**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

УТВЕРЖДАЮ

Зам. директора по УПР

С.Ю. Назаров\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_20\_\_\_г.

**ОТЧЕТ**

**по учебной практике (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) IV курса 42919/4 группы

Давыдов Даниил Денисович

(Фамилия, имя, отчество)

Место прохождения практики: Вычислительный центр Институт среднего профессионального образования, пр. Энгельса д.23

(наименование и адрес организации)

Период прохождения практики

с «6» ноября 2023 г. по «18» ноября 2023 г.

Руководитель практики Коннова А.Е.

(подпись) (расшифровка подписи)

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

М.П.

Санкт-Петербург

2023

**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) IV курса 42919/4 группы

Давыдов Даниил Денисович

(Фамилия, имя, отчество)

Место прохождения практики: Вычислительный центр Институт среднего профессионального образования, пр. Энгельса д.23

(наименование и адрес организации)

Период прохождения практики

с «6» ноября 2023 г. по «18» ноября 2023 г.

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

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

**Индивидуальное задание: ВАРИАНТ 27**

Задание выдал «6» ноября 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_ Коннова А.Е.

(подпись) (Ф.И.О.)

Задание получил «6» ноября 2023 г. \_ Давыдов Д. Д.(подпись) (Ф.И.О.)

СОГЛАСОВАНО

Председатель ПЦК

\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_ 202\_\_ г.

**Министерство науки и высшего образования Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики**

**(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) IV курса 42919/4 группы

Давыдов Даниил Денисович

(Фамилия, имя, отчество)

Место прохождения практики: Вычислительный центр Институт среднего профессионального образования, пр. Энгельса д.23

(наименование и адрес организации)

Период прохождения практики

с «6» ноября 2023 г. по «18» ноября 2023 г.

Руководитель практики Коннова А.Е.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

М.П.

Санкт-Петербург

2023

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 6.11.23 | Анализ предметной области. Выявление требований к программе. |  |
| 7.11.23 | Разработка технического задания. UML. Проектирование диаграммы вариантов использования. |  |
| 8.11.23 | UML. Проектирование диаграммы последовательности. UML. Проектирование диаграммы активности. |  |
| 9.11.23 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 10.11.23 | Проектирование интерфейса пользователя. Создание Wireframe эскизов. Разработка дизайна программы в соответствии с руководством по стилю. |  |
| 11.11.23 | Разработка базы данных. Разработка словаря данных. |  |
| 13.11.23 | Создание приложения. Форма авторизации. Создание приложения. Форма заказов. |  |
| 14.11.23 | Создание приложения. Основные формы приложения. Разработка библиотеки классов. |  |
| 15.11.23 | Подготовка отчетов и выгрузка документов для печати. Отладка программных модулей. |  |
| 16.11.23 | Модульное тестирование. Создание тестовых случаев. |  |
| 17.11.23 | Интеграционное тестирование. Разработка самодокументирующегося кода. |  |
| 18.11.23 | Инспекция кода на соответствие стандартам кодирования. Подготовка отчета и размещение результатов в репозитории контроля версий. |  |

**Содержание**

[ВВЕДЕНИЕ 7](#_Toc151202750)

[1. Импорт данных в БД 8](#_Toc151202751)

[2. Общая часть 15](#_Toc151202752)

[2.1. Проектирование системы 15](#_Toc151202753)

[2.1.1. ER-модель БД 15](#_Toc151202754)

[2.2. Создание базы данных и заполнение таблиц данными 24](#_Toc151202755)

[2.3. Разработка библиотеки и подключение её к проекту 26](#_Toc151202756)

[2.4. Разработка приложения 26](#_Toc151202757)

[2.5. Тестирование приложения 40](#_Toc151202758)

[2.6. Выгрузка готового проекта в репозиторий Git 41](#_Toc151202759)

[Заключение 42](#_Toc151202760)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc151202761)

[ПРИЛОЖЕНИЕ А 44](#_Toc151202762)

[ПРИЛОЖЕНИЕ Б 45](#_Toc151202763)

[ПРИЛОЖЕНИЕ В 46](#_Toc151202764)

[ПРИЛОЖЕНИЕ Г 63](#_Toc151202765)

[ПРИЛОЖЕНИЕ Д 67](#_Toc151202766)

# **ВВЕДЕНИЕ**

В рамках учебной практики передо предстоит продемонстрировать теоретические знания в области баз данных и практические навыки работы с ними. Этот процесс включает в себя изучение различных аспектов создания и оптимизации баз данных, разработки SQL-запросов для эффективного извлечения данных из больших объемов информации. В процессе работы необходимо изучить предметную область проекта, адаптировать базу данных под конкретные потребности.

