**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(а) 4 курса 42919/2 группы

Бронин Евгений 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.

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(а) 4 курса 42919/2 группы

Бронин Евгений 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 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. \_\_\_\_\_\_\_\_\_\_\_\_ Бронин Е.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(а) 4 курса 42919/2 группы

Бронин Евгений 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.

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**

TravelPro — это современное турагентство, начавшее свою деятельность в сфере туризма. На данный момент компания предлагает широкий ассортимент туристических услуг, но уже зарекомендовала себя как надежный и эффективный партнер для путешественников в регионе.

Компания TravelPro специализируется на высококачественных туристических услугах, предлагая оптимальные условия для бронирования и организации любых видов туров, будь то отдых на море или экскурсионные туры.

TravelPro сотрудничает с ведущими туристическими операторами, такими как Thomas Cook и TUI, и постоянно обновляет свой ассортимент, отслеживая новейшие тенденции в области туризма.

TravelPro разработала систему управления туристическими услугами, которая обеспечивает эффективное управление бронированием и организацией туров в агентстве. Это мощный инструмент, созданный с учетом потребностей современных турагентств, стремящихся к оптимизации своих внутренних процессов и максимизации прибыли.

В TravelPro уделяют большое внимание деталям и индивидуальному подходу к каждому клиенту. Команда опытных специалистов готова помочь вам с выбором тура, бронированием и организацией путешествия, а также предоставить поддержку на всех этапах работы с TravelPro.

TravelPro создает атмосферу надежности и доверия, обеспечивая спокойствие клиентов и их уверенность в качестве своих туров. TravelPro предлагает следующие услуги:

* Бронирование туров и экскурсий
* Организация индивидуальных и групповых туров
* Подбор туров по индивидуальным запросам
* Консультации по выбору тура и организации путешествия
* Поддержка на всех этапах работы с TravelPro

**Техническое задание**

1. **Общие сведения**1.1. Наименование проекта: Разработка программного модуля для учета заявок на туристические услуги.  
   1.2. Заказчик: Турагентство «TravelPro».  
   1.3. Исполнитель: Турагентство «TravelPro».
2. **Функциональные требования**  
   2.1. Возможность добавления заявок в базу данных с указанием следующих параметров:

* Номер заявки;
* Дата добавления;
* Тип тура (отдых, экскурсия, бизнес-тур);
* Дата начала и окончания тура;
* Количество человек;
* ФИО клиента;
* Номер телефона;
* Статус заявки (новая заявка, в процессе бронирования, подтверждена).  
  2.2. Возможность редактирования заявок:
* Изменение типа тура;
* Изменение даты начала и окончания тура;
* Изменение количества человек;
* Изменение статуса заявки.

2.3. Возможность отслеживания статуса заявки:

* Отображение списка заявок;
* Получение уведомлений о смене статуса заявки;
* Поиск заявки по номеру или по параметрам.

2.4. Возможность назначения ответственных за обработку заявок:

* Добавление сотрудника к заявке;
* Отслеживание состояния работы и получение уведомлений о ее завершении;
* Сотрудник может добавлять комментарии на форме заявки и фиксировать информацию о бронировании.

2.5. Расчет статистики работы турагентства:

* Количество подтвержденных заявок;
* Среднее время обработки заявки;
* Статистика по типам туров.

1. **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.

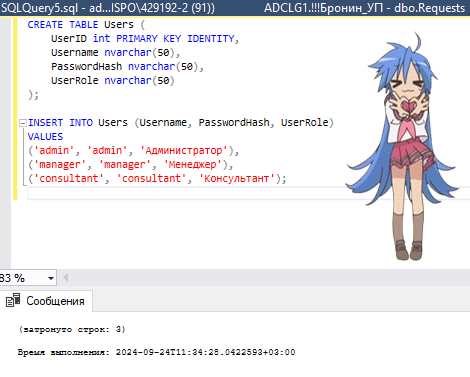
1. **Требования к реализации**  
   4.1. Язык программирования: C# Windows Forms  
   4.2. СУБД: MSSQL
2. **Требования к документации**  
   5.1. Техническое задание на разработку программного модуля.
3. **Руководство по стилю**  
   6.1. Шрифт: Times New Roman 12  
   6.2. Цветовая схема: Основной: Фуксия Крайола(RGB: 195,100,197), Черный(RGB: 0,0,0); Второстепенный: Зеленый(RGB: 80,217,167), Белый(RGB: 255,255,255)

**Задание 2. Разработка алгоритмов и диаграмм**

Для базы данных по предметной области информационной системы турагентства TravelPro необходимы следующие сущности:

* Requests (Заявки) с атрибутами: RequestID, DateAdded, TourType, TourStartDate, TourEndDate, NumberOfPeople, ClientName, ClientPhone, Status;
* Employees (Сотрудники) с атрибутами: EmployeeID, EmployeeName, EmployeePhone, Role;
* Bookings (Бронирования) с атрибутами: BookingID, RequestID, EmployeeID, BookingDate, BookingStatus;
* Comments (Комментарии) с атрибутами: CommentID, RequestID, EmployeeID, CommentText, CommentDate;
* Statistic (Статистика) с атрибутами: StatisticID, TourType, NumberOfBookings, AverageProcessingTime;
* Users (Пользователи) с атрибутами: UserID, Username, PasswordHash, UserRole;

Скрипт создания таблиц БД:

* CREATE TABLE Requests (
* RequestID INT PRIMARY KEY IDENTITY(1,1),
* DateAdded DATETIME NOT NULL,
*  TourType NVARCHAR(50) NOT NULL,
* TourStartDate DATETIME NOT NULL,
* TourEndDate DATETIME NOT NULL,
* NumberOfPeople INT NOT NULL,
* ClientName NVARCHAR(100) NOT NULL,
* ClientPhone NVARCHAR(20) NOT NULL,
* Status NVARCHAR(50) NOT NULL
* );
* CREATE TABLE Employees (
* EmployeeID INT PRIMARY KEY IDENTITY(1,1),
* EmployeeName NVARCHAR(100) NOT NULL,
* EmployeePhone NVARCHAR(20) NOT NULL,
* Role NVARCHAR(50) NOT NULL
* );
* CREATE TABLE Bookings (
* BookingID INT PRIMARY KEY IDENTITY(1,1),
* RequestID INT NOT NULL,
* EmployeeID INT NOT NULL,
* BookingDate DATETIME NOT NULL,
* BookingStatus NVARCHAR(50) NOT NULL,
* CONSTRAINT FK\_Bookings\_Requests FOREIGN KEY (RequestID) REFERENCES Requests(RequestID),
* CONSTRAINT FK\_Bookings\_Employees FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
* );
* CREATE TABLE Comments (
* CommentID INT PRIMARY KEY IDENTITY(1,1),
* RequestID INT NOT NULL,
* EmployeeID INT NOT NULL,
* CommentText NVARCHAR(200) NOT NULL,
* CommentDate DATETIME NOT NULL,
* CONSTRAINT FK\_Comments\_Requests FOREIGN KEY (RequestID) REFERENCES Requests(RequestID),
* CONSTRAINT FK\_Comments\_Employees FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
* );
* CREATE TABLE Statistic (
* StatisticID INT PRIMARY KEY IDENTITY(1,1),
* TourType NVARCHAR(50) NOT NULL,
* NumberOfBookings INT NOT NULL,
* AverageProcessingTime FLOAT NOT NULL
* );
* CREATE TABLE Users (
* UserID int PRIMARY KEY IDENTITY,
* Username nvarchar(50),
* PasswordHash nvarchar(50),
* UserRole nvarchar(50)
* );
* Заполнение таблиц:
* INSERT INTO Requests (DateAdded, TourType, TourStartDate, TourEndDate, NumberOfPeople, ClientName, ClientPhone, Status)
* VALUES
* ('01-01-2024', 'Отдых на море', '01-06-2024', '15-06-2024', 2, 'Иванов Иван Иванович', '+7 999 123 45 67', 'Новая заявка'),
* ('05-01-2024', 'Экскурсионный тур', '01-07-2024', '10-07-2024', 4, 'Петрова Мария Сергеевна', '+7 999 901 23 45', 'В процессе бронирования'),
* ('10-01-2024', 'Бизнес-тур', '01-08-2024', '2022-08-05', 1, 'Сидоров Сергей Иванович', '+7 999 111 22 33', 'Подтверждена');
* INSERT INTO Employees (EmployeeName, EmployeePhone, Role)
* VALUES
* ('Иванова Наталья Сергеевна', '+7 999 123 45 67', 'Менеджер'),
* ('Петров Дмитрий Иванович', '+7 999 901 23 45', 'Консультант'),
* ('Сидорова Елена Васильевна', '+7 999 111 22 33', 'Администратор');
* INSERT INTO Bookings (RequestID, EmployeeID, BookingDate, BookingStatus)
* VALUES
* (4, 1, '15-01-2024', 'Подтверждено'),
* (5, 2, '20-01-2024', 'В процессе бронирования'),
* (6, 3, '25-01-2024', 'Отменено');
* INSERT INTO Comments (RequestID, EmployeeID, CommentText, CommentDate)
* VALUES
* (4, 1, 'Заявка подтверждена', '15-01-2024'),
* (5, 2, 'Заявка в процессе бронирования', '20-01-2024'),
* (6, 3, 'Заявка отменена', '25-01-2024');
* INSERT INTO Statistic (TourType, NumberOfBookings, AverageProcessingTime)
* VALUES
* ('Отдых на море', 10, 3.5),
* ('Экскурсионный тур', 5, 2.2),
* ('Бизнес-тур', 2, 1.8);
* INSERT INTO Users (Username, PasswordHash, UserRole)
* VALUES
* ('admin', 'admin', 'Администратор'),
* ('manager', 'manager', 'Менеджер'),
* ('consultant', 'consultant', 'Консультант');

