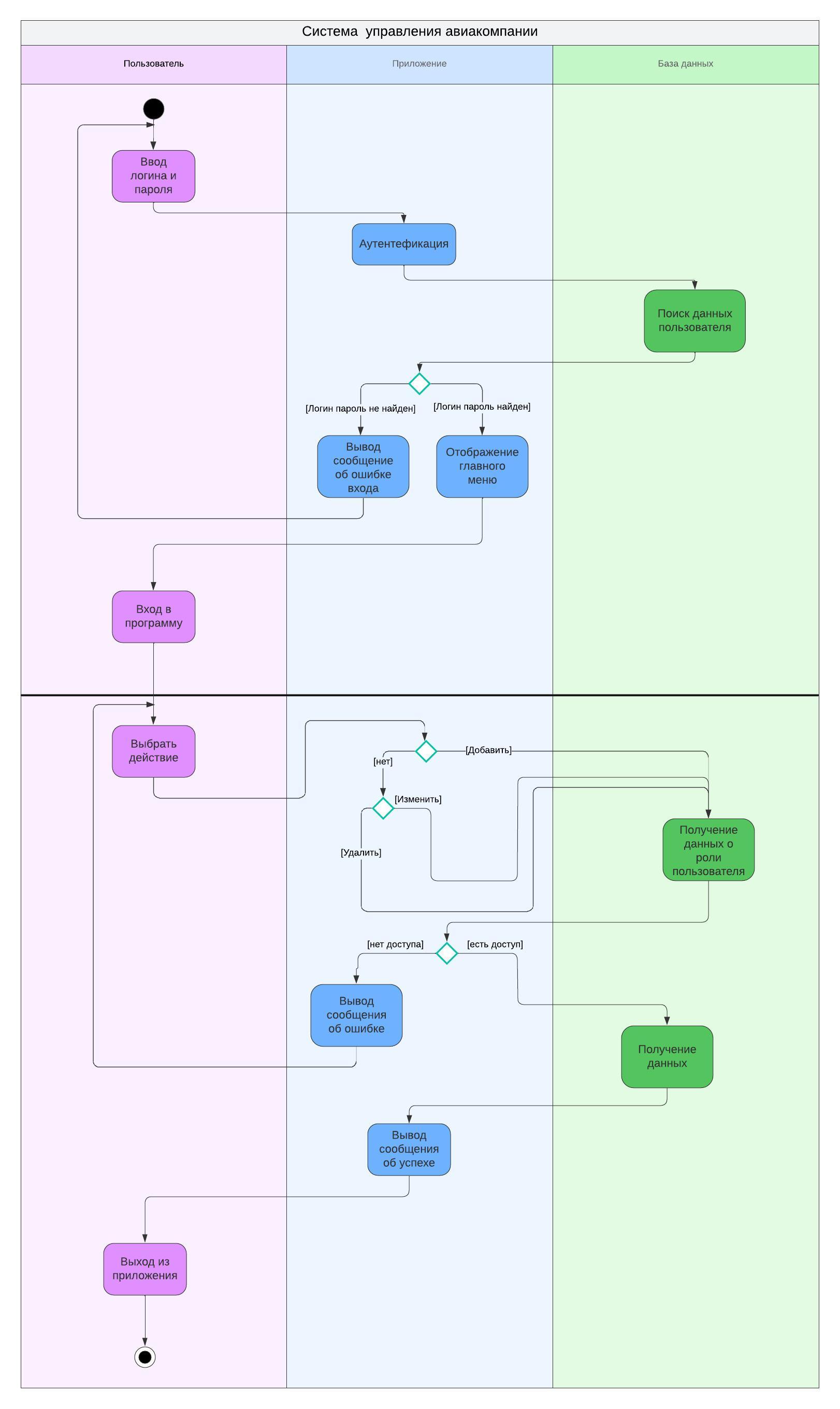


Рисунок 1 – Диаграмма активности

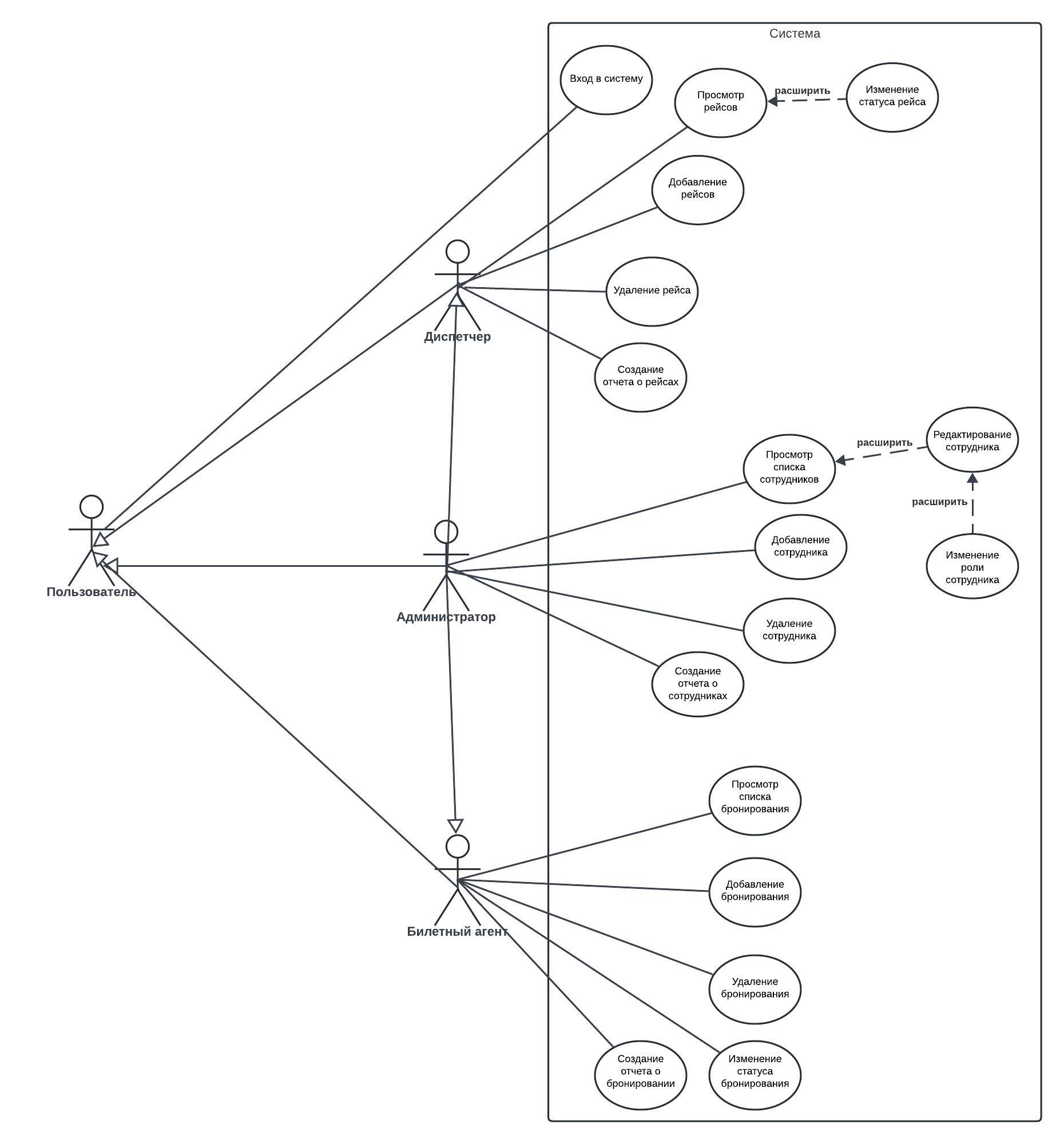


Рисунок 2 – Диаграмма вариантов использования

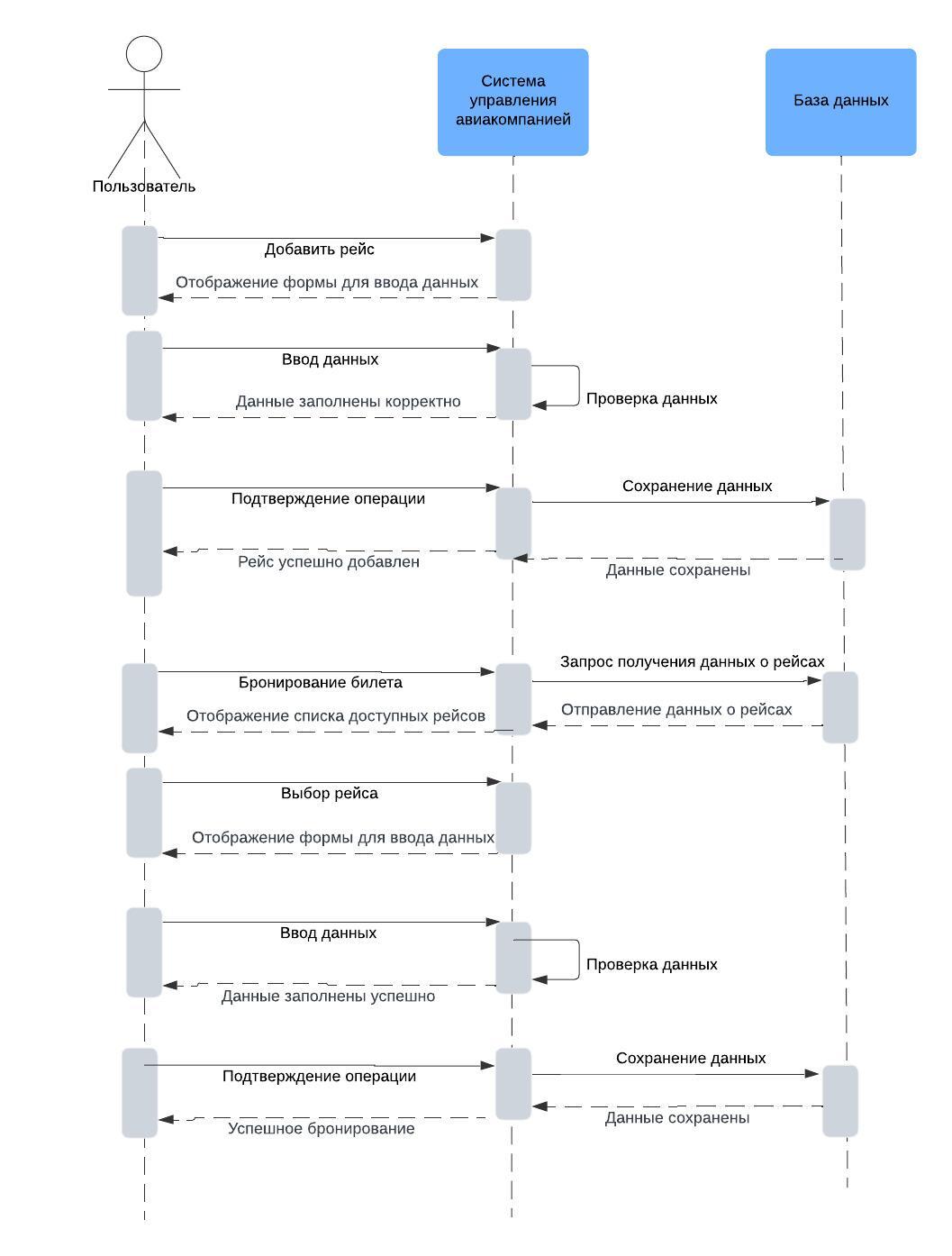


Рисунок 3 – Диаграмма последовательности

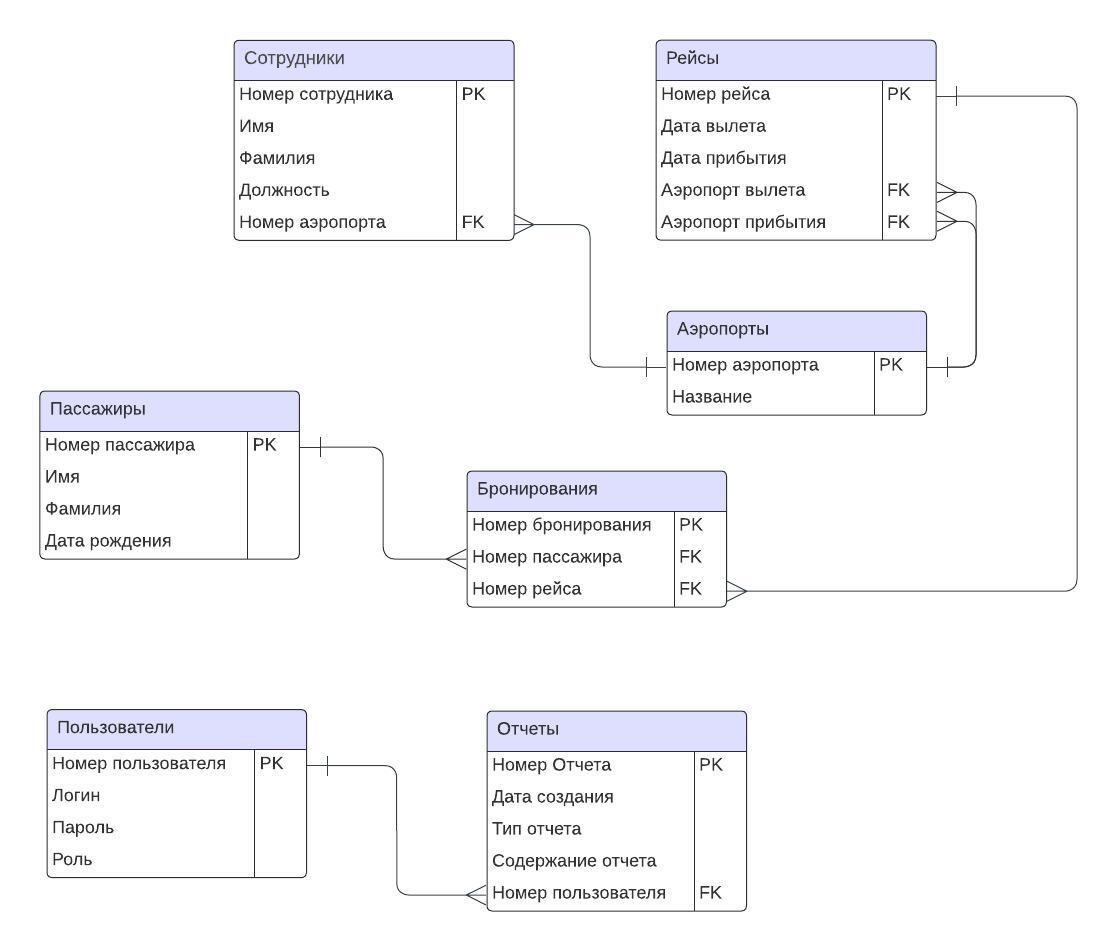
****

Рисунок 4 – ER-Модель

Исходя из ER-модели, я описал словарь данных (Таблица 1), а также вставил в таблицы значения (Рисунок 5 – 11).

Таблица 1 – Словарь данных

| Таблица | Поле | Тип данных | Описание | Ограничения |
| --- | --- | --- | --- | --- |
| Рейсы | Номер рейса | INT | Уникальный идентификатор рейса | Primary Key |
| Дата вылета | DATE | Дата вылета рейса | NOT NULL |
| Дата прибытия | DATE | Дата прибытия рейса | NOT NULL |
| Аэропорт вылета | INT | Идентификатор аэропорта вылета | Foreign Key (Аэропорты) |
| Аэропорт прибытия | INT | Идентификатор аэропорта прибытия | Foreign Key (Аэропорты) |
| Пассажиры | Номер пассажира | INT | Уникальный идентификатор пассажира | Primary Key |
| Имя | VARCHAR(50) | Имя пассажира | NOT NULL |
| Фамилия | VARCHAR(50) | Фамилия пассажира | NOT NULL |
| Дата рождения | DATE | Дата рождения пассажира | NOT NULL |
| Бронирования | Номер бронирования | INT | Уникальный идентификатор бронирования | Primary Key |
| Номер пассажира | INT | Идентификатор пассажира | Foreign Key (Пассажиры) |
| Номер рейса | INT | Идентификатор рейса | Foreign Key (Рейсы) |
| Сотрудники | Номер сотрудника | INT | Уникальный идентификатор сотрудника | Primary Key |
| Имя | VARCHAR(50) | Имя сотрудника | NOT NULL |
| Фамилия | VARCHAR(50) | Фамилия сотрудника | NOT NULL |