Одним из ключевых этапов проекта является создание диаграмм баз данных, которые помогают визуализировать структуру данных и взаимосвязи между таблицами, что является важным шагом для понимания архитектуры базы данных и оптимизации ее структуры.

В заключение работы выполнения учебной практики предстоит разработать приложение на языке программирования C# для взаимодействия с базой данных, которое будет не только функциональным, но и иметь удобный и интуитивно понятный для пользователей интерфейс.

# **Импорт данных в БД**

ER-модель

Изображение выглядит как текст, диаграмма, План, Параллельный

Автоматически созданное описание

Рисунок 1 - ER-модель

Преобразование таблицы users.

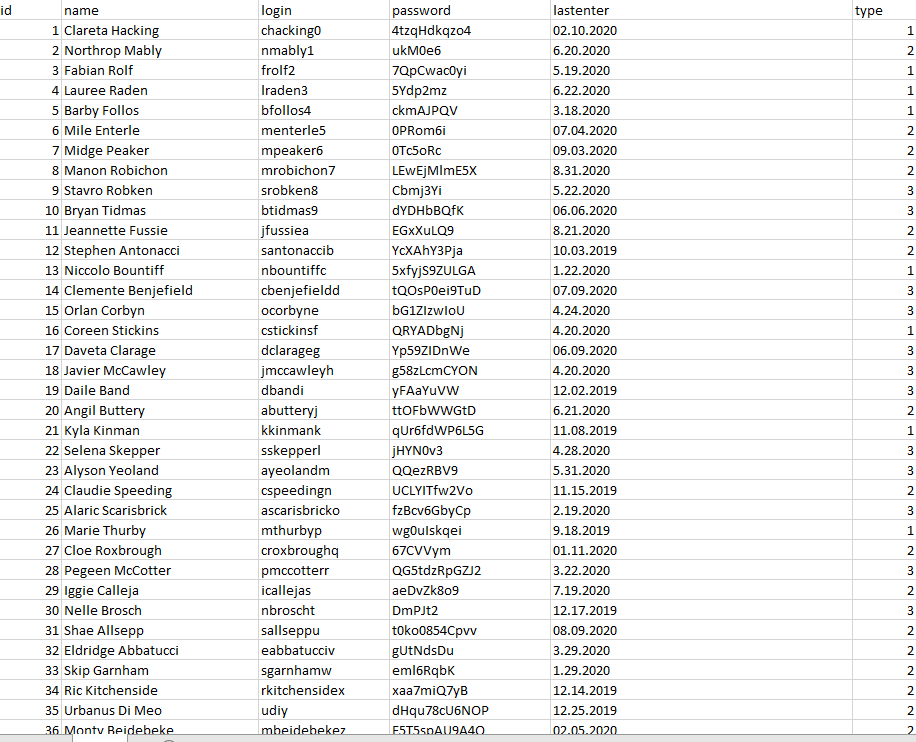


Рисунок 2 - Преобразованная таблица users

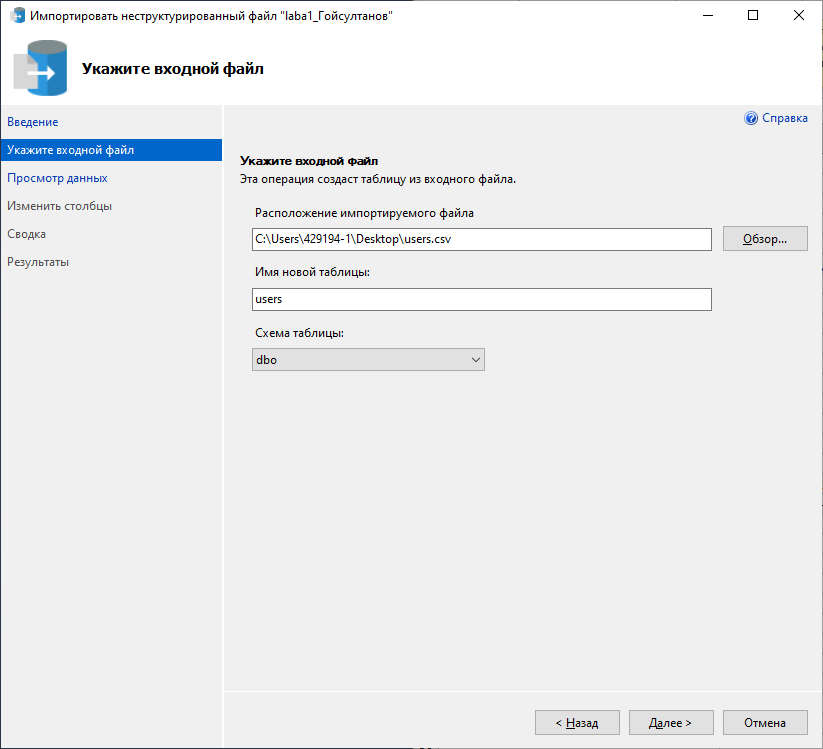


Рисунок 3 - Импорт таблицы users



Рисунок 4 - Просмотр данных таблицы users

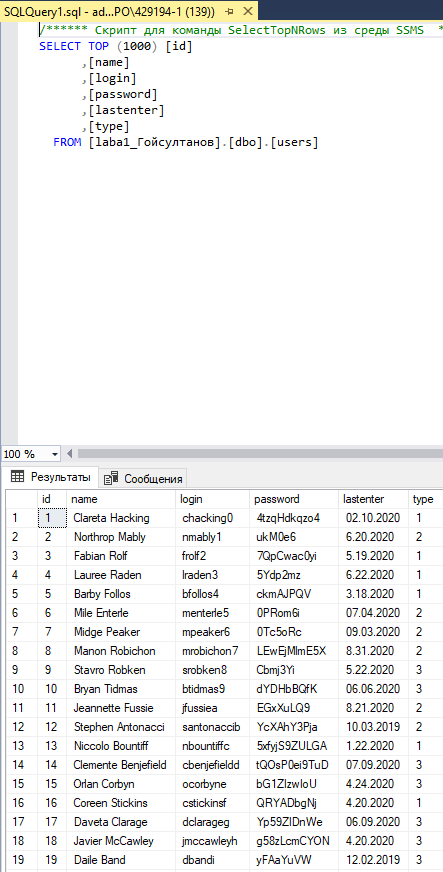


Рисунок 5 - Результат импортирования таблицы users

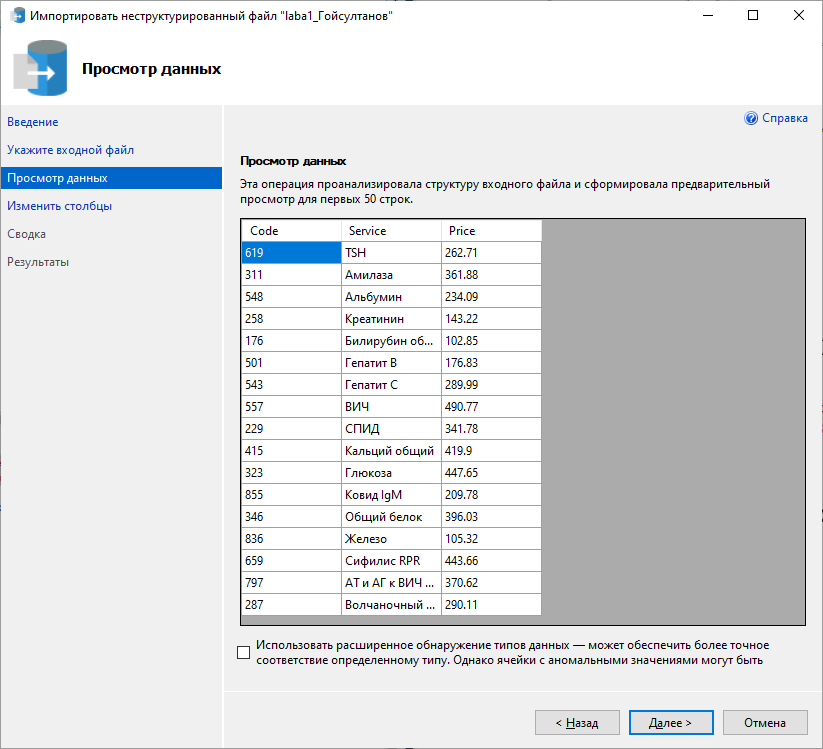


Рисунок 6 - Импортирования таблицы services



Рисунок 7 - Результат импортирования таблицы services



Рисунок 8 - Коды услуг из таблицы users

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 9 - Создание скрипта для преобразование файла user.txt