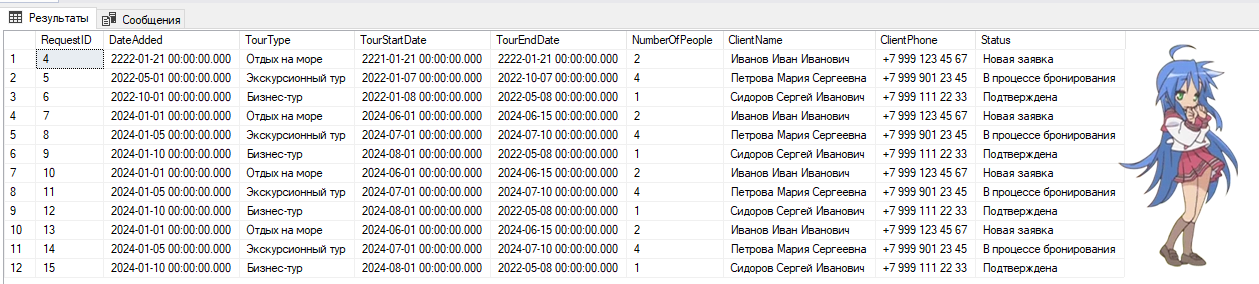


Рис. 1 – таблица Requests

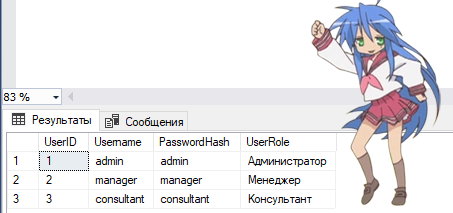


Рис. 2 – таблица Users

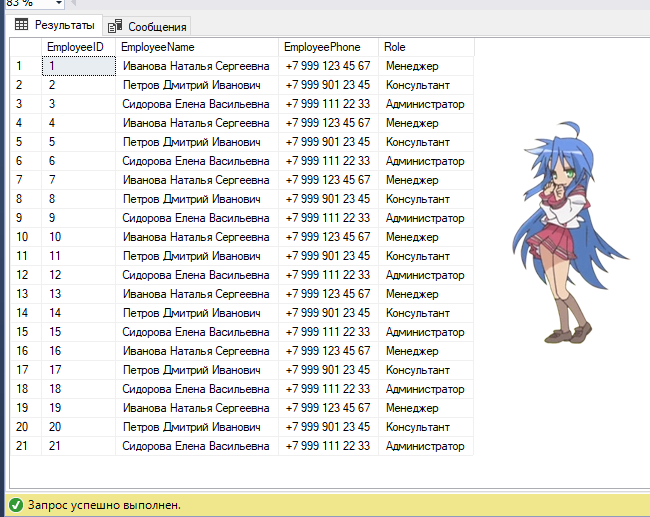


Рис. 3 – таблица Employees

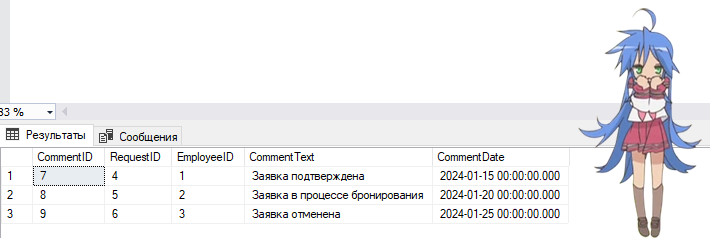


Рис. 4 – таблица Comments

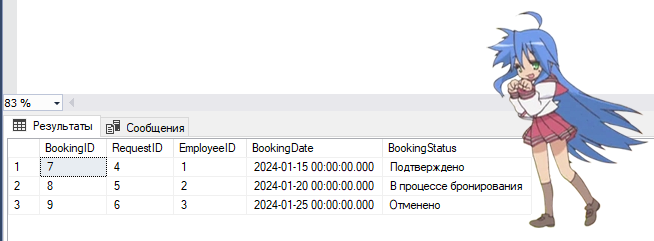


Рис. 5 – таблица Bookings

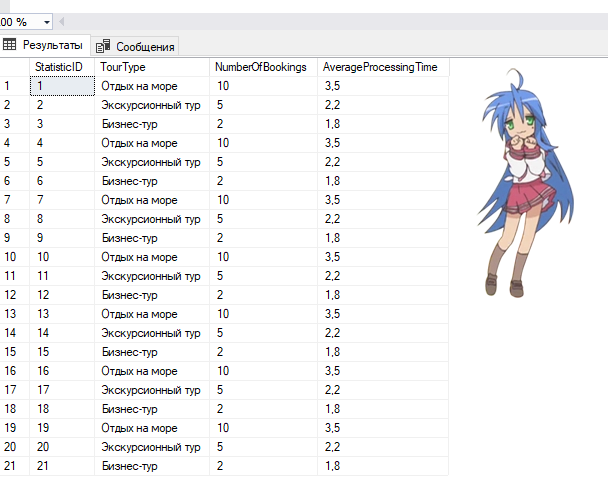


Рис. 6 – таблица Requests

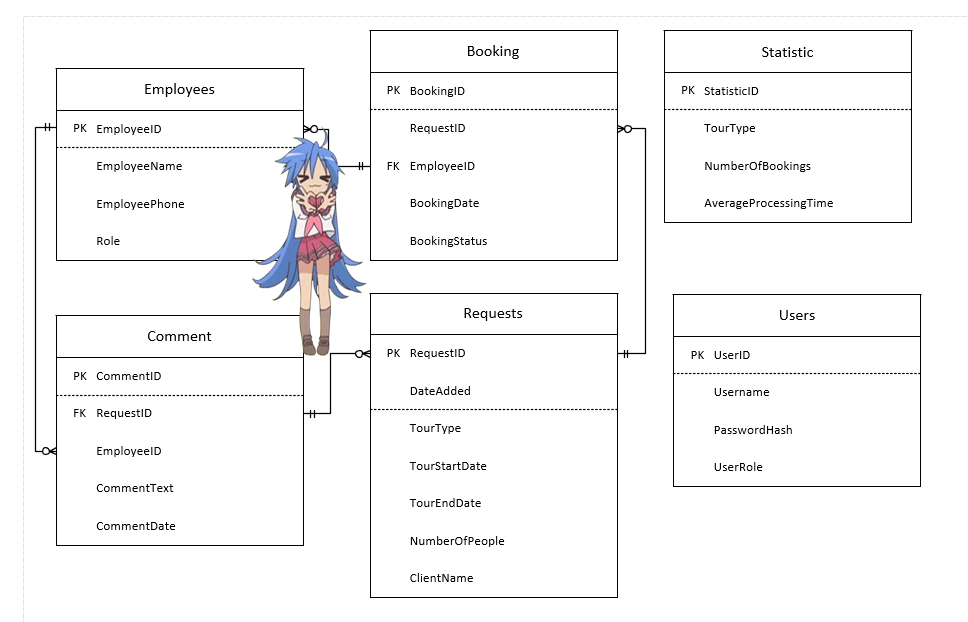


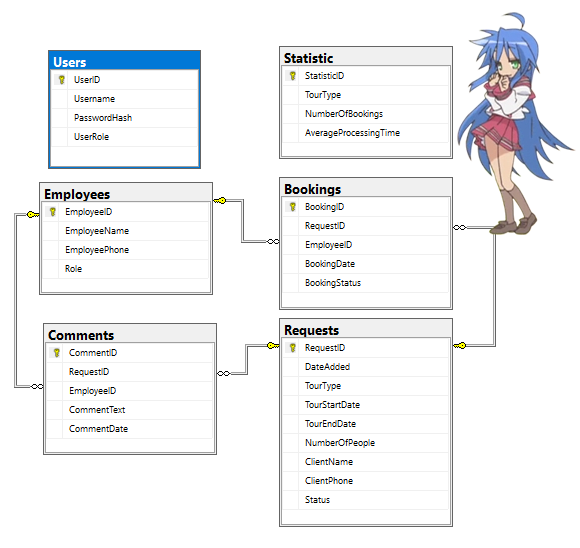
Рис. 7 – ER-модель БД система турагентство 