| Должность | VARCHAR(50) | Должность сотрудника | NOT NULL |
| Номер аэропорта | INT | Идентификатор аэропорта, в котором работает сотрудник | Foreign Key (Аэропорты) |
| Аэропорты | Номер аэропорта | INT | Уникальный идентификатор аэропорта | Primary Key |
| Название аэропорта | VARCHAR(100) | Название аэропорта | NOT NULL |
| Пользователи | Номер пользователя | INT | Уникальный идентификатор пользователя | Primary Key |
| Логин | VARCHAR(50) | Логин пользователя | NOT NULL |
| Пароль | VARCHAR(50) | Пароль пользователя | NOT NULL |
| Роль | VARCHAR(20) | Роль пользователя | NOT NULL |
| Отчеты | Номер отчета | INT | Уникальный идентификатор отчета | Primary Key |
| Дата создания | DATE | Дата создания отчета | NOT NULL |
| Тип отчета | VARCHAR(50) | Тип отчета | NOT NULL |
| Содержание отчета | TEXT | Содержание отчета | NOT NULL |
| Номер пользователя | INT | Идентификатор пользователя, создавшего отчет | Foreign Key (Пользователи) |

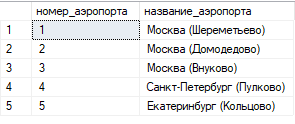


Рисунок 5 – Таблица Аэропорты

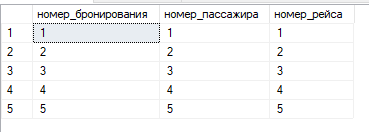


Рисунок 6 – Таблица Бронирования

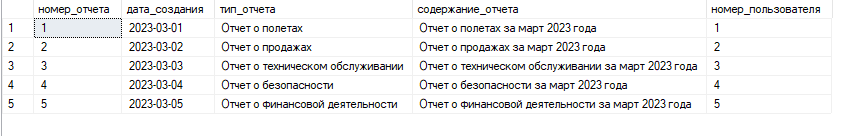


Рисунок 7 – Таблица Отчеты

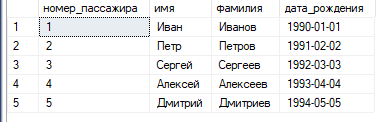


Рисунок 8 – Таблица Пассажиры

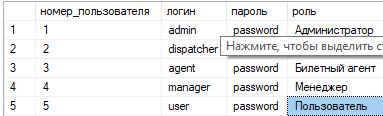


Рисунок 9 – Таблица Пользователи

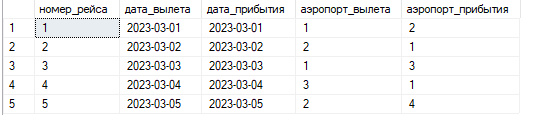


Рисунок 10 – Таблица рейсы

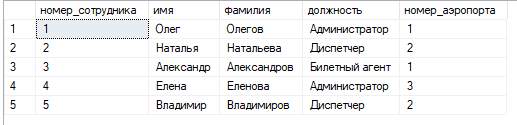


Рисунок 11 – Таблица Сотрудники

Задание № 3 «Разработка макетов программы»

В этом задании я сделал карту навигации будущей программы (Рисунок 12), а также макеты пользовательского интерфейса (Рисунок 13 – 14).