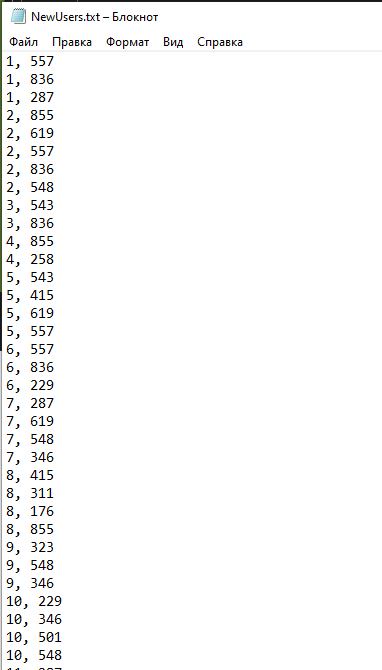


Рисунок 10 - Результат работы скрипта

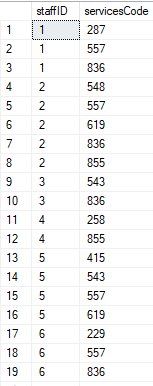


Рисунок 11 - Результат импортирования файла NewUsers.txt



Рисунок 12 - Преобразование файла patients.xml в формат .csv

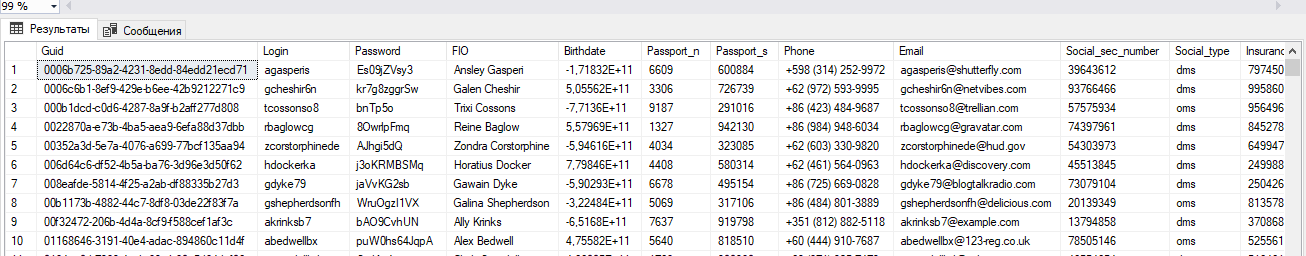


Рисунок 13 - Результат импортирования файла patients

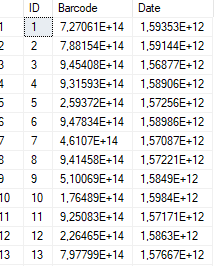


Рисунок 14 - Импортирование данных из файла blood.xml

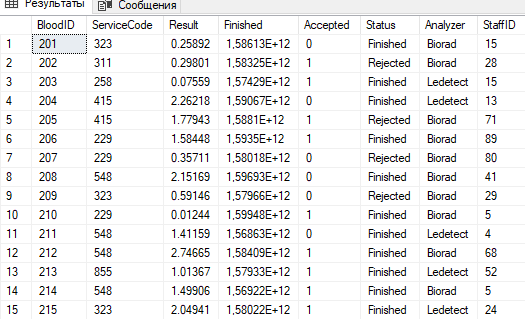


Рисунок 15 - Импортирование данных из файла blood\_services.xml

# **Общая часть**

* 1. **Проектирование системы**

Предметная область базы данных «Авиалинии» связана с составлением маршрутов, рейсов и назначении на них авиасудов. В данной предметной области оптимизация ПО должна решать ряд задач для обеспечения эффективного управления и улучшения производительности бизнес-процессов.

Мной была спроектирована диаграмму отношений сущностей, затрагивающую основные сущности в базе данных и их взаимосвязи.

* + 1. ER-модель БД

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 16 – ER-модель БД

* + 1. Словарь данных

Таблица 1 – Данные сущности «КомандирСудна»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **table\_nm** | **column\_nm** | **keys** | **data\_typ** | **nullable** |
| КомандирСудна | ИН | PK | int(10) | NOT NULL |
| КомандирСудна | ФИО |  | nvarchar(50) | NOT NULL |
| КомандирСудна | АДРЕС |  | nvarchar(50) | NULL |
| КомандирСудна | ТЕЛЕФОН |  | nvarchar(50) | NULL |
| КомандирСудна | НАЛЁТ |  | int(10) | NOT NULL |

Таблица 2 – Данные сущности «Авиасудна»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **table\_nm** | **column\_nm** | **keys** | **data\_typ** | **nullable** |
| Авиасудна | БОРТ № | PK | int(10) | NOT NULL |
| Авиасудна | МОДЕЛЬ |  | nvarchar(50) | NOT NULL |
| Авиасудна | ДАТА ИЗГОТОВЛЕНИЯ |  | date(0) | NOT NULL |
| Авиасудна | НАЛЁТ |  | int(10) | NOT NULL |
| Авиасудна | ГОТОВНОСТЬ |  | bit(1) | NOT NULL |