Рис. 8 – Диаграмма БД

**UML диаграммы**



Рис. 9 – Диаграмма претендентов



Рис. 10 – Диаграмма последовательности



Рис. 11 – Диаграмма активности

**Словарь данных**

**Таблица 1: Заявки (Requests)**

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| RequestID | int | Уникальный идентификатор заявки |
| DateAdded | datetime | Дата добавления заявки |
| TourType | nvarchar(50) | Тип тура (отдых, экскурсия, бизнес-тур) |
| TourStartDate | datetime | Дата начала тура |
| TourEndDate | datetime | Дата окончания тура |
| NumberOfPeople | int | Количество человек |
| ClientName | nvarchar(100) | ФИО клиента |
| ClientPhone | nvarchar(20) | Номер телефона клиента |
| Status | nvarchar(50) | Статус заявки (новая заявка, в процессе бронирования, подтверждена) |

**Таблица 2: Сотрудники (Employees)**

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| EmployeeID | int | Уникальный идентификатор сотрудника |
| EmployeeName | nvarchar(100) | ФИО сотрудника |
| EmployeePhone | nvarchar(20) | Номер телефона сотрудника |
| Role | nvarchar(50) | Роль сотрудника (менеджер, консультант, администратор) |

**Таблица 3: Бронирования (Bookings)**

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| BookingID | int | Уникальный идентификатор бронирования |
| RequestID | int | Идентификатор заявки |
| EmployeeID | int | Идентификатор сотрудника, обработавшего бронирование |
| BookingDate | datetime | Дата бронирования |
| BookingStatus | nvarchar(50) | Статус бронирования (подтверждено, отменено) |

**Таблица 4: Комментарии (Comments)**

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| CommentID | int | Уникальный идентификатор комментария |
| RequestID | int | Идентификатор заявки |
| EmployeeID | int | Идентификатор сотрудника, оставившего комментарий |
| CommentText | nvarchar(200) | Текст комментария |
| CommentDate | datetime | Дата добавления комментария |

**Таблица 5: Статистика (Statistic)**

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| StatisticID | int | Уникальный идентификатор статистики |
| TourType | nvarchar(50) | Тип тура |
| NumberOfBookings | int | Количество бронирований |
| AverageProcessingTime | float | Среднее время обработки заявок |

**Задание 3. Разработка макетов программы**

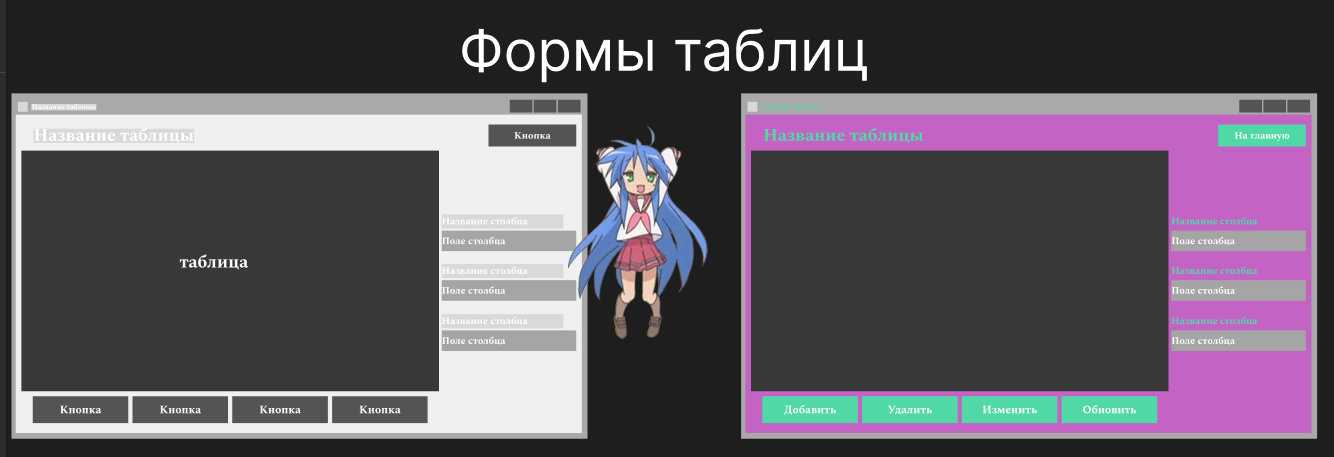


Рис.12 – макеты форм таблиц

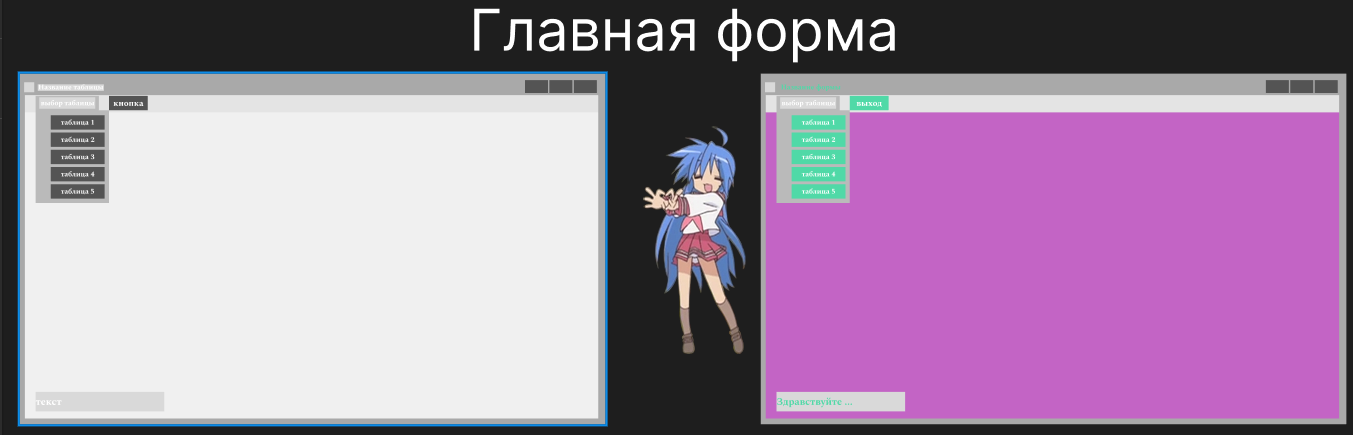


Рис.13 – макет главная форма

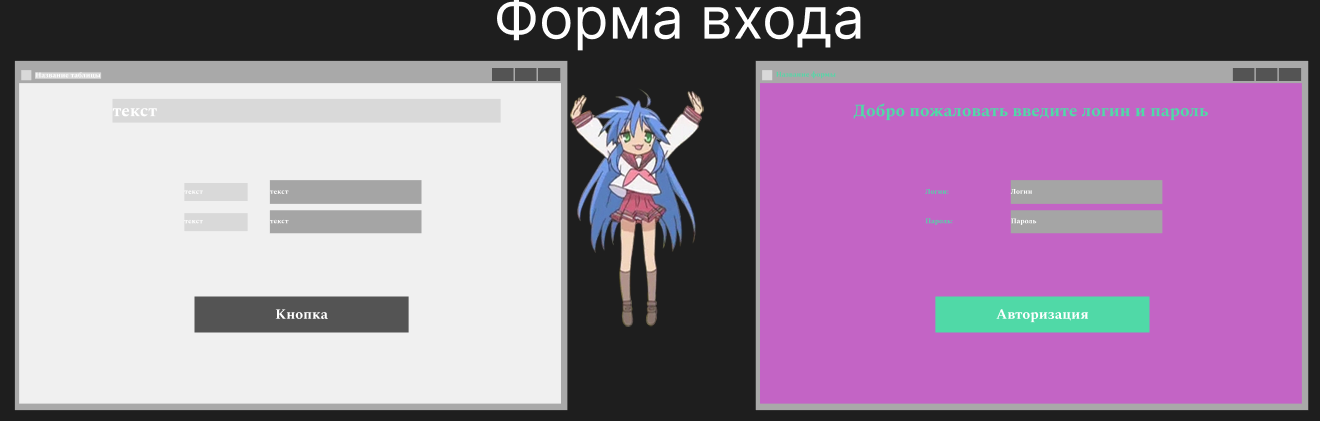


Рис.14 – макеты форм таблиц

**Карта навигации**

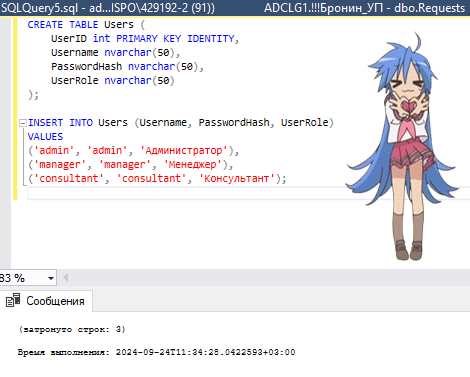


Рис.15 – Карта навигации

**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 и пароль. (Рисунок 16).

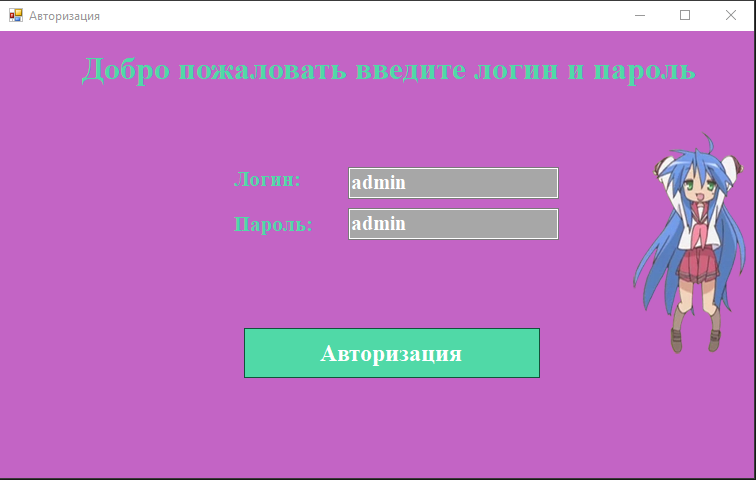


Рисунок 16 – Окно авторизации

После успешной авторизации пользователь попадает в главное меню (Рисунок 17).