Рисунок 12 – Карта навигации

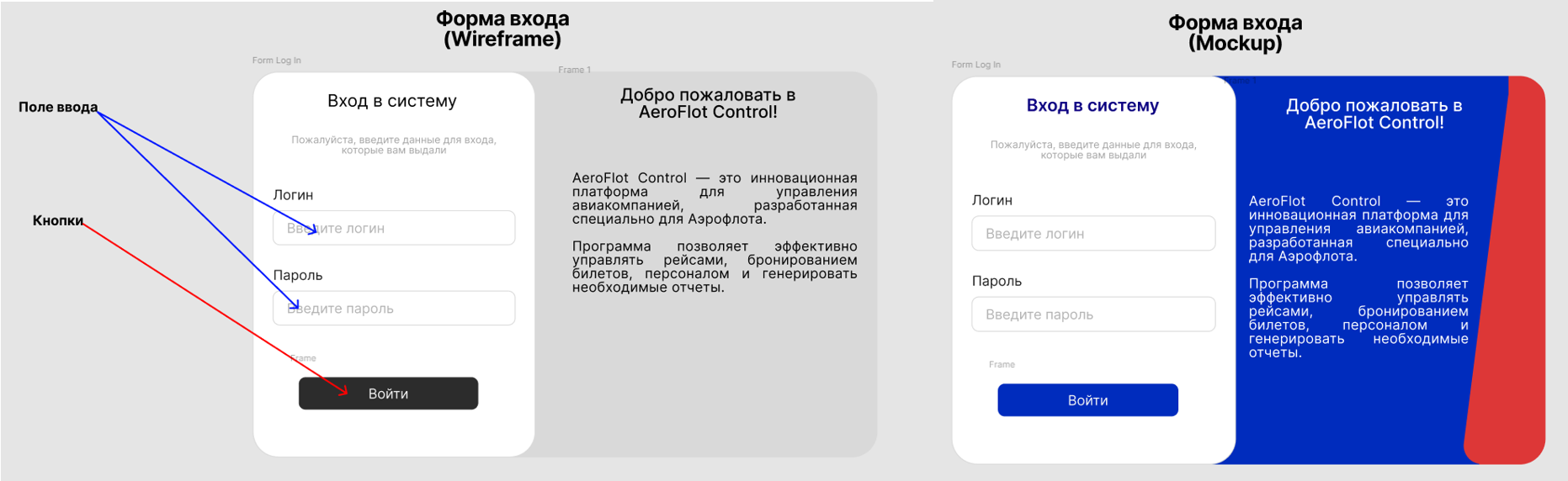


Рисунок 13 – Форма авторизации

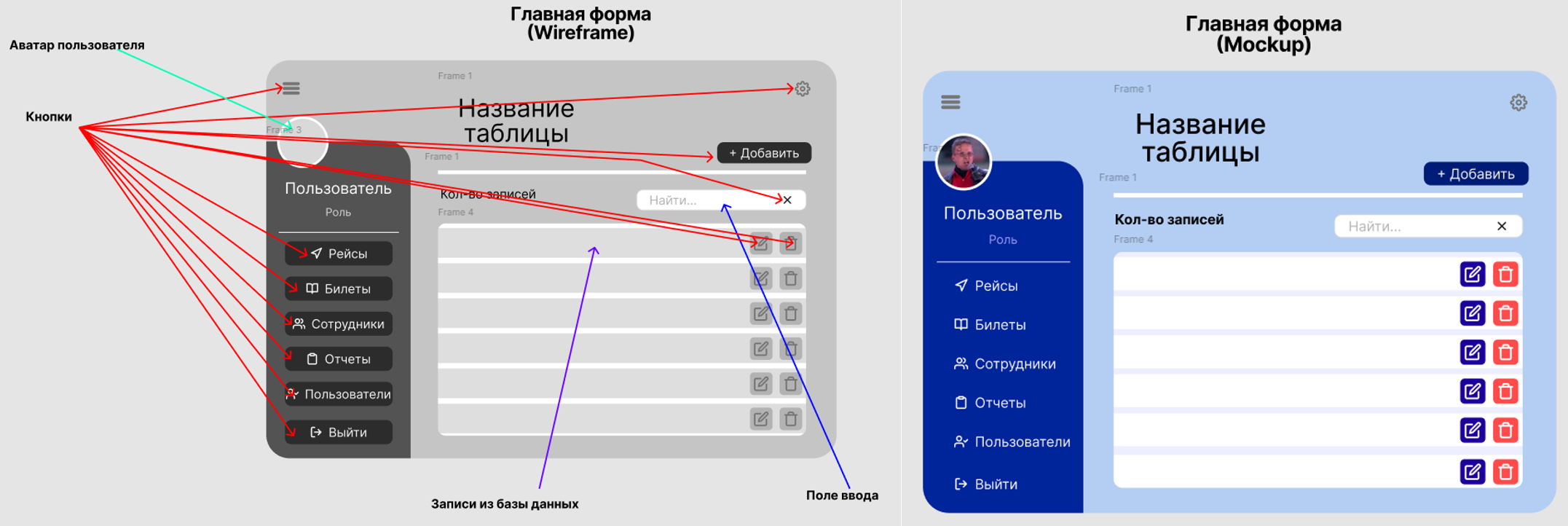


Рисунок 14 – Главная форма

Задание № 4 «Разработка программы»

При входе в программу появляется окно авторизации. Для дальнейшего перехода в главное меню программы необходимо ввести логин и пароль (Рисунок 15).

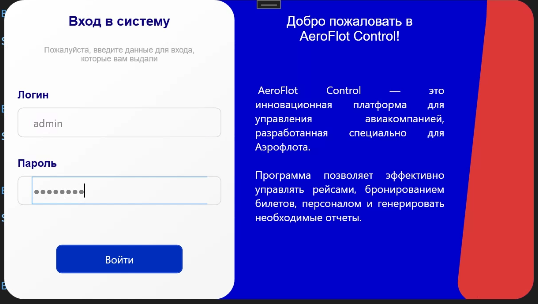


Рисунок 15 – Окно авторизации

После успешной авторизации пользователь попадает в главное меню (Рисунок 16).

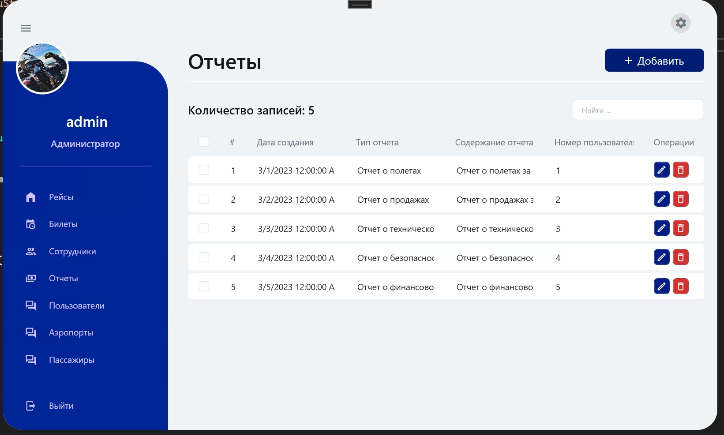


Рисунок 16 – Главное окно

Для открытия любой таблицы и вывода её на экран необходимо в облаcти меню нажать на кнопки соответвующие названиям таблицам. Например откроем таблицу «Билеты» (Рисунок 17).

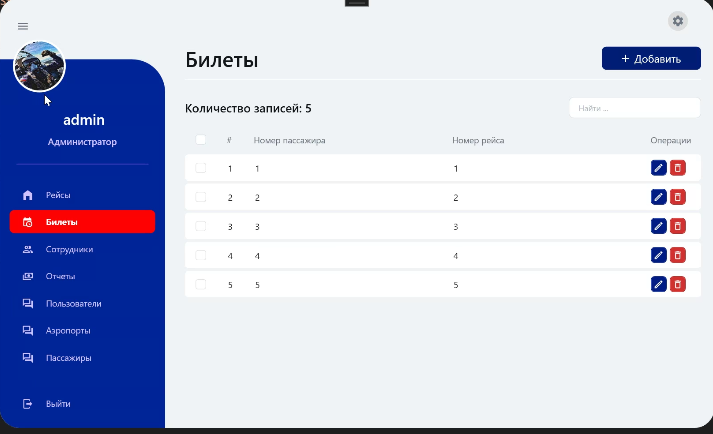


Рисунок 17 – Открытая таблица «Билеты»

После нажатия но кнопку «Добавить» в области над таблицей справа, происходит вызов окна для заполнения новой записи (Рисунок 18).

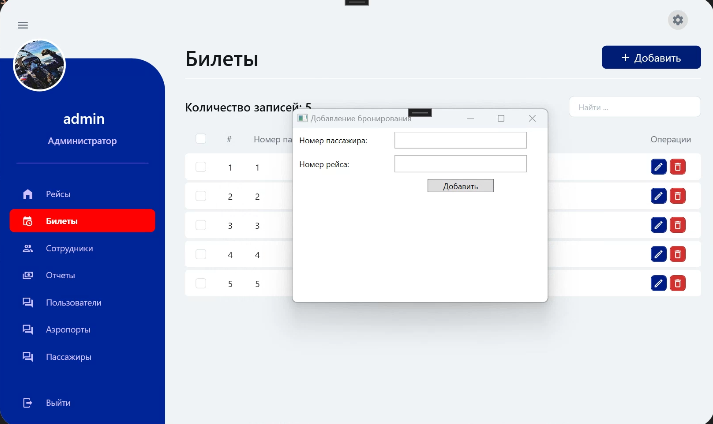


Рисунок 18 – Окно добавления новой записи

При нажатии на кнопку операции редактирования «карандаш на синем фоне», происходит вывод окна с данными выбранной записи для их изменения (Рисунок 19).

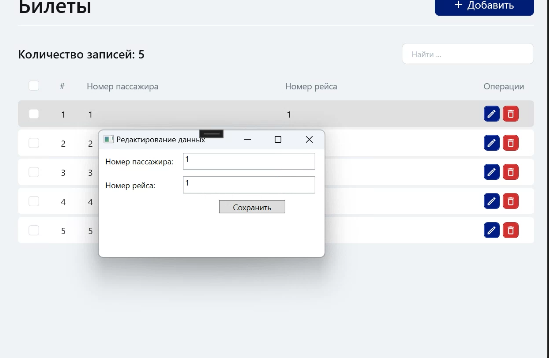


Рисунок 19 – Окно редактирования записи

После мы можем отредактировать любое поле. Нажав кнопку «Сохранить» данные сохраняться в таблице (Рисунок 20 – 21).

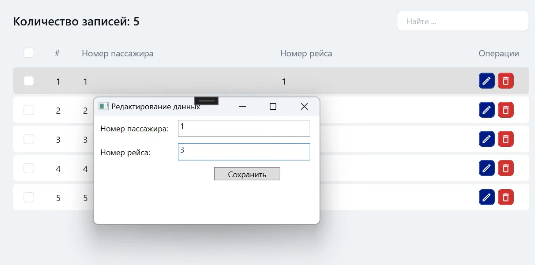


Рисунок 20 – Изменение полей записи

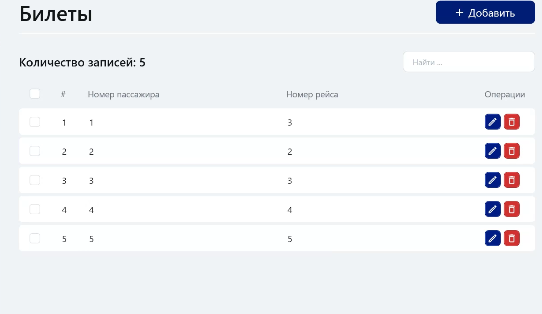


Рисунок 21 – Измененная таблица

При нажатии на кнопку «Добавить», заполним данными поля и добавим их в таблицу (Рисунок 22 – 23).

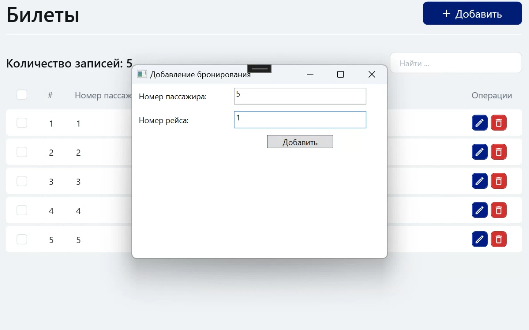


Рисунок 22 – Добавление данных

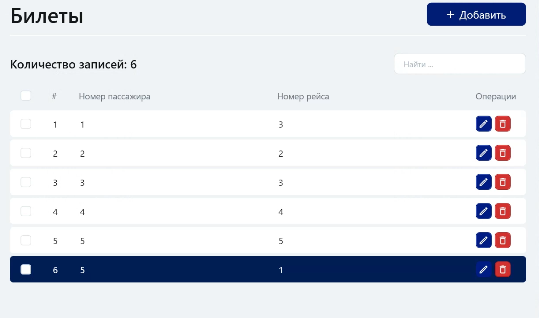


Рисунок 23 – Таблица с новыми данными

При нажатии на кнопку «Удалить» данные удаляться из таблицы, а перед удалением появиться окно с подтверждением операции (Рисунок 24 – 25).

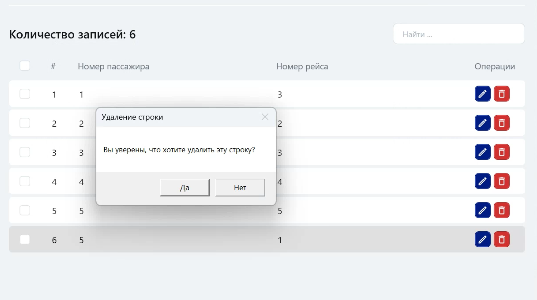


Рисунок 24 – Окно подтверждения

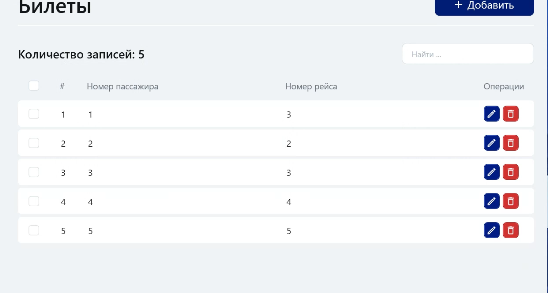


Рисунок 25 – Таблица с удаленной строкой

При нажатии кнопки меню «Выйти» выводится окно с подтверждением действия (Рисунок 26).