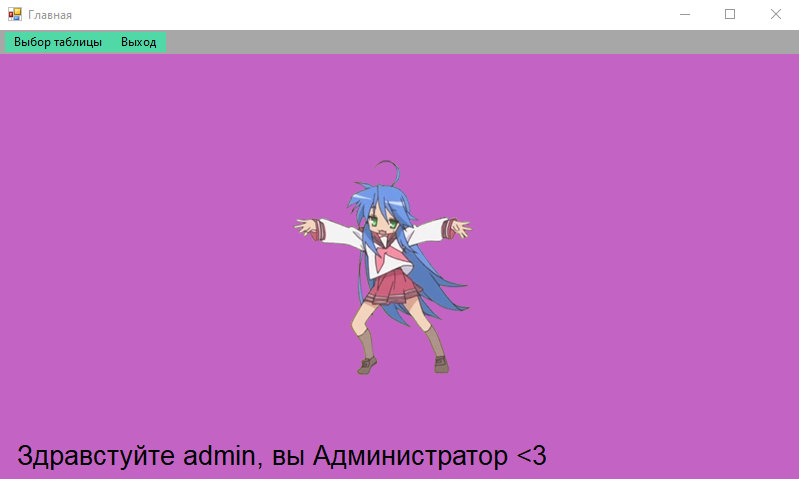


Рисунок 17 – Главное меню

Для открытия любой таблицы и вывода её на экран необходимо в облавти меню нажать на кнопку «Таблицы». Например откроем таблицу «Бронирование» (Рисунок 18).

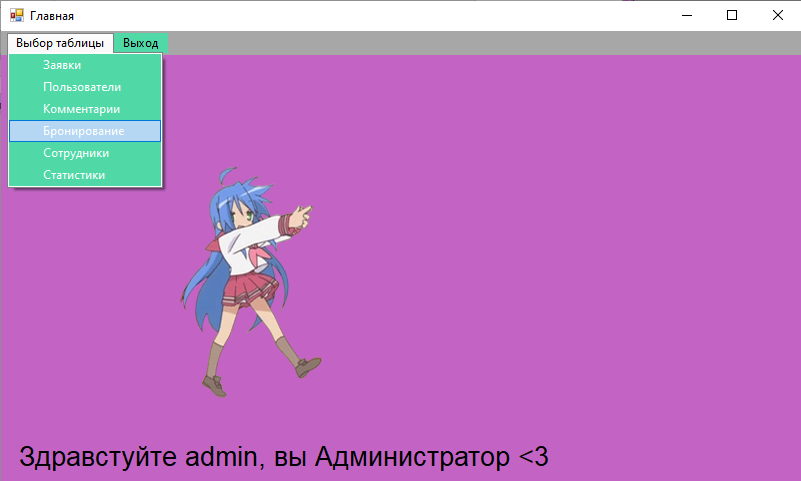


Рисунок 18 – Главное меню с открытым меню «Выбор таблицы»

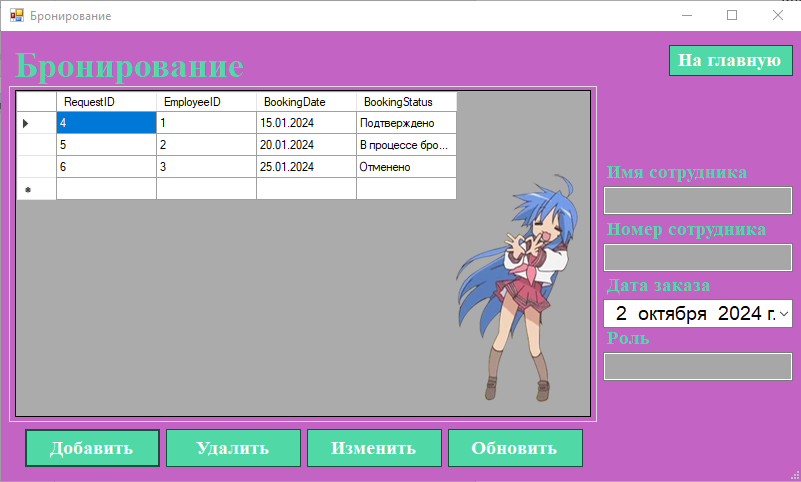
После нажатия но кнопку «Бронирование» в области меню «Выбор таблицы», происходит переход на форму таблицы «Бронирование» (Рисунок 19). 

Рисунок 19 – Таблица «Сотрудники»

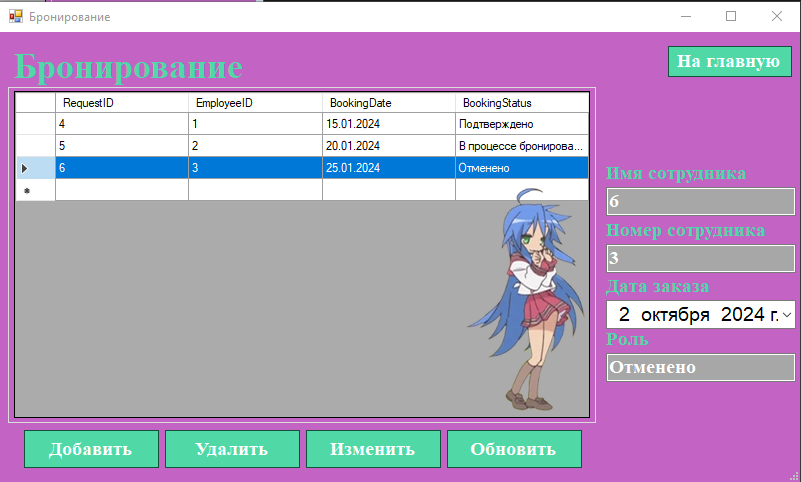
При нажатии на ячейку под красной стрелкой данные из таблицы вводятся в поля формы (Рисунок 20). 

Рисунок 20 – Заполнение полей формы

После мы можем отредактировать любое поле. Нажав кнопку «Изменить» данные сохраняться в таблице. (Рисунок 21-22).

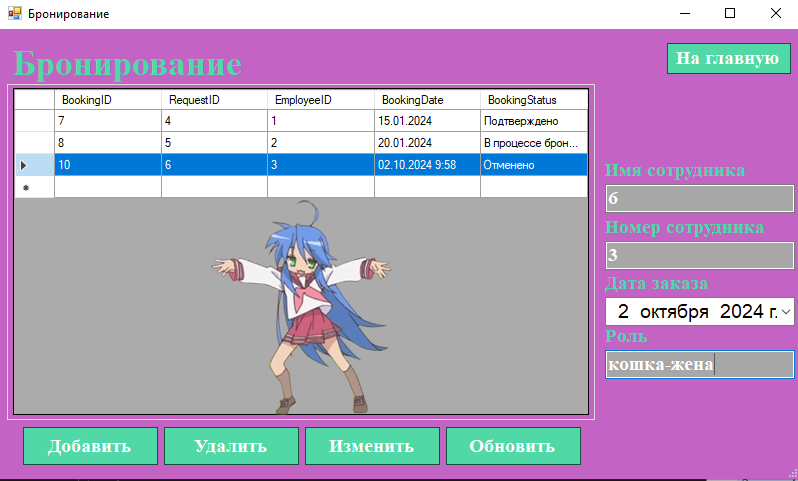


Рисунок 21 – изменение полей формы

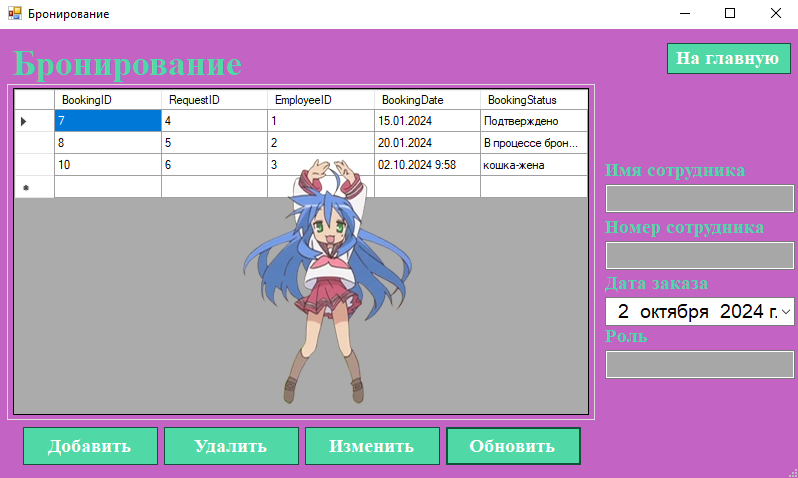


Рисунок 22 – измененная таблица

Впишем в поля данные и добавим их в таблицу (Рисунок 23-24).

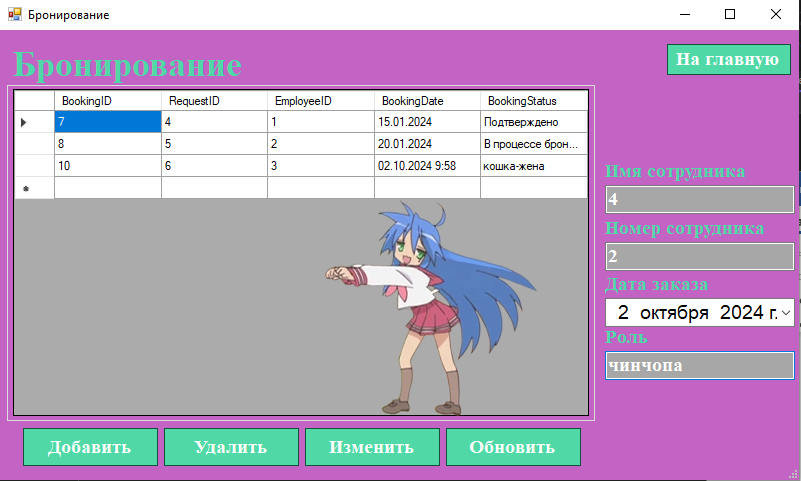


Рисунок 23 – вписанные данные



Рисунок 24 – Обновленная таблица

При нажатии на ячейку под красной стрелкой данные из таблицы вводятся в поля формы. При нажатии кнопки «Удалить» данные удаляться из таблицы (Рисунок 25-26). 

Рисунок 25 – выделенная строка

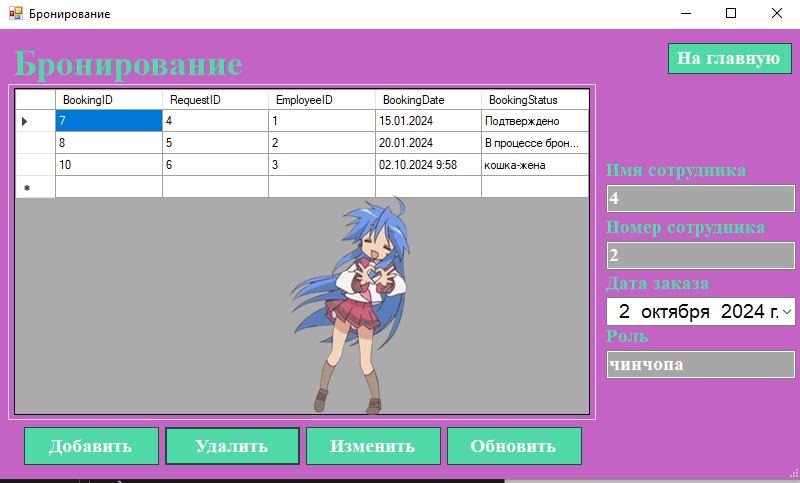


Рисунок 26 – удаленная строка из таблицы

**Задание 5. Тестирование**

Код программы находятся в Приложении Б.

|  |  |
| --- | --- |
| **Название проекта** | TravelPro |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | zxcnut |
| **Дата(ы) теста** | 03.10.2024 |

|  |  |
| --- | --- |
| **Тестовый пример #** | user interface test case #1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации |
| **Краткое изложение теста** | Проверим, будет ли переход на главную форму если вписать в поля логина и пароля пустыми |
| **Этапы теста** | Отрыть программу, вписать логин и пароль, нажать на кнопку авторизоваться |
| **Тестовые данные** | Логин – testlogin, пароль - testpassword |
| **Ожидаемый результат** | Успешный вход |
| **Фактический результат** | Успешный вход |
| **Статус** | Зачет |
| **Предварительное условие** | Рабочий sql server(без него программа не сможет работать), добавить в таблицу пользователи тестовый пароль и логин |
| **Постусловие** | Программа работает, открывается главная форма |
| **Примечания/комментарии** | Я хочу эклерчик //: |

|  |  |
| --- | --- |
| **Тестовый пример #** | user interface test case #2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации |
| **Краткое изложение теста** | Проверим, будет ли переход на главную форму если оставить поле логина пустым |
| **Этапы теста** | Отрыть программу, нажать на кнопку авторизоваться |
| **Тестовые данные** | Логин – «», пароль – testpassword |
| **Ожидаемый результат** | Ошибка |
| **Фактический результат** | Ошибка |
| **Статус** | Зачет |
| **Предварительное условие** | Рабочий sql server(без него программа не сможет работать) |
| **Постусловие** | Ошибка программы, главная форма не откроется |
| **Примечания/комментарии** | Я хочу эклерчик //: |

|  |  |
| --- | --- |
| **Тестовый пример #** | user interface test case #3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка наличия роли |
| **Краткое изложение теста** | Проверим, будет ли выводиться на главной форме любая роль пользователя после входа |
| **Этапы теста** | Отрыть программу, нажать на кнопку авторизоваться |
| **Тестовые данные** | Логин – testlogin, пароль - testpassword |
| **Ожидаемый результат** | Успешный вход, вывод на экран роли пользователя |
| **Фактический результат** | Успешный вход, вывод на экран роли пользователя |
| **Статус** | Зачет |
| **Предварительное условие** | Рабочий sql server(без него программа не сможет работать), добавить в таблицу пользователи тестовый пароль и логин |
| **Постусловие** | Программа работает, открывается главная форма |
| **Примечания/комментарии** | Я хочу эклерчик //: |

|  |  |
| --- | --- |
| **Тестовый пример #** | user interface test case #4 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации |
| **Краткое изложение теста** | Проверим, будет ли вход если ссылка подключения будет неверно |
| **Этапы теста** | Отрыть программу, нажать на кнопку авторизоваться |
| **Тестовые данные** | Логин – testlogin, пароль – testpassword, connectionString - Invalid Connection String |
| **Ожидаемый результат** | Ошибка, вход не выполнится |
| **Фактический результат** | Ошибка, вход не выполнится |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | Ошибка программы, нет подключения |
| **Примечания/комментарии** | Я хочу эклерчик //: |

|  |  |
| --- | --- |
| **Тестовый пример #** | user interface test case #5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации |
| **Краткое изложение теста** | Проверим, будет ли выводиться на главной форме роль пользователя после входа |
| **Этапы теста** | Отрыть программу, нажать на кнопку авторизоваться |
| **Тестовые данные** | Логин – testlogin, пароль – testpassword, expectedUserRole - Мартышка |
| **Ожидаемый результат** | Успешный вход, вывод на экран правильной роли пользователя |
| **Фактический результат** | Успешный вход, вывод на экран правильной роли пользователя |
| **Статус** | Зачет |
| **Предварительное условие** | Рабочий sql server(без него программа не сможет работать), добавить в таблицу пользователи тестовый пароль и логин, указать роль пользователя |
| **Постусловие** | - |
| **Примечания/комментарии** | Я хочу эклерчик //: |

**Результаты тестов**

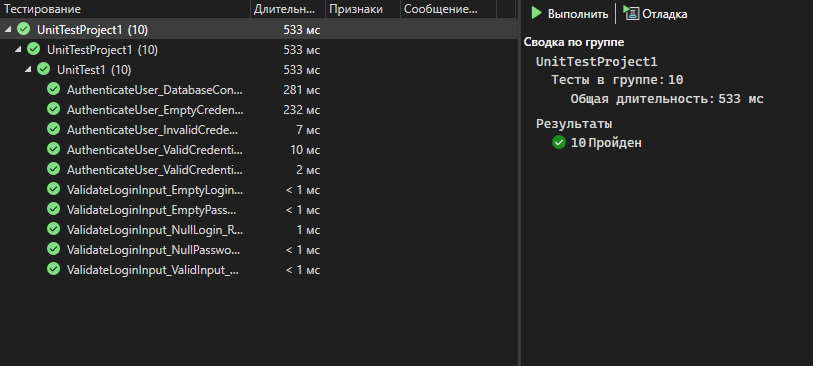


Рисунок 27 – Результат проведения UnitTest

**Задание 6. GitHub. Оценка проекта**

Создаём пустую папку и инициализируем репозиторий командой “git init” (Рисунок 28).

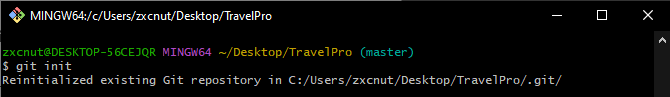


Рисунок 28 – Устанавливаем репозиторий

Статус хранения репозитория можно проверить командой “git status” (Рисунок 29).