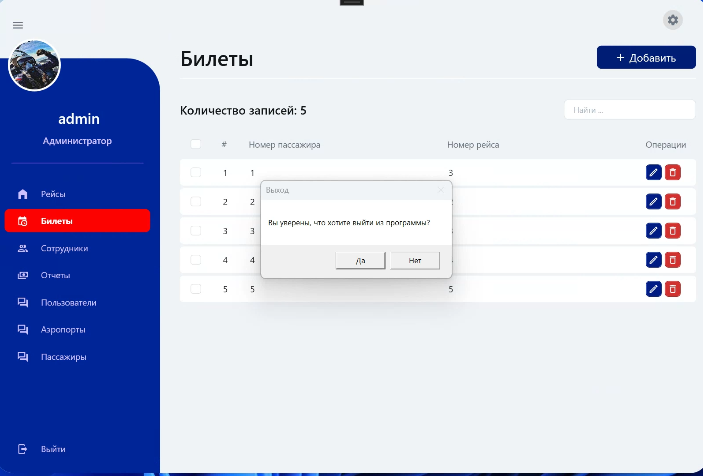


Рисунок 26 – Попытка выхода из программы

Задание № 5 «Тестирование»

В этом задании я составил 5 тест кейсов (Таблица 2 – 6), а также провёл тестирование программы (Рисунок 27). Тестирование прошло успешно.

Таблица 2 – Аннотация теста

|  |  |
| --- | --- |
| Название проекта | Система управления Авиакомпанией |
| Рабочая версия | 1.0 |
| Имя тестирующего | Ярослав |
| Дата(ы) теста | 02.10.2024 |

Таблица 3 – Тест кейс 1

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_1 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Пользователь не может авторизироваться с неправильным логином или/и паролем |
| Краткое изложение теста | Тест проверяет невозможность авторизации пользователя |
| Этапы теста | 1. Написать логин в текстовое поле 2. Написать пароль в текстовое поле 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Если записи не существует, выводится ошибка |
| Тестовые данные | Логин: «admin1», пароль: «123456» |
| Ожидаемый результат | Появляется сообщение об ошибке: «Неправильный логин или пароль» |
| Фактический результат | Появляется сообщение об ошибке: «Неправильный логин или пароль» |
| Статус | Успешно |
| Предварительное условие | Пользователь не авторизован, введён неправильный логин или/и пароль |

Таблица 4 – Тест кейс 2

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_2 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Пользователь авторизовался под администратором |
| Краткое изложение теста | Тест проверяет возможность авторизоваться под администратором |
| Этапы теста | 1. Написать логин админа в текстовое поле 2. Написать пароль админа в текстовое поле 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Успешная авторизация 5. Окно имеет доступ ко всем таблицам |
| Тестовые данные | Логин: «admin», пароль: «password» |
| Ожидаемый результат | Пользователь попадает на главное окно c доступом ко всем таблицам |
| Фактический результат | Пользователь попадает на главную окно c доступом ко всем таблицам |
| Статус | Успешно |
| Предварительное условие | Пользователь ввёл правильный логин и пароль для администратора |

Таблица 5 – Тест кейс 3

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_3 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Пользователь добавил рейс с корректными данными |
| Краткое изложение теста | Тест проверяет возможность добавления рейса с корректными данными |
| Этапы теста | 1. Ввести корректные данные для добавления рейса 2. Нажать кнопку "Добавить" 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Успешное добавление рейса |
| Тестовые данные | Дата вылета: «2022-01-01», дата прибытия: «2022-01-02», аэропорт вылета: «1», аэропорт прибытия: «2» |
| Ожидаемый результат | Рейс добавлен в базу данных |
| Фактический результат | Рейс добавлен в базу данных |
| Статус | Успешно |
| Предварительное условие | Пользователь авторизован, введены корректные данные для добавления рейса |

Таблица 6 – Тест кейс 4

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_4 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Пользователь не может добавить рейс с некорректными данными |
| Краткое изложение теста | Тест проверяет невозможность добавления рейса с некорректными данными |
| Этапы теста | 1. Ввести некорректные данные для добавления рейса 2. Нажать кнопку "Добавить" 3. Проверить, есть ли запись в таблице с таким же логином и паролем 4. Если записи не существует, выводится ошибка |
| Тестовые данные | Дата вылета: «2022-01-01», дата прибытия: «2022-01-01», аэропорт вылета: «1», аэропорт прибытия: «2» |
| Ожидаемый результат | Появляется сообщение об ошибке |
| Фактический результат | Появляется сообщение об ошибке |
| Статус | Успешно |
| Предварительное условие | Пользователь авторизован, введены некорректные данные для добавления рейса |

Таблица 7 – Тест кейс 5

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_5 |
| Приоритет тестирования | Высокий |
| Заголовок/название теста | Пользователь получил список рейсов |
| Краткое изложение теста | Тест проверяет возможность получения списка рейсов |
| Этапы теста | 1. Нажать кнопку "Получить список рейсов" 2. Проверить, есть ли записи в таблице с рейсами 3. Успешное получение списка рейсов |
| Тестовые данные | - |
| Ожидаемый результат | Список рейсов отображается на экране |
| Фактический результат | Список рейсов отображается на экране |
| Статус | Успешно |
| Предварительное условие | Пользователь авторизован |

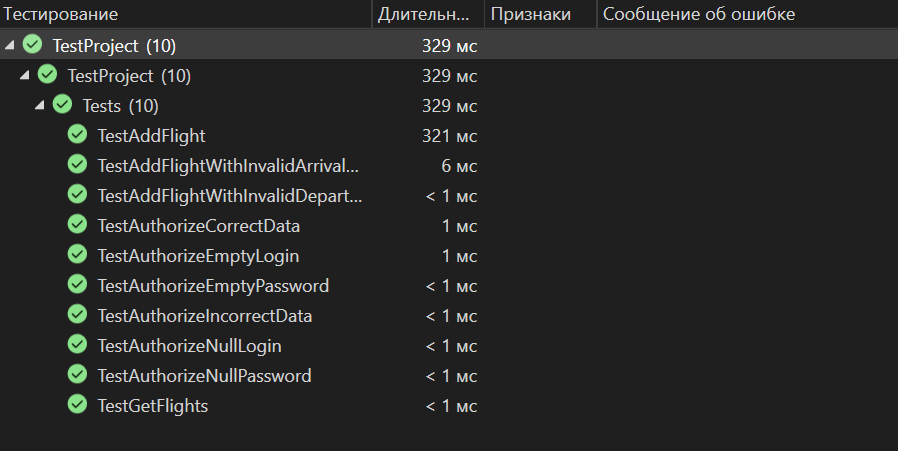


Рисунок 27 – Результаты тестов

Задание № 6 «GitHub. Оценка проекта»

В этом задании я создал новый репозиторий на GitHub и перенес на него мой проект (Рисунок 28 – 29).

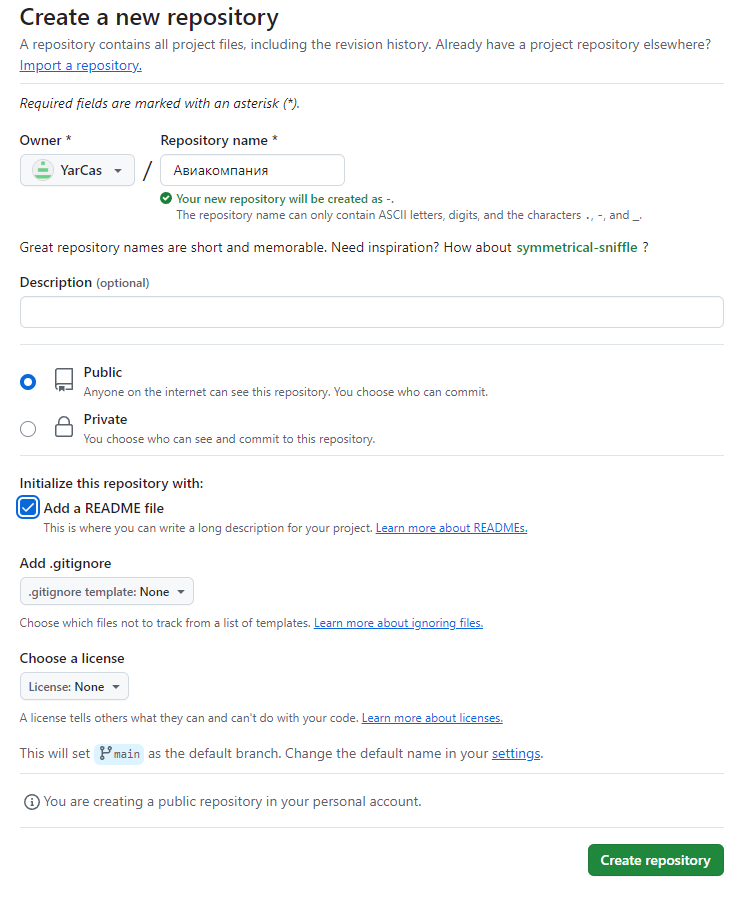


Рисунок 21 – Создание нового репозитория на GitHub

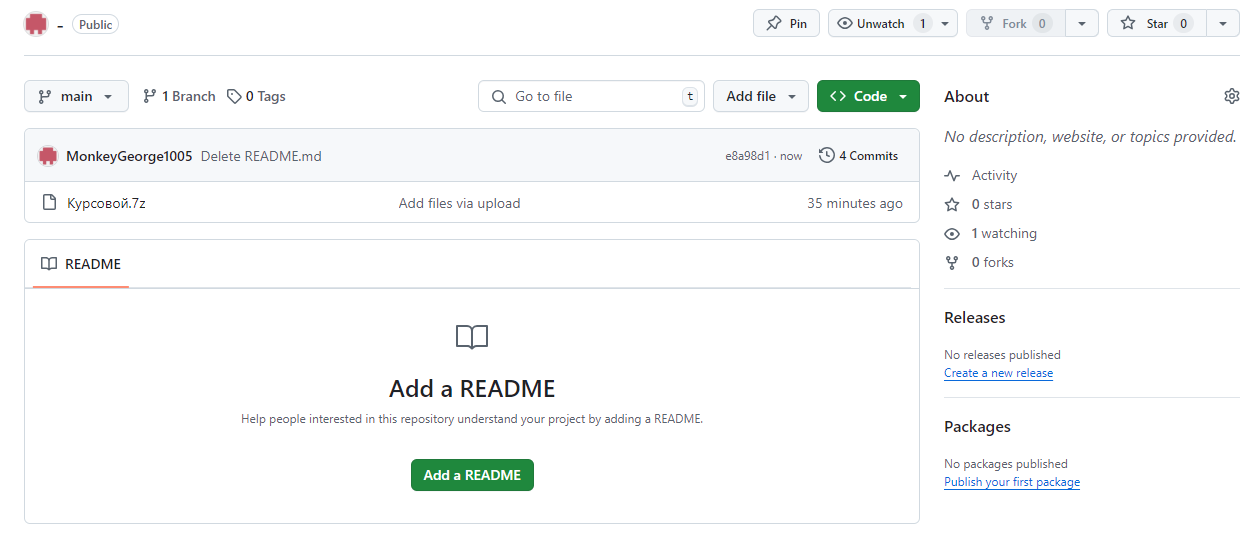


Рисунок 22 – Файл добавлен в репозиторий

ЗАКЛЮЧЕНИЕ

За это время я научился составлять техническое задание, составлять диаграммы активности, последовательности, вариантов использования, создавать макет интерфейса программы, а также написание программного кода и тестирование программы. Заказчик оценивает данный программный продукт на оценку 5. Его ожидания полностью соответствуют результату.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* + - 1. Microsoft Learn – C# Windows Forms [Электронный ресурс]: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control?view=windowsdesktop-8.0>;
      2. Metanit.com – Руководство C# Windows Forms [Электронный ресурс]: <https://metanit.com/sharp/windowsforms/>;
      3. Flexberry Designer Desktop – Руководство Диаграмма активности [Электронный ресурс]: <https://flexberry.github.io/ru/fd_activity-diagram.html>;
      4. Flexberry Designer Desktop – Руководство Диаграмма последовательности [Электронный ресурс]: <https://flexberry.github.io/ru/fd_sequence-diagram.html>;
      5. Flexberry Designer Desktop – Руководство Диаграмма вариантов использования [Электронный ресурс]: <https://flexberry.github.io/ru/fd_use-case-diagram.html>;
      6. Worldskills – Руковоство ER-модель [Электронный ресурс]: <https://nationalteam.worldskills.ru/skills/proektirovanie-er-diagrammy/>