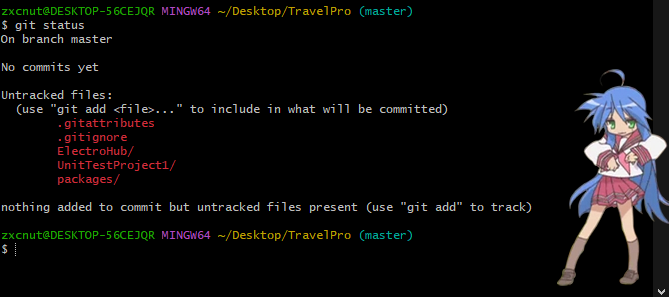


Рисунок 29 – Проверка на статус хранения

Несохранные данные выделенные красным цветом, чтобы сохранить используем команду “git add .” (Рисунок 30).

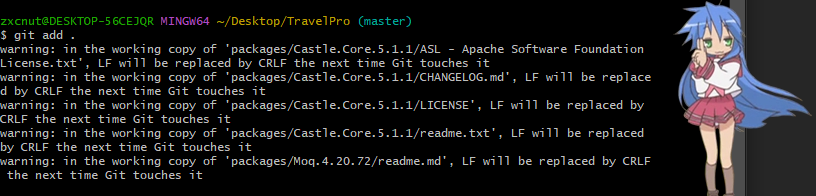


Рисунок 30 – Сохранением данные

Проверим сохранение данных командой “git status” (Рисунок 31).



Рисунок 31 – Проверка сохранения данных

Создадим комит командой “git commit -m «Inittial commit»” (Рисунок 32).

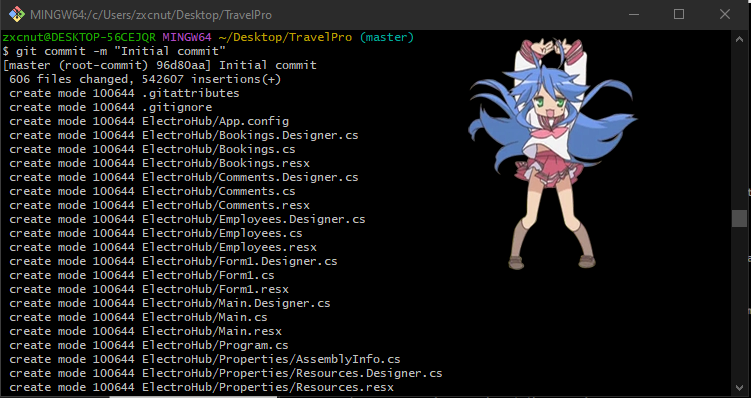


Рисунок 32 – Создание коммита с именем «Inittial commit»

Создаем репозиторий на сайте Github (Рисунок 33).

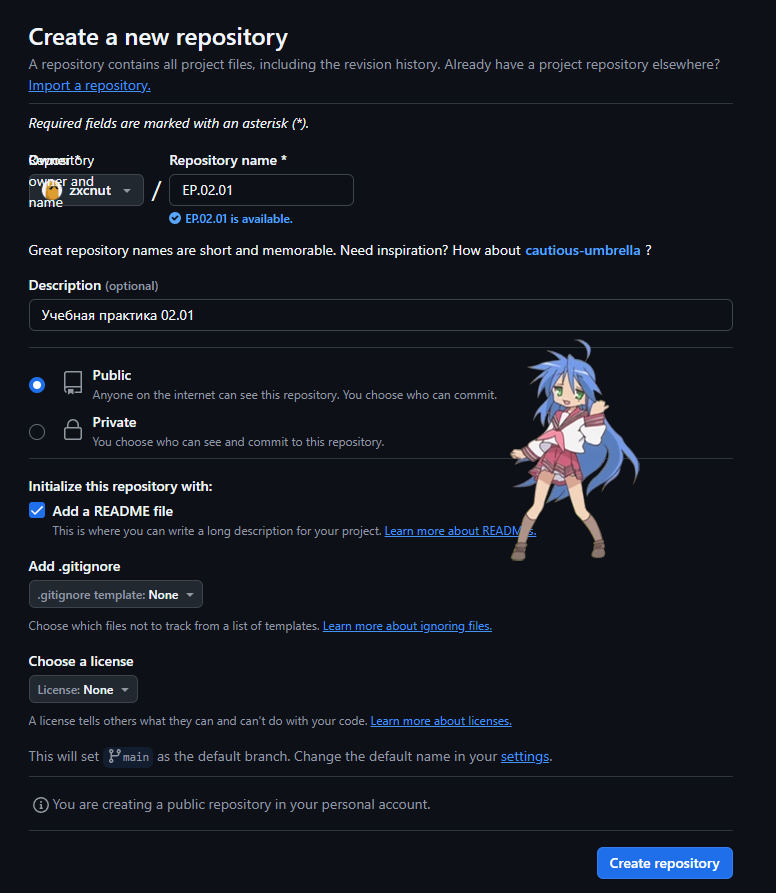


Рисунок 33 – Создаём удалённый репозиторий

Установим в Git свой репозиторий как удаленный через “git remote add origin” и установим репозиторий как удалённый командой “git push -u origin master” (Рисунок 34).

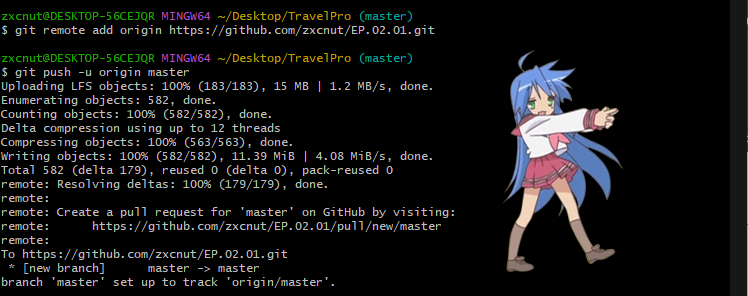


Рисунок 34 – Добавляем в git проект

Результат репозитория github (Рисунок 35).

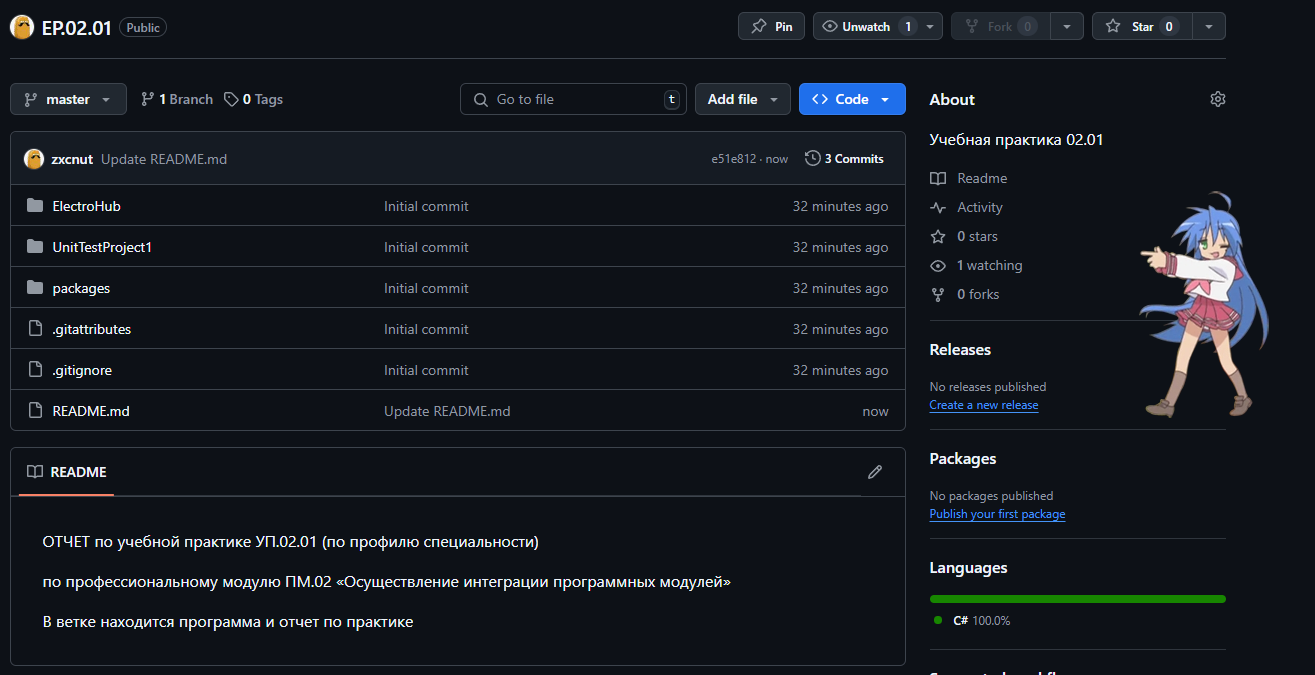


Рисунок 35 – Репозиторий github

**Приложение А**

Код программы

**Код формы Form1:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ElectroHub

{

public partial class Form1 : Form

{

//string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private string connectionString = @"Data Source= adclg1; Initial catalog=!!!Бронин\_УП; Integrated Security=True";

public Form1()

{

InitializeComponent();

}

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

{

string login = textBox1.Text.Trim();

string password = textBox2.Text.Trim();

if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))

{

MessageBox.Show("Пожалуйста, введите логин и пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT UserRole FROM Users WHERE Username = @Username and PasswordHash = @PasswordHash";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Username", login);