ПРИЛОЖЕНИЕ А

Форма авторизации:

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.KeyPreview = true;

this.KeyUp += new System.Windows.Forms.KeyEventHandler(this.Form1\_KeyUp);

this.button2.Click += new System.EventHandler(this.button2\_Click);

}

public bool Admin;

public bool ADMIN

{

get { return Admin; }

set { value = Admin; }

}

public bool Employee;

public bool EMPLOYEE

{

get { return Employee; }

set { value = Employee; }

}

public int index;

public int INDEX

{

get { return index; }

set { index = value; }

}

private void Form1\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetUsers.Users". При необходимости она может быть перемещена или удалена.

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers.Users);

}

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

{

button1.Enabled = false;

button3.Enabled = false;

button3.Visible = true;

button2.Visible = false;

button1.Visible = false;

button4.Visible = true;

label1.Text = "Регистрация";

textBox1.Clear();

textBox2.Clear();

}

public void Create(string log, string pass)

{

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers.Users);

try

{

log = textBox1.Text;

pass = textBox2.Text;

string role = "клиент";

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDell = "SELECT COUNT(\*) FROM [Users] where [login] = @log";

SqlCommand cmd2 = new SqlCommand(ComDell, MyConnection);

cmd2.Parameters.AddWithValue("@log", log);

MyConnection.Open();

int result = 0;

result = (int)cmd2.ExecuteScalar();

if (result == 0 && log != "")

{

string ComDel = $"Insert into Users([login],[role],[password]) VALUES(@login,@role,@password);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@login", log);

cmd1.Parameters.AddWithValue("@role", role);

cmd1.Parameters.AddWithValue("@password", pass);

cmd1.ExecuteNonQuery();

}

else

{

MessageBox.Show("Такой логин уже занят, выберите другое имя пользователя");

}

MyConnection.Close();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

button1.Enabled = false;

button3.Enabled = false;

button3.Visible = false;

button2.Visible = true;

button1.Visible = true;

button4.Visible = false;

label1.Text = "Авторизация";

textBox1.Clear();

textBox2.Clear();

}

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

{

try

{

string log = textBox1.Text;

string pass = textBox2.Text;

Create(log, pass);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

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

{

button1.Enabled = false;

button3.Enabled = false;

button3.Visible = false;

button2.Visible = true;

button1.Visible = true;

button4.Visible = false;

label1.Text = "Авторизация";

textBox1.Clear();

textBox2.Clear();

}

private void button5\_MouseUp(object sender, MouseEventArgs e)

{

button4.ForeColor = Color.Gray;

textBox2.PasswordChar = '\*';

}

private void button5\_MouseDown(object sender, MouseEventArgs e)

{

button4.ForeColor = Color.White;

textBox2.PasswordChar = '\0';

}

public void Check(string log, string pass)

{

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers.Users);

try

{

log = textBox1.Text;

pass = textBox2.Text;

int j = 0;

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

if (dataGridView1.Rows[i].Cells[2].Value.ToString() == log && dataGridView1.Rows[i].Cells[3].Value.ToString() == pass)

{

j++;

break;

}

}

if (j == 0)

{

throw new Exception("Неправильный логин или пароль!");

}

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = "Select login from Users where login = @log";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@log", log);

string ComDell = "Select password from Users where password = @pass and login = @log";

SqlCommand cmd2 = new SqlCommand(ComDell, MyConnection);

cmd2.Parameters.AddWithValue("@log", log);

cmd2.Parameters.AddWithValue("@pass", pass);

string ComDelll = "Select role from Users where password = @pass and login = @log";

SqlCommand cmd3 = new SqlCommand(ComDelll, MyConnection);

cmd3.Parameters.AddWithValue("@log", log);

cmd3.Parameters.AddWithValue("@pass", pass);

MyConnection.Open();

string a = cmd1.ExecuteScalar().ToString().TrimEnd();

string b = cmd2.ExecuteScalar().ToString().TrimEnd();

string c = cmd3.ExecuteScalar().ToString().TrimEnd();

if (a == log && b == pass)

{

MainForm form = new MainForm(); // создание экземпляра первой формы

if (c == "администратор")

{

Admin = true;

Employee = false;

}

else if (c == "сотрудник")

{

Admin = false;

Employee = true;

}

else

{

Admin = false;

Employee = false;

}

string commandForIndex = "Select id from Users where password = @pass and login = @log";

SqlCommand cmdForIndex = new SqlCommand(commandForIndex, MyConnection);

cmdForIndex.Parameters.AddWithValue("@log", log);

cmdForIndex.Parameters.AddWithValue("@pass", pass);

string id = cmdForIndex.ExecuteScalar().ToString().TrimEnd();

INDEX = Convert.ToInt32(id);

form.Owner = this;

form.ShowDialog();

this.Visible = false;

this.Enabled = false;

}

MyConnection.Close();

}

catch (Exception ex)

{

throw new Exception(ex.Message);

}

textBox1.Clear();

textBox2.Clear();

}

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

{

string log = textBox1.Text;

string pass = textBox2.Text;

try

{

Check(log, pass);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void Form1\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter && button1.Visible == true && button3.Visible == false)

button1.PerformClick();

else if (e.KeyCode == Keys.Enter && button3.Visible == true && button1.Visible == false)

button3.PerformClick();

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

Application.Exit();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text != string.Empty && textBox2.Text != string.Empty)

{

button1.Enabled = true;

button3.Enabled = true;

}

else

{

button1.Enabled = false;

button3.Enabled = false;

}

}

}

Главная форма:

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

Form1 form1;

public bool a = false;

public bool em = false;

private void MainForm\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetPurchases.Purchases". При необходимости она может быть перемещена или удалена.

this.purchasesTableAdapter.Fill(this.практика\_КудаевDataSetPurchases.Purchases);

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetProducts.Products". При необходимости она может быть перемещена или удалена.

this.productsTableAdapter.Fill(this.практика\_КудаевDataSetProducts.Products);

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetUsers1.Users". При необходимости она может быть перемещена или удалена.

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers1.Users);

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetWarehouse\_Stocks.Warehouse\_Stocks". При необходимости она может быть перемещена или удалена.

this.warehouse\_StocksTableAdapter.Fill(this.практика\_КудаевDataSetWarehouse\_Stocks.Warehouse\_Stocks);

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetProduction\_Operations.Production\_Operations". При необходимости она может быть перемещена или удалена.

this.production\_OperationsTableAdapter.Fill(this.практика\_КудаевDataSetProduction\_Operations.Production\_Operations);

// TODO: данная строка кода позволяет загрузить данные в таблицу "практика\_КудаевDataSetOrders.Orders". При необходимости она может быть перемещена или удалена.

this.ordersTableAdapter.Fill(this.практика\_КудаевDataSetOrders.Orders);

form1 = (Form1)this.Owner;

form1.Hide();

a = form1.ADMIN;

em = form1.EMPLOYEE;

if (a)

{

panel1.Visible = true;

panel2.Visible = false;

addOrders.Visible = false;

}

else if (em)

{

tabControl1.TabPages.Remove(tabPage4);

tabControl1.TabPages.Remove(tabPage6);

panel1.Visible = true;

panel2.Visible = false;

addOrders.Visible = false;

SettingsColor.Visible = false;

SettingsStyle.Visible = false;

SettingsStyleButtons.Visible = false;

}

else

{

dataGridView1.Visible = false;

editOrders.Visible = false;

deleteOrders.Visible = false;

panel1.Visible = false;

groupBox1.Visible = false;

groupBox2.Visible = false;

panel2.Visible = true;

tabPage1.Size = new Size(200, 200);

tabControl1.Size = new Size(200, 200);

this.Size = new Size(320, 320);

SettingsColor.Visible = false;

SettingsStyle.Visible = false;

SettingsStyleButtons.Visible = false;

SignOut.Location = new Point(220, 64);

tabControl1.TabPages.Remove(tabPage2);

tabControl1.TabPages.Remove(tabPage3);

tabControl1.TabPages.Remove(tabPage4);

tabControl1.TabPages.Remove(tabPage6);

panel6.Visible = false;

addProducts.Visible = false;

editProducts.Visible = false;

deleteProducts.Visible = false;

}

}

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

{

this.Close();

a = false;

em = false;

this.Hide();

Form1 form = new Form1();

form.ShowDialog();

this.Show();

}

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)

{

form1 = null;

Application.Exit();

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (!a && !em)

{

TabPage selectedTabPage = tabControl1.SelectedTab;

switch (selectedTabPage.Name)

{

case "tabPage1":

tabPage1.Size = new Size(200, 200);

tabControl1.Size = new Size(200, 200);

this.Size = new Size(320, 320);

SettingsColor.Visible = false;

SignOut.Location = new Point(220, 64);

break;

case "tabPage5":

tabPage5.Size = new Size(768, 461);

tabControl1.Size = new Size(776, 487);

this.Size = new Size(904, 550);

SignOut.Location = new Point(801, 34);

break;

}

}

}

private void product\_client\_id\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != (int)Keys.Back)

e.Handled = true;

}

private void addOrders\_Click(object sender, EventArgs e) //ПЕРВАЯ ТАБЛИЦА

{

string productID = product\_client\_id.Text;

string quantity = quantity\_client.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Insert into Orders([client\_id], [product\_id], [quantity], [status]) VALUES(@id, @product\_id, @quan, @status);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", form1.INDEX);

cmd1.Parameters.AddWithValue("@product\_id", productID);

cmd1.Parameters.AddWithValue("@quan", quantity);

cmd1.Parameters.AddWithValue("@status", "новый");