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

connection.Open();

object role = command.ExecuteScalar();

if (role != null)

{

string userRole = (string)role;

UserManager.SetCurrentUser(login, userRole);

Main form2 = new Main();

form2.Show();

this.Hide();

}

else

{

MessageBox.Show("Неверный логин или пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

**Код формы Employees:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

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

namespace ElectroHub

{

public partial class Employees : Form

{

private string connectionString = @"Data Source=adclg1; Initial catalog=!!!Бронин\_УП; Integrated Security=True";

private SqlConnection connection;

public Employees()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

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

{

LoadEmployees();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["EmployeeName"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["EmployeePhone"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["Role"].Value.ToString();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

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

{

string employeeName = textBox1.Text;

string employeePhone = textBox2.Text;

string role = textBox3.Text;

string query = "INSERT INTO Employees (EmployeeName, EmployeePhone, Role) VALUES (@EmployeeName, @EmployeePhone, @Role)";

ExecuteNonQuery(query, ("@EmployeeName", employeeName), ("@EmployeePhone", employeePhone), ("@Role", role));

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int employeesID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["RequestsID"].Value);

string query1 = "DELETE FROM Employees WHERE EmployeesID = @EmployeesID";

string query2 = "DELETE FROM Bookings WHERE EmployeesID = @EmployeesID";

string query3 = "DELETE FROM Comments WHERE EmployeesID = @EmployeesID";

ExecuteNonQuery(query1, ("@EmployeesID", employeesID));

ExecuteNonQuery(query2, ("@EmployeesID", employeesID));

ExecuteNonQuery(query3, ("@EmployeesID", employeesID));

}

else

{

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

}

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int employeesID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["EmployeesID"].Value);

string employeeName = textBox1.Text;

string employeePhone = textBox2.Text;

string role = textBox3.Text;

string query = "UPDATE Requests SET EmployeeName = @EmployeeName, EmployeePhone = @EmployeePhone, Role = @Role WHERE EmployeesID = @EmployeesID";

ExecuteNonQuery(query, ("@EmployeeName", employeeName), ("@EmployeePhone", employeePhone), ("@Role", role));

}

else

{

MessageBox.Show("Выберите заказ для обновления.");

}

}

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

{

LoadEmployees();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadEmployees(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadEmployees()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT EmployeeName, EmployeePhone, Role FROM Employees";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

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

{

Main.Instance.Show();

}

}

}

**Код формы Statistic:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TravelPro

{

public partial class Statistic : Form

{

private string connectionString = @"Data Source=adclg1; Initial catalog=!!!Бронин\_УП; Integrated Security=True";

private SqlConnection connection;

public Statistic()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

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

{

LoadStatistic();

}

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

{

Main.Instance.Show();

}

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

{

string tourType = textBox1.Text;

int numberOfBookings = Convert.ToInt32(textBox2.Text);

float averageProcessingTime = Convert.ToSingle(textBox3.Text);

string query = "INSERT INTO Statistic (TourType, NumberOfBookings, AverageProcessingTime) VALUES (@RequestID, @NumberOfBookings, @AverageProcessingTime)";

ExecuteNonQuery(query, ("@TourType ", tourType), ("@NumberOfBookings", numberOfBookings), ("@AverageProcessingTime", averageProcessingTime));

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int statisticID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["StatisticID"].Value);

string query1 = "DELETE FROM Statistic WHERE StatisticID = @StatisticID";

ExecuteNonQuery(query1, ("@StatisticID", statisticID));

}

else

{

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

}

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int statisticID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["StatisticID"].Value);

string tourType = textBox1.Text;

int numberOfBookings = Convert.ToInt32(textBox2.Text);

float averageProcessingTime = Convert.ToSingle(textBox3.Text);

string query = "UPDATE Statistic SET TourType = @TourType, NumberOfBookings = @NumberOfBookings, AverageProcessingTime = @AverageProcessingTime WHERE StatisticID = @StatisticID";

ExecuteNonQuery(query, ("@TourType", tourType), ("@NumberOfBookings", numberOfBookings), ("@AverageProcessingTime", averageProcessingTime), ("@StatisticID", statisticID));

}

else

{

MessageBox.Show("Выберите заказ для обновления.");

}

}

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

{

LoadStatistic();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["TourType"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["NumberOfBookings"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["AverageProcessingTime"].Value.ToString();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadStatistic(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadStatistic()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT TourType, NumberOfBookings, AverageProcessingTime FROM Statistic";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

}

**Код формы Requests:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

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

namespace TravelPro

{

public partial class Requests : Form

{

private string connectionString = @"Data Source=adclg1; Initial catalog=!!!Бронин\_УП; Integrated Security=True";

private SqlConnection connection;

public Requests()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

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

{

Main.Instance.Show();

}

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

{

DateTime dateAdded = dateTimePicker1.Value;

DateTime tourStartDate = dateTimePicker2.Value;

DateTime tourEndDate = dateTimePicker3.Value;

string tourType = textBox1.Text;

int numberOfPeople = Convert.ToInt32(textBox2.Text);

string clientName = textBox3.Text;

string clientPhone = textBox4.Text;

string status = textBox5.Text;

string query = "INSERT INTO Requests (DateAdded, TourType, TourStartDate, TourEndDate, NumberOfPeople, ClientName, ClientPhone, Status) VALUES (@DateAdded, @TourType, @TourStartDate, @TourEndDate, @NumberOfPeople, @ClientName, @ClientPhone, @Status)";

ExecuteNonQuery(query, ("@DateAdded", dateAdded), ("@TourType", tourType), ("@TourStartDate", tourStartDate), ("@TourEndDate", tourEndDate), ("@NumberOfPeople", numberOfPeople), ("@ClientName", clientName), ("@ClientPhone", clientPhone), ("@Status", status));

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int requestsID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["RequestsID"].Value);

string query1 = "DELETE FROM Requests WHЕRE RequestsID = @RequestsID";

string query2 = "DELETE FROM Bookings WHERE RequestsID = @RequestsID";

string query3 = "DELETE FROM Comments WHERE RequestsID = @RequestsID";

ExecuteNonQuery(query1, ("@RequestsID", requestsID));

ExecuteNonQuery(query2, ("@RequestsID", requestsID));

ExecuteNonQuery(query3, ("@RequestsID", requestsID));

}

else

{

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

}

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int requestsID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["RequestsID"].Value);

DateTime dateAdded = dateTimePicker1.Value;

DateTime tourStartDate = dateTimePicker2.Value;

DateTime tourEndDate = dateTimePicker3.Value;

string tourType = textBox1.Text;

int numberOfPeople = Convert.ToInt32(textBox2.Text);

string clientName = textBox3.Text;

string clientPhone = textBox4.Text;

string status = textBox5.Text;

string query = "UPDATE Requests SET DateAdded = @DateAdded, TourType = @TourType, TourStartDate = @TourStartDate, TourEndDate = @TourEndDate, NumberOfPeople = @NumberOfPeople, ClientName = @ClientName, ClientPhone = @ClientPhone, Status = @Status WHERE RequestsID = @RequestsID";

ExecuteNonQuery(query, ("@DateAdded", dateAdded), ("@TourType", tourType), ("@TourStartDate", tourStartDate), ("@TourEndDate", tourEndDate), ("@NumberOfPeople", numberOfPeople), ("@ClientName", clientName), ("@ClientPhone", clientPhone), ("@Status", status), ("@RequestsID", requestsID));

}

else

{

MessageBox.Show("Выберите заказ для обновления.");

}

}

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

{

LoadRequests();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

DateTime dateAdded = DateTime.Today;

DateTime tourStartDate = DateTime.Today;

DateTime tourEndDate = DateTime.Today;

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

string dateString1 = dataGridView1.SelectedRows[0].Cells["DateAdded"].Value.ToString();

DateTime date1;

string dateString2 = dataGridView1.SelectedRows[0].Cells["TourStartDate"].Value.ToString();

DateTime date2;

string dateString3 = dataGridView1.SelectedRows[0].Cells["TourEndDate"].Value.ToString();

DateTime date3;

if (DateTime.TryParseExact(dateString1, "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out date1))

{

dateTimePicker1.Value = date1;

}

if (DateTime.TryParseExact(dateString2, "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out date2))

{

dateTimePicker2.Value = date2;

}

if (DateTime.TryParseExact(dateString3, "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out date3))

{

dateTimePicker3.Value = date3;

}

textBox1.Text = dataGridView1.SelectedRows[0].Cells["TourType"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["NumberOfPeople"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["ClientName"].Value.ToString();

textBox4.Text = dataGridView1.SelectedRows[0].Cells["ClientPhone"].Value.ToString();

textBox5.Text = dataGridView1.SelectedRows[0].Cells["Status"].Value.ToString();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadRequests(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadRequests()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT DateAdded, TourType, TourStartDate, TourEndDate, NumberOfPeople, ClientName, ClientPhone, Status FROM Requests";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

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

{

LoadRequests();

}

}

}

**Код формы Main:**

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.Windows.Forms;

namespace TravelPro

{

public partial class Main : Form

{

public static Main Instance { get; private set; }

public Main()

{

InitializeComponent();

Instance = this;