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.ordersTableAdapter.Fill(this.практика\_КудаевDataSetOrders.Orders);

order\_id.Clear();

order\_client\_id.Clear();

order\_product\_id.Clear();

order\_quantity.Clear();

order\_status.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

if (dataGridView1.Rows[i].Cells[0].Value.ToString() == order\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Заказ не найден!");

string orderId = order\_id.Text;

string clientID = order\_client\_id.Text;

string productID = order\_product\_id.Text;

string quantity = order\_client\_id.Text;

string status = order\_status.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Update Orders set [client\_id] = @client\_id, [product\_id] = @product\_id, [quantity] = @quan, [status] = @status where [id] = @id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", orderId);

cmd1.Parameters.AddWithValue("@client\_id", clientID);

cmd1.Parameters.AddWithValue("@product\_id", productID);

cmd1.Parameters.AddWithValue("@quan", quantity);

cmd1.Parameters.AddWithValue("@status", status);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.ordersTableAdapter.Fill(this.практика\_КудаевDataSetOrders.Orders);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

order\_id.Clear();

order\_client\_id.Clear();

order\_product\_id.Clear();

order\_quantity.Clear();

order\_status.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

if (dataGridView1.Rows[i].Cells[0].Value.ToString() == order\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Игрок не найден!");

string id = order\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = "Delete from Orders where [id] = @id";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.ordersTableAdapter.Fill(this.практика\_КудаевDataSetOrders.Orders);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

order\_id.Clear();

order\_client\_id.Clear();

order\_product\_id.Clear();

order\_quantity.Clear();

order\_status.Clear();

}

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

{

dataGridView1.ClearSelection();

string searchValue = find\_id\_orders.Text;

dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

try

{

int j = 0;

for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)

{

if (dataGridView1.Rows[i].Cells[0].Value.ToString() == find\_id\_orders.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Заказ не найден!");

foreach (DataGridViewRow row in dataGridView1.Rows)

{

if (row.Cells[0].Value.ToString().TrimEnd() == searchValue)

{

row.Selected = true;

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

dataGridView1.Refresh();

find\_id\_orders.Clear();

findOrders.Enabled = false;

}

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

{

if (descOrders.Checked)

{

switch (comboBox1.Text)

{

case "id":

dataGridView1.Sort(dataGridView1.Columns[0], ListSortDirection.Descending);

break;

case "id клиента":

dataGridView1.Sort(dataGridView1.Columns[1], ListSortDirection.Descending);

break;

case "id продукта":

dataGridView1.Sort(dataGridView1.Columns[2], ListSortDirection.Descending);

break;

case "Количество":

dataGridView1.Sort(dataGridView1.Columns[3], ListSortDirection.Descending);

break;

case "Статус":

dataGridView1.Sort(dataGridView1.Columns[4], ListSortDirection.Descending);

break;

}

}

else if (ascOrders.Checked)

{

switch (comboBox1.Text)

{

case "id":

dataGridView1.Sort(dataGridView1.Columns[0], ListSortDirection.Ascending);

break;

case "id клиента":

dataGridView1.Sort(dataGridView1.Columns[1], ListSortDirection.Ascending);

break;

case "id продукта":

dataGridView1.Sort(dataGridView1.Columns[2], ListSortDirection.Ascending);

break;

case "Количество":

dataGridView1.Sort(dataGridView1.Columns[3], ListSortDirection.Ascending);

break;

case "Статус":

dataGridView1.Sort(dataGridView1.Columns[4], ListSortDirection.Ascending);

break;

}

}

}

private void descOrders\_CheckedChanged(object sender, EventArgs e)

{

if (descOrders.Checked)

ascOrders.Checked = false;

else if (ascOrders.Checked)

descOrders.Checked = false;

}

private void find\_id\_orders\_TextChanged(object sender, EventArgs e)

{

if (find\_id\_orders.Text == "")

findOrders.Enabled = false;

else findOrders.Enabled = true;

}

private void addOperations\_Click(object sender, EventArgs e) //ВТОРАЯ ТАБЛИЦА

{

string orderId = operation\_order\_id.Text;

string productId = operation\_product\_id.Text;

string type = operation\_type.Text;

DateTime endDate = dateTimePicker1.Value;

string emp = operation\_resp.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Insert into Production\_Operations([order\_id], [product\_id], [operation\_type], [end\_date], [responsible\_employee\_id]) VALUES(@id, @product\_id, @type, @date, @emp);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", orderId);

cmd1.Parameters.AddWithValue("@product\_id", productId);

cmd1.Parameters.AddWithValue("@type", type);

cmd1.Parameters.AddWithValue("@date", endDate);

cmd1.Parameters.AddWithValue("@emp", emp);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.production\_OperationsTableAdapter.Fill(this.практика\_КудаевDataSetProduction\_Operations.Production\_Operations);

operation\_id.Clear();

operation\_order\_id.Clear();

operation\_product\_id.Clear();

operation\_type.Clear();

operation\_resp.Clear();

dateTimePicker1.Value = DateTime.Now;

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView2.Rows.Count - 1; i++)

{

if (dataGridView2.Rows[i].Cells[0].Value.ToString() == operation\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Производственная операция не найдена!");

string id = operation\_id.Text;

string orderId = operation\_order\_id.Text;

string productId = operation\_product\_id.Text;

string type = operation\_type.Text;

DateTime endDate = dateTimePicker1.Value;

string emp = operation\_resp.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Update Production\_Operations set [order\_id] = @order\_id, [product\_id] = @product\_id, [operation\_type] = @type, [end\_date] = @date, [responsible\_employee\_id] = @emp where [id] = @id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

cmd1.Parameters.AddWithValue("@order\_id", orderId);

cmd1.Parameters.AddWithValue("@product\_id", productId);

cmd1.Parameters.AddWithValue("@type", type);

cmd1.Parameters.AddWithValue("@date", endDate);

cmd1.Parameters.AddWithValue("@emp", emp);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.production\_OperationsTableAdapter.Fill(this.практика\_КудаевDataSetProduction\_Operations.Production\_Operations);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

operation\_id.Clear();

operation\_order\_id.Clear();

operation\_product\_id.Clear();

operation\_type.Clear();

operation\_resp.Clear();

dateTimePicker1.Value = DateTime.Now;

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView2.Rows.Count - 1; i++)

{

if (dataGridView2.Rows[i].Cells[0].Value.ToString() == operation\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Производственная операция не найдена!");

string id = operation\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = "Delete from Production\_Operations where [id] = @id";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.production\_OperationsTableAdapter.Fill(this.практика\_КудаевDataSetProduction\_Operations.Production\_Operations);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

operation\_id.Clear();

operation\_order\_id.Clear();

operation\_product\_id.Clear();

operation\_type.Clear();

operation\_resp.Clear();

dateTimePicker1.Value = DateTime.Now;

}

private void descOperations\_CheckedChanged(object sender, EventArgs e)

{

if (descOperations.Checked)

ascOperations.Checked = false;

else if (descOperations.Checked)

ascOperations.Checked = false;

}

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

{

if (descOperations.Checked)

{

switch (comboBox2.Text)

{

case "id":

dataGridView2.Sort(dataGridView2.Columns[0], ListSortDirection.Descending);

break;

case "id заказа":

dataGridView2.Sort(dataGridView2.Columns[1], ListSortDirection.Descending);

break;

case "id продукта":

dataGridView2.Sort(dataGridView2.Columns[2], ListSortDirection.Descending);

break;

case "Тип операции":

dataGridView2.Sort(dataGridView2.Columns[3], ListSortDirection.Descending);

break;

case "Дата окончания":

dataGridView2.Sort(dataGridView2.Columns[4], ListSortDirection.Descending);

break;

case "Ответственный сотрудник":

dataGridView2.Sort(dataGridView2.Columns[5], ListSortDirection.Descending);

break;

}

}

else if (ascOperations.Checked)

{

switch (comboBox2.Text)

{

case "id":

dataGridView2.Sort(dataGridView2.Columns[0], ListSortDirection.Ascending);

break;

case "id заказа":

dataGridView2.Sort(dataGridView2.Columns[1], ListSortDirection.Ascending);

break;

case "id продукта":

dataGridView2.Sort(dataGridView2.Columns[2], ListSortDirection.Ascending);

break;

case "Тип операции":

dataGridView2.Sort(dataGridView2.Columns[3], ListSortDirection.Ascending);

break;

case "Дата окончания":

dataGridView2.Sort(dataGridView2.Columns[4], ListSortDirection.Ascending);

break;

case "Ответственный сотрудник":

dataGridView2.Sort(dataGridView2.Columns[5], ListSortDirection.Ascending);

break;

}

}

}

private void find\_id\_operations\_TextChanged(object sender, EventArgs e)

{

if (find\_id\_operations.Text == "")

findOperations.Enabled = false;

else findOperations.Enabled = true;

}

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

{

dataGridView2.ClearSelection();

string searchValue = find\_id\_operations.Text;

dataGridView2.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

try

{

int j = 0;

for (int i = 0; i < dataGridView2.Rows.Count - 1; i++)

{

if (dataGridView2.Rows[i].Cells[0].Value.ToString() == find\_id\_operations.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Производственная операция не найдена!");

foreach (DataGridViewRow row in dataGridView2.Rows)

{

if (row.Cells[0].Value.ToString().TrimEnd() == searchValue)

{

row.Selected = true;

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

dataGridView2.Refresh();

find\_id\_operations.Clear();

findOperations.Enabled = false;

}

private void addStocks\_Click(object sender, EventArgs e) //ТРЕТЬЯ ТАБЛИЦА

{

string productId = stocks\_product\_id.Text;

string quan = stocks\_quantity.Text;

string loc = stocks\_location.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Insert into Warehouse\_Stocks([product\_id], [quantity], [location]) VALUES(@product\_id, @quan, @loc);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@product\_id", productId);

cmd1.Parameters.AddWithValue("@quan", quan);

cmd1.Parameters.AddWithValue("@loc", loc);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.warehouse\_StocksTableAdapter.Fill(this.практика\_КудаевDataSetWarehouse\_Stocks.Warehouse\_Stocks);

stocks\_id.Clear();

stocks\_product\_id.Clear();

stocks\_quantity.Clear();

stocks\_location.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView3.Rows.Count - 1; i++)

{

if (dataGridView3.Rows[i].Cells[0].Value.ToString() == stocks\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Склад не найден!");

string id = stocks\_id.Text;

string productId = stocks\_product\_id.Text;

string quan = stocks\_quantity.Text;

string loc = stocks\_location.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Update Warehouse\_Stocks set [product\_id] = @product\_id, [quantity] = @quan, [location] = @loc where [id] = @id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

cmd1.Parameters.AddWithValue("@product\_id", productId);

cmd1.Parameters.AddWithValue("@quan", quan);

cmd1.Parameters.AddWithValue("@loc", loc);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.warehouse\_StocksTableAdapter.Fill(this.практика\_КудаевDataSetWarehouse\_Stocks.Warehouse\_Stocks);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

stocks\_id.Clear();

stocks\_product\_id.Clear();

stocks\_quantity.Clear();

stocks\_location.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView3.Rows.Count - 1; i++)

{

if (dataGridView3.Rows[i].Cells[0].Value.ToString() == stocks\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Склад не найден!");

string id = stocks\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = "Delete from Warehouse\_Stocks where [id] = @id";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.warehouse\_StocksTableAdapter.Fill(this.практика\_КудаевDataSetWarehouse\_Stocks.Warehouse\_Stocks);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

stocks\_id.Clear();

stocks\_product\_id.Clear();

stocks\_quantity.Clear();

stocks\_location.Clear();

}

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

{

if (descStocks.Checked)

{

switch (comboBox3.Text)

{

case "id":

dataGridView3.Sort(dataGridView3.Columns[0], ListSortDirection.Descending);

break;

case "id продукта":

dataGridView3.Sort(dataGridView3.Columns[1], ListSortDirection.Descending);

break;

case "Количество":

dataGridView3.Sort(dataGridView3.Columns[2], ListSortDirection.Descending);

break;

case "Адрес":

dataGridView3.Sort(dataGridView3.Columns[3], ListSortDirection.Descending);

break;

}

}

else if (ascStocks.Checked)

{

switch (comboBox3.Text)

{

case "id":

dataGridView3.Sort(dataGridView3.Columns[0], ListSortDirection.Ascending);

break;

case "id продукта":

dataGridView3.Sort(dataGridView3.Columns[1], ListSortDirection.Ascending);

break;

case "Количество":

dataGridView3.Sort(dataGridView3.Columns[2], ListSortDirection.Ascending);

break;

case "Адрес":

dataGridView3.Sort(dataGridView3.Columns[3], ListSortDirection.Ascending);

break;

}

}

}

private void descStocks\_CheckedChanged(object sender, EventArgs e)

{

if (descStocks.Checked)

ascStocks.Checked = false;

else if (descStocks.Checked)

ascStocks.Checked = false;

}

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

{

dataGridView3.ClearSelection();

string searchValue = find\_id\_stocks.Text;

dataGridView3.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

try

{

int j = 0;

for (int i = 0; i < dataGridView3.Rows.Count - 1; i++)

{

if (dataGridView3.Rows[i].Cells[0].Value.ToString() == find\_id\_stocks.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Склад не найден!");

foreach (DataGridViewRow row in dataGridView3.Rows)

{

if (row.Cells[0].Value.ToString().TrimEnd() == searchValue)

{

row.Selected = true;

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

dataGridView3.Refresh();

find\_id\_stocks.Clear();

findStocks.Enabled = false;

}

private void find\_id\_stocks\_TextChanged(object sender, EventArgs e)

{

if (find\_id\_stocks.Text == "")

findStocks.Enabled = false;

else findStocks.Enabled = true;

}

private void addUsers\_Click(object sender, EventArgs e) //ЧЕТВЁРТАЯ ТАБЛИЦА

{

string role = user\_role.Text;

string log = user\_login.Text;

string pass = user\_password.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Insert into Users([role], [login], [password]) VALUES(@role, @log, @pass);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@role", role);

cmd1.Parameters.AddWithValue("@log", log);

cmd1.Parameters.AddWithValue("@pass", pass);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers1.Users);

user\_id.Clear();

user\_role.Text = "";

user\_login.Clear();

user\_password.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView4.Rows.Count - 1; i++)

{

if (dataGridView4.Rows[i].Cells[0].Value.ToString() == user\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Пользователь не найден!");

string id = user\_id.Text;

string role = user\_role.Text;

string log = user\_login.Text;

string pass = user\_password.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Update Users set [role] = @role, [login] = @log, [password] = @pass where [id] = @id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

cmd1.Parameters.AddWithValue("@role", role);

cmd1.Parameters.AddWithValue("@log", log);

cmd1.Parameters.AddWithValue("@pass", pass);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers1.Users);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

user\_id.Clear();

user\_role.Text = "";

user\_login.Clear();

user\_password.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView4.Rows.Count - 1; i++)

{

if (dataGridView4.Rows[i].Cells[0].Value.ToString() == user\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Пользователь не найден!");

string id = user\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = "Delete from Users where [id] = @id";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.usersTableAdapter.Fill(this.практика\_КудаевDataSetUsers1.Users);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

user\_id.Clear();

user\_role.Text = "";

user\_login.Clear();

user\_password.Clear();

}

private void descUsers\_CheckedChanged(object sender, EventArgs e)

{

if (descUsers.Checked)

ascUsers.Checked = false;

else if (ascUsers.Checked)

descUsers.Checked = false;

}

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

{

if (descUsers.Checked)

{

switch (comboBox4.Text)

{

case "id":

dataGridView4.Sort(dataGridView4.Columns[0], ListSortDirection.Descending);

break;

case "Роль":

dataGridView4.Sort(dataGridView4.Columns[1], ListSortDirection.Descending);

break;

case "Логин":

dataGridView4.Sort(dataGridView4.Columns[2], ListSortDirection.Descending);

break;

case "Пароль":

dataGridView4.Sort(dataGridView4.Columns[3], ListSortDirection.Descending);

break;

}

}

else if (ascUsers.Checked)

{

switch (comboBox4.Text)

{

case "id":

dataGridView4.Sort(dataGridView4.Columns[0], ListSortDirection.Ascending);

break;

case "Роль":

dataGridView4.Sort(dataGridView4.Columns[1], ListSortDirection.Ascending);

break;

case "Логин":

dataGridView4.Sort(dataGridView4.Columns[2], ListSortDirection.Ascending);

break;

case "Пароль":

dataGridView4.Sort(dataGridView4.Columns[3], ListSortDirection.Ascending);

break;

}

}

}

private void find\_id\_users\_TextChanged(object sender, EventArgs e)

{

if (find\_id\_users.Text == "")

findUsers.Enabled = false;

else findUsers.Enabled = true;

}

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

{

dataGridView4.ClearSelection();

string searchValue = find\_id\_users.Text;

dataGridView4.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

try

{

int j = 0;

for (int i = 0; i < dataGridView4.Rows.Count - 1; i++)

{

if (dataGridView4.Rows[i].Cells[0].Value.ToString() == find\_id\_users.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Пользователь не найден!");

foreach (DataGridViewRow row in dataGridView4.Rows)

{

if (row.Cells[0].Value.ToString().TrimEnd() == searchValue)

{

row.Selected = true;

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

dataGridView4.Refresh();

find\_id\_users.Clear();

findUsers.Enabled = false;

}

private void addProducts\_Click(object sender, EventArgs e) //ПЯТАЯ ТАБЛИЦА

{

string name = product\_name.Text;

string desc = product\_desc.Text;

string price = product\_price.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Insert into Products([name], [description], [price]) VALUES(@name, @desc, @price);";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@name", name);

cmd1.Parameters.AddWithValue("@desc", desc);

cmd1.Parameters.AddWithValue("@price", price);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.productsTableAdapter.Fill(this.практика\_КудаевDataSetProducts.Products);

product\_id.Clear();

product\_name.Clear();

product\_desc.Clear();

product\_price.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView5.Rows.Count - 1; i++)

{

if (dataGridView5.Rows[i].Cells[0].Value.ToString() == product\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Продукт не найден!");

string id = product\_id.Text;

string name = product\_name.Text;

string desc = product\_desc.Text;

string price = product\_price.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Update Products set [name] = @name, [description] = @desc, [price] = @price where [id] = @id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

cmd1.Parameters.AddWithValue("@name", name);

cmd1.Parameters.AddWithValue("@desc", desc);

cmd1.Parameters.AddWithValue("@price", price);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.productsTableAdapter.Fill(this.практика\_КудаевDataSetProducts.Products);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

product\_id.Clear();

product\_name.Clear();

product\_desc.Clear();

product\_price.Clear();

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView5.Rows.Count - 1; i++)

{

if (dataGridView5.Rows[i].Cells[0].Value.ToString() == product\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Продукт не найден!");

string id = product\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = "Delete from Products where [id] = @id";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.productsTableAdapter.Fill(this.практика\_КудаевDataSetProducts.Products);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

product\_id.Clear();

product\_name.Clear();

product\_desc.Clear();

product\_price.Clear();

}

private void descProducts\_CheckedChanged(object sender, EventArgs e)

{

if (descProducts.Checked)

ascProducts.Checked = false;

else if (ascProducts.Checked)

descProducts.Checked = false;

}

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

{

if (descProducts.Checked)

{

switch (comboBox5.Text)

{

case "id":

dataGridView5.Sort(dataGridView5.Columns[0], ListSortDirection.Descending);

break;

case "Название":

dataGridView5.Sort(dataGridView5.Columns[1], ListSortDirection.Descending);

break;

case "Описание":

dataGridView5.Sort(dataGridView5.Columns[2], ListSortDirection.Descending);

break;

case "Цена":

dataGridView5.Sort(dataGridView5.Columns[3], ListSortDirection.Descending);

break;

}

}

else if (ascProducts.Checked)

{

switch (comboBox5.Text)

{

case "id":

dataGridView5.Sort(dataGridView5.Columns[0], ListSortDirection.Ascending);

break;

case "Название":

dataGridView5.Sort(dataGridView5.Columns[1], ListSortDirection.Ascending);

break;

case "Описание":

dataGridView5.Sort(dataGridView5.Columns[2], ListSortDirection.Ascending);

break;

case "Цена":

dataGridView5.Sort(dataGridView5.Columns[3], ListSortDirection.Ascending);

break;

}

}

}

private void find\_id\_products\_TextChanged(object sender, EventArgs e)

{

if (find\_id\_products.Text == "")

findProducts.Enabled = false;

else findProducts.Enabled = true;

}

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

{

dataGridView5.ClearSelection();

string searchValue = find\_id\_products.Text;

dataGridView5.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

try

{

int j = 0;

for (int i = 0; i < dataGridView5.Rows.Count - 1; i++)

{

if (dataGridView5.Rows[i].Cells[0].Value.ToString() == find\_id\_products.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Продукт не найден!");

foreach (DataGridViewRow row in dataGridView5.Rows)

{

if (row.Cells[0].Value.ToString().TrimEnd() == searchValue)

{

row.Selected = true;

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

dataGridView5.Refresh();

find\_id\_products.Clear();

findProducts.Enabled = false;

}

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

{

DateTime purchaseDate = dateTimePicker2.Value;

DateTime del = dateTimePicker3.Value;

string sup = purchase\_sup.Text;

string tot = purchase\_total.Text;

string product\_id = purchase\_product\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"INSERT INTO Purchases ([purchase\_date], [purchase\_delivery], [supplier], [total\_cost], [total\_items], [product\_id]) SELECT @date, @del, @sup, p.price\* @tot, @tot, @product\_id FROM products p WHERE p.id = @product\_id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@date", purchaseDate);

cmd1.Parameters.AddWithValue("@del", del);

cmd1.Parameters.AddWithValue("@sup", sup);

cmd1.Parameters.AddWithValue("@tot", tot);

cmd1.Parameters.AddWithValue("@product\_id", product\_id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.purchasesTableAdapter.Fill(this.практика\_КудаевDataSetPurchases.Purchases);

purchase\_id.Clear();

purchase\_sup.Clear();

purchase\_total.Clear();

purchase\_product\_id.Clear();

dateTimePicker2.Value = DateTime.Now;

dateTimePicker3.Value = DateTime.Now;

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView6.Rows.Count - 1; i++)

{

if (dataGridView6.Rows[i].Cells[0].Value.ToString() == purchase\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Покупка не найдена!");

string id = purchase\_id.Text;

DateTime purchaseDate = dateTimePicker2.Value;

DateTime del = dateTimePicker3.Value;

string sup = purchase\_sup.Text;

string tot = purchase\_total.Text;

string product\_id = purchase\_product\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"UPDATE p SET p.purchase\_date = @date, p.purchase\_delivery = @del, p.supplier = @sup, p.total\_items = @tot, p.product\_id = @product\_id, p.total\_cost = pr.price \* @tot FROM purchases p INNER JOIN products pr ON p.product\_id = pr.id WHERE p.id = @id;";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

cmd1.Parameters.AddWithValue("@date", purchaseDate);

cmd1.Parameters.AddWithValue("@del", del);

cmd1.Parameters.AddWithValue("@sup", sup);

cmd1.Parameters.AddWithValue("@tot", tot);

cmd1.Parameters.AddWithValue("@product\_id", product\_id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.purchasesTableAdapter.Fill(this.практика\_КудаевDataSetPurchases.Purchases);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

purchase\_id.Clear();

purchase\_sup.Clear();

purchase\_total.Clear();

purchase\_product\_id.Clear();

dateTimePicker2.Value = DateTime.Now;

dateTimePicker3.Value = DateTime.Now;