}

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

{

string currentRole = UserManager.CurrentUser.Role;

string userName = UserManager.CurrentUser.Username;

label1.Text = $"Здравстуйте {userName}, вы {currentRole} <3";

if (currentRole != "Администратор")

{

пользователиToolStripMenuItem.Enabled = false;

}

}

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

{

Requests category = new Requests();

this.Hide();

category.ShowDialog();

}

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

{

Employees category = new Employees();

this.Hide();

category.ShowDialog();

}

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

{

Bookings category = new Bookings();

this.Hide();

category.ShowDialog();

}

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

{

Comments category = new Comments();

this.Hide();

category.ShowDialog();

}

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

{

Statistic category = new Statistic();

this.Hide();

category.ShowDialog();

}

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

{

Users category = new Users();

this.Hide();

category.ShowDialog();

}

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

{

Environment.Exit(Environment.ExitCode);

}

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

{

Environment.Exit(Environment.ExitCode);

}

}

}

**Код формы Comments:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.ComponentModel.Design;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TravelPro

{

public partial class Comments : Form

{

private string connectionString = @"Data Source=adclg1; Initial catalog=!!!Бронин\_УП; Integrated Security=True";

private SqlConnection connection;

public Comments()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

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

{

LoadComments();

}

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

{

Main.Instance.Show();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

string dateString1 = dataGridView1.SelectedRows[0].Cells["CommentDate"].Value.ToString();

DateTime date1;

if (DateTime.TryParseExact(dateString1, "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out date1))

{

dateTimePicker1.Value = date1;

}

textBox1.Text = dataGridView1.SelectedRows[0].Cells["RequestID"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["EmployeeID"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["CommentText"].Value.ToString();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

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

{

int requestID = Convert.ToInt32(textBox1.Text);

int employeeID = Convert.ToInt32(textBox2.Text);

string commentText = textBox3.Text;

DateTime commentDate = dateTimePicker1.Value;

string query = "INSERT INTO Comments (RequestID, EmployeeID, CommentText, CommentDate) VALUES (@RequestID, @EmployeeID, @CommentText, @CommentDate)";

ExecuteNonQuery(query, ("@RequestID ", requestID), ("@EmployeeID", employeeID), ("@CommentText", commentText), ("@CommentDate", commentDate));

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int commentID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["CommentID"].Value);

string query1 = "DELETE FROM Comments WHERE CommentID = @CommentID";

ExecuteNonQuery(query1, ("@CommentID", commentID));

}

else

{

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

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int commentID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["CommentID"].Value);

int requestID = Convert.ToInt32(textBox1.Text);

int employeeID = Convert.ToInt32(textBox2.Text);

string commentText = textBox3.Text;

DateTime commentDate = dateTimePicker1.Value;

string query = "UPDATE Bookings SET RequestID = @RequestID, EmployeeID = @EmployeeID, CommentText = @CommentText, CommentDate = @CommentDate WHERE CommentID = @CommentID";

ExecuteNonQuery(query, ("@RequestID", requestID), ("@EmployeeID", employeeID), ("@CommentText", commentText), ("@CommentDate", commentDate), ("@CommentID", commentID));

}

else

{

MessageBox.Show("Выберите заказ для обновления.");

}

}

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

{

LoadComments();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadComments(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadComments()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT RequestID, EmployeeID, CommentText, CommentDate FROM Comments";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

}

**Код формы Bookings:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TravelPro

{

public partial class Bookings : Form

{

private string connectionString = @"Data Source=adclg1; Initial catalog=!!!Бронин\_УП; Integrated Security=True";

private SqlConnection connection;

public Bookings()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

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

{

Main.Instance.Show();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

string dateString1 = dataGridView1.SelectedRows[0].Cells["BookingDate"].Value.ToString();

DateTime date1;

if (DateTime.TryParseExact(dateString1, "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out date1))

{

dateTimePicker1.Value = date1;

}

textBox1.Text = dataGridView1.SelectedRows[0].Cells["RequestID"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["EmployeeID"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["BookingStatus"].Value.ToString();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

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

{

int requestID = Convert.ToInt32(textBox1.Text);

int employeeID = Convert.ToInt32(textBox2.Text);

DateTime bookingDate = dateTimePicker1.Value;

string bookingStatus = textBox3.Text;

string query = "INSERT INTO Bookings (RequestID, EmployeeID, BookingDate, BookingStatus) VALUES (@RequestID, @EmployeeID, @BookingDate, @BookingStatus)";

ExecuteNonQuery(query, ("@RequestID ", requestID), ("@EmployeeID", employeeID), ("@BookingDate", bookingDate), ("@BookingStatus", bookingStatus));

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int bookingID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["BookingID"].Value);

string query1 = "DELETE FROM Bookings WHERE BookingID = @BookingID";

ExecuteNonQuery(query1, ("@BookingID", bookingID));

}

else

{

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

}

}

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

{

if (dataGridView1.SelectedRows.Count > 0)

{

int bookingID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["BookingID"].Value);

int requestID = Convert.ToInt32(textBox1.Text);

int employeeID = Convert.ToInt32(textBox2.Text);

DateTime bookingDate = dateTimePicker1.Value;

string bookingStatus = textBox3.Text;

string query = "UPDATE Bookings SET RequestID = @RequestID, EmployeeID = @EmployeeID, BookingDate = @BookingDate, BookingStatus = @BookingStatus WHERE BookingID = @BookingID";

ExecuteNonQuery(query, ("@RequestID", requestID), ("@EmployeeID", employeeID), ("@BookingDate", bookingDate), ("@BookingStatus", bookingStatus), ("@BookingID", bookingID));

}

else

{

MessageBox.Show("Выберите заказ для обновления.");

}

}

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

{

LoadBookings();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

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

{

LoadBookings();

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadBookings(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadBookings()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT RequestID, EmployeeID, BookingDate, BookingStatus FROM Bookings";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

}

**Приложение Б**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data;

using System.Data.SqlClient;

using TravelPro;

namespace UnitTestProject1

{

[TestClass]

public class UnitTest1

{

private Form1 \_form;

[TestInitialize]

public void Initialize()

{

\_form = new Form1();

}

[TestMethod]

public void ValidateLoginInput\_ValidInput\_ReturnsTrue()

{

// Arrange

string login = "testlogin";

string password = "testpassword";

// Act

bool result = \_form.ValidateLoginInput(login, password);

// Assert

Assert.IsTrue(result);

}

[TestMethod]

public void ValidateLoginInput\_EmptyLogin\_ReturnsFalse()

{

// Arrange

string login = "";

string password = "testpassword";

// Act

bool result = \_form.ValidateLoginInput(login, password);

// Assert

Assert.IsFalse(result);

}

[TestMethod]

public void ValidateLoginInput\_EmptyPassword\_ReturnsFalse()

{

// Arrange

string login = "testlogin";

string password = "";

// Act

bool result = \_form.ValidateLoginInput(login, password);

// Assert

Assert.IsFalse(result);

}

[TestMethod]

public void ValidateLoginInput\_NullLogin\_ReturnsFalse()

{

// Arrange

string login = null;

string password = "testpassword";

// Act

bool result = \_form.ValidateLoginInput(login, password);

// Assert

Assert.IsFalse(result);

}

[TestMethod]

public void ValidateLoginInput\_NullPassword\_ReturnsFalse()

{

// Arrange

string login = "testlogin";

string password = null;

// Act

bool result = \_form.ValidateLoginInput(login, password);

// Assert

Assert.IsFalse(result);

}

[TestMethod]

public void AuthenticateUser\_ValidCredentials\_ReturnsUserRole()

{

// Arrange

string login = "testlogin";

string password = "testpassword"; // Ensure these credentials exist in your test database

// Act

string userRole = \_form.AuthenticateUser(login, password);

// Assert

Assert.IsNotNull(userRole);

}

[TestMethod]

public void AuthenticateUser\_InvalidCredentials\_ReturnsNull()

{

// Arrange

string login = "invalidlogin";

string password = "invalidpassword";

// Act

string userRole = \_form.AuthenticateUser(login, password);

// Assert

Assert.IsNull(userRole);

}

[TestMethod]

//[ExpectedException(typeof(SqlException))]

[ExpectedException(typeof(System.ArgumentException))]

public void AuthenticateUser\_DatabaseConnectionFailure\_ThrowsException()

{

// Arrange

// Simulate a connection failure by using an invalid connection string

\_form.connectionString = "Invalid Connection String";

string login = "testlogin";

string password = "testpassword";

// Act

\_form.AuthenticateUser(login, password);

}

[TestMethod]

public void AuthenticateUser\_EmptyCredentials\_ReturnsNull()

{

// Arrange

string login = "";

string password = "";

// Act

string userRole = \_form.AuthenticateUser(login, password);

// Assert

Assert.IsNull(userRole);

}

[TestMethod]

public void AuthenticateUser\_ValidCredentials\_ReturnsCorrectUserRole()

{

// Arrange

string login = "testlogin";

string password = "testpassword";

string expectedUserRole = "Мартышка";

// Act

string userRole = \_form.AuthenticateUser(login, password);

// Assert

Assert.AreEqual(expectedUserRole, userRole);

}

}

}

**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(а) 4 курса 42919/2 группы

Бронин Евгений 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.