}

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

{

try

{

int j = 0;

for (int i = 0; i < dataGridView6.Rows.Count - 1; i++)

{

if (dataGridView6.Rows[i].Cells[0].Value.ToString() == purchase\_id.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Покупка не найдена!");

string id = purchase\_id.Text;

string connectionString = @"Data Source = ADCLG1; Initial catalog=Практика\_Кудаев; Integrated Security=True";

SqlConnection MyConnection = new SqlConnection(connectionString);

string ComDel = $"Delete from Purchases where [id] = @id";

SqlCommand cmd1 = new SqlCommand(ComDel, MyConnection);

cmd1.Parameters.AddWithValue("@id", id);

MyConnection.Open();

cmd1.ExecuteNonQuery();

MyConnection.Close();

this.purchasesTableAdapter.Fill(this.практика\_КудаевDataSetPurchases.Purchases);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

purchase\_id.Clear();

purchase\_sup.Clear();

purchase\_total.Clear();

purchase\_product\_id.Clear();

dateTimePicker2.Value = DateTime.Now;

dateTimePicker3.Value = DateTime.Now;

}

private void descPurchases\_CheckedChanged(object sender, EventArgs e)

{

if (descPurchases.Checked)

ascPurchases.Checked = false;

else if (ascPurchases.Checked)

descPurchases.Checked = false;

}

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

{

if (descPurchases.Checked)

{

switch (comboBox6.Text)

{

case "id":

dataGridView6.Sort(dataGridView6.Columns[0], ListSortDirection.Descending);

break;

case "Дата покупки":

dataGridView6.Sort(dataGridView6.Columns[1], ListSortDirection.Descending);

break;

case "Дата доставки":

dataGridView6.Sort(dataGridView6.Columns[2], ListSortDirection.Descending);

break;

case "Поставщик":

dataGridView6.Sort(dataGridView6.Columns[3], ListSortDirection.Descending);

break;

case "Вся стоимость":

dataGridView6.Sort(dataGridView6.Columns[4], ListSortDirection.Descending);

break;

case "Всего предметов":

dataGridView6.Sort(dataGridView6.Columns[5], ListSortDirection.Descending);

break;

case "id продукта":

dataGridView6.Sort(dataGridView6.Columns[6], ListSortDirection.Descending);

break;

}

}

else if (ascPurchases.Checked)

{

switch (comboBox6.Text)

{

case "id":

dataGridView6.Sort(dataGridView6.Columns[0], ListSortDirection.Ascending);

break;

case "Дата покупки":

dataGridView6.Sort(dataGridView6.Columns[1], ListSortDirection.Ascending);

break;

case "Дата доставки":

dataGridView6.Sort(dataGridView6.Columns[2], ListSortDirection.Ascending);

break;

case "Поставщик":

dataGridView6.Sort(dataGridView6.Columns[3], ListSortDirection.Ascending);

break;

case "Вся стоимость":

dataGridView6.Sort(dataGridView6.Columns[4], ListSortDirection.Ascending);

break;

case "Всего предметов":

dataGridView6.Sort(dataGridView6.Columns[5], ListSortDirection.Ascending);

break;

case "id продукта":

dataGridView6.Sort(dataGridView6.Columns[6], ListSortDirection.Ascending);

break;

}

}

}

private void find\_id\_purchases\_TextChanged(object sender, EventArgs e)

{

if (find\_id\_purchases.Text == "")

findPurchases.Enabled = false;

else findPurchases.Enabled = true;

}

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

{

dataGridView6.ClearSelection();

string searchValue = find\_id\_purchases.Text;

dataGridView6.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

try

{

int j = 0;

for (int i = 0; i < dataGridView6.Rows.Count - 1; i++)

{

if (dataGridView6.Rows[i].Cells[0].Value.ToString() == find\_id\_purchases.Text)

{

j++;

break;

}

}

if (j == 0)

throw new Exception("Покупка не найдена!");

foreach (DataGridViewRow row in dataGridView6.Rows)

{

if (row.Cells[0].Value.ToString().TrimEnd() == searchValue)

{

row.Selected = true;

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

dataGridView6.Refresh();

find\_id\_purchases.Clear();

findPurchases.Enabled = false;

}

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

{

colorDialog1.AllowFullOpen = false;

colorDialog1.ShowHelp = true;

colorDialog1.Color = this.BackColor;

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

this.BackColor = colorDialog1.Color;

tabPage1.BackColor = colorDialog1.Color;

tabPage2.BackColor = colorDialog1.Color;

tabPage3.BackColor = colorDialog1.Color;

tabPage4.BackColor = colorDialog1.Color;

tabPage5.BackColor = colorDialog1.Color;

tabPage6.BackColor = colorDialog1.Color;

panel1.BackColor = colorDialog1.Color;

panel2.BackColor = colorDialog1.Color;

panel3.BackColor = colorDialog1.Color;

panel4.BackColor = colorDialog1.Color;

panel5.BackColor = colorDialog1.Color;

panel6.BackColor = colorDialog1.Color;

}

}

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

{

fontDialog1.ShowColor = true;

fontDialog1.Font = panel1.Font;

fontDialog1.Color = panel1.ForeColor;

if (fontDialog1.ShowDialog() != DialogResult.Cancel)

{

Panel panel = this.panel1;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

panel = this.panel2;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

panel = this.panel3;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

panel = this.panel4;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

panel = this.panel5;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

panel = this.panel6;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

panel = this.panel7;

panel.Font = fontDialog1.Font;

panel.ForeColor = fontDialog1.Color;

foreach (Control control in panel.Controls)

{

control.Font = panel.Font;

}

ascOrders.Font = fontDialog1.Font;

ascOrders.ForeColor = fontDialog1.Color;

descOrders.Font = fontDialog1.Font;

descOrders.ForeColor = fontDialog1.Color;

ascOperations.Font = fontDialog1.Font;

ascOperations.ForeColor = fontDialog1.Color;

descOperations.Font = fontDialog1.Font;

descOperations.ForeColor = fontDialog1.Color;

ascStocks.Font = fontDialog1.Font;

ascStocks.ForeColor = fontDialog1.Color;

descStocks.Font = fontDialog1.Font;

descStocks.ForeColor = fontDialog1.Color;

ascUsers.Font = fontDialog1.Font;

ascUsers.ForeColor = fontDialog1.Color;

descUsers.Font = fontDialog1.Font;

descUsers.ForeColor = fontDialog1.Color;

ascProducts.Font = fontDialog1.Font;

ascProducts.ForeColor = fontDialog1.Color;

descProducts.Font = fontDialog1.Font;

descProducts.ForeColor = fontDialog1.Color;

ascPurchases.Font = fontDialog1.Font;

ascPurchases.ForeColor = fontDialog1.Color;

descPurchases.Font = fontDialog1.Font;

descPurchases.ForeColor = fontDialog1.Color;

}

}

private void button1\_Click\_1(object sender, EventArgs e)

{

fontDialog1.ShowColor = true;

fontDialog1.Font = panel1.Font;

fontDialog1.Color = panel1.ForeColor;

if (fontDialog1.ShowDialog() != DialogResult.Cancel)

{

addOrders.Font = fontDialog1.Font;

addOrders.ForeColor = fontDialog1.Color;

editOrders.Font = fontDialog1.Font;

editOrders.ForeColor = fontDialog1.Color;

deleteOrders.Font = fontDialog1.Font;

deleteOrders.ForeColor = fontDialog1.Color;

addOperations.Font = fontDialog1.Font;

addOperations.ForeColor = fontDialog1.Color;

editOperations.Font = fontDialog1.Font;

editOperations.ForeColor = fontDialog1.Color;

deleteOperations.Font = fontDialog1.Font;

deleteOperations.ForeColor = fontDialog1.Color;

addStocks.Font = fontDialog1.Font;

addStocks.ForeColor = fontDialog1.Color;

editStocks.Font = fontDialog1.Font;

editStocks.ForeColor = fontDialog1.Color;

deleteStocks.Font = fontDialog1.Font;

deleteStocks.ForeColor = fontDialog1.Color;

addUsers.Font = fontDialog1.Font;

addUsers.ForeColor = fontDialog1.Color;

editUsers.Font = fontDialog1.Font;

editUsers.ForeColor = fontDialog1.Color;

deleteUsers.Font = fontDialog1.Font;

deleteUsers.ForeColor = fontDialog1.Color;

addProducts.Font = fontDialog1.Font;

addProducts.ForeColor = fontDialog1.Color;

editProducts.Font = fontDialog1.Font;

editProducts.ForeColor = fontDialog1.Color;

deleteProducts.Font = fontDialog1.Font;

deleteProducts.ForeColor = fontDialog1.Color;

addPurchases.Font = fontDialog1.Font;

addPurchases.ForeColor = fontDialog1.Color;

editPurchases.Font = fontDialog1.Font;

editPurchases.ForeColor = fontDialog1.Color;

deletePurchases.Font = fontDialog1.Font;

deletePurchases.ForeColor = fontDialog1.Color;

groupBox1.Font = fontDialog1.Font;

groupBox1.ForeColor = fontDialog1.Color;

groupBox2.Font = fontDialog1.Font;

groupBox2.ForeColor = fontDialog1.Color;

groupBox3.Font = fontDialog1.Font;

groupBox3.ForeColor = fontDialog1.Color;

groupBox4.Font = fontDialog1.Font;

groupBox4.ForeColor = fontDialog1.Color;

groupBox5.Font = fontDialog1.Font;

groupBox5.ForeColor = fontDialog1.Color;

groupBox6.Font = fontDialog1.Font;

groupBox6.ForeColor = fontDialog1.Color;

groupBox7.Font = fontDialog1.Font;

groupBox7.ForeColor = fontDialog1.Color;

groupBox8.Font = fontDialog1.Font;

groupBox8.ForeColor = fontDialog1.Color;

groupBox9.Font = fontDialog1.Font;

groupBox9.ForeColor = fontDialog1.Color;

groupBox10.Font = fontDialog1.Font;

groupBox10.ForeColor = fontDialog1.Color;

groupBox11.Font = fontDialog1.Font;

groupBox11.ForeColor = fontDialog1.Color;

groupBox12.Font = fontDialog1.Font;

groupBox12.ForeColor = fontDialog1.Color;

}

}

}

ПРИЛОЖЕНИЕ Б

Код тестов:

[TestClass]

public class UnitTest1

{

[TestMethod]

public void NotAuthorization1()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("123", "124"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void Client1()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("client", "password3"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void Employee1()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("employee", "password2"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void Admin1()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("admin", "password1"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void CreateAndClient1()

{

Form1 form = new Form1();

form.Create("123", "123");

var exception = Assert.ThrowsException<Exception>(() => form.Check("123", "123"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void NonAuthorization2()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("", ""));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void Client2()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("клиент", "клиент"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void Employee2()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("сотрудник", "сотрудник"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void Admin2()

{

Form1 form = new Form1();

var exception = Assert.ThrowsException<Exception>(() => form.Check("админ", "админ"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

[TestMethod]

public void CreateAndClient2()

{

Form1 form = new Form1();

form.Create("124", "124");

var exception = Assert.ThrowsException<Exception>(() => form.Check("124", "124"));

Assert.AreEqual("Неправильный логин или пароль!", exception.Message);

}

**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

Кудаев Георгий Александрович

(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.