Таблица 3 – Данные сущности «Пассажиры»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **table\_nm** | **column\_nm** | **keys** | **data\_typ** | **nullable** |
| Пассажиры | НОМЕР ПАСПОРТА | PK | int(10) | NOT NULL |
| Пассажиры | № РЕЙСА | FK | int(10) | NOT NULL |
| Пассажиры | ФИО |  | nvarchar(50) | NOT NULL |
| Пассажиры | АДРЕС |  | nvarchar(50) | NULL |
| Пассажиры | ТЕЛЕФОН |  | nvarchar(50) | NULL |

Таблица 4 – Данные сущности «Рейс»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **table\_nm** | **column\_nm** | **keys** | **data\_typ** | **nullable** |
| Рейс | № РЕЙСА | PK | int(10) | NOT NULL |
| Рейс | № МАРШРУТА | FK | int(10) | NOT NULL |
| Рейс | БОРТ № | FK | int(10) | NOT NULL |
| Рейс | ДАТА/ВРЕМЯ ВЫЛЕТА |  | datetime (0) | NOT NULL |
| Рейс | СТАТУС |  | bit(1) | NOT NULL |

Таблица 5 – Данные сущности «Маршрут»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **table\_nm** | **column\_nm** | **keys** | **data\_typ** | **nullable** |
| Маршрут | НОМЕР | PK | int(10) | NOT NULL |
| Маршрут | ОТБЫТИЕ |  | nvarchar(50) | NOT NULL |
| Маршрут | ПРИБЫТИЕ |  | nvarchar(50) | NOT NULL |
| Маршрут | ЦЕНА |  | int(10) | NOT NULL |
| Маршрут | ДЛИТЕЛЬНОСТЬ |  | int(10) | NOT NULL |

* + 1. Диаграмма вариантов использования

Диаграмма прецендентов – диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Мной была разработана диаграмма прецедентов информационной системы (Рисунок 17).



Рисунок 17 – Диаграмма прецедентов

* + 1. Диаграмма активности

Диаграмма активности в UML (Unified Modeling Language) используется для визуализации потока управления и последовательности действий в системе или процессе. Она отображает последовательность шагов, решений и условий в виде узлов (активностей), связанных стрелками, показывающими порядок выполнения. Диаграммы активности позволяют моделировать бизнес-процессы, алгоритмы, их ветвления и условия, что делает их полезными для анализа и оптимизации рабочих процессов в различных областях, включая разработку программного обеспечения и управление проектами.

Мной разработана диаграмма активности для процесса (Рисунок 18).



Рисунок 18 – Диаграмма активности

* + 1. Диаграмма последовательности

Диаграмма последовательности – это тип UML-диаграммы, представляющий взаимодействие между объектами в определенной последовательности. Она показывает, как объекты в системе взаимодействуют друг с другом и какие сообщения они обмениваются в определенном порядке выполнения операций. Диаграмма последовательности отображает активности и сообщения между объектами во временной последовательности, что помогает понять логику работы системы и улучшить ее проектирование.

Мной разработана диаграммы последовательности для процесса авторизации в системе и для процесса добавления сотрудника в базу данных (Рисунок 19).



Рисунок 19 – Диаграмма последовательностей

* + 1. Разработка макетов интерфейса системы

Разработка макетов интерфейса системы включает в себя создание визуальных концепций пользовательского интерфейса, которые визуализируют структуру и взаимодействие элементов на экране. Этот процесс включает в себя создание макетов страниц, распределение компонентов, выбор цветовой схемы, шрифтов и графики, что позволяет разработчикам и заказчикам лучше представить себе внешний вид и функциональность будущей системы. Основная цель разработки макетов – обеспечить удобство использования, интуитивно понятный интерфейс и улучшить пользовательский опыт взаимодействия с системой.

Макет формы «Администратор» в двух видах: wireframes (Рисунок 20) и mockups (Рисунок 21).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 20 – Макет формы администратора(wireframes)

Изображение выглядит как текст, снимок экрана, дисплей, диаграмма

Автоматически созданное описание

Рисунок 21 – Макет формы администратора(mockups)

Макет формы «Окно заказа билетов» в двух видах: **wireframes (Рисунок 22) и mockups (Рисунок 23)**.

Изображение выглядит как текст, программное обеспечение, снимок экрана, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 22 – Макет формы заказа билетов(wireframes)

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 23 – Макет формы регистрации нового работника(mockups)

Макет формы «Вход» в двух видах: **wireframes (Рисунок 24) и mockups** (Рисунок 25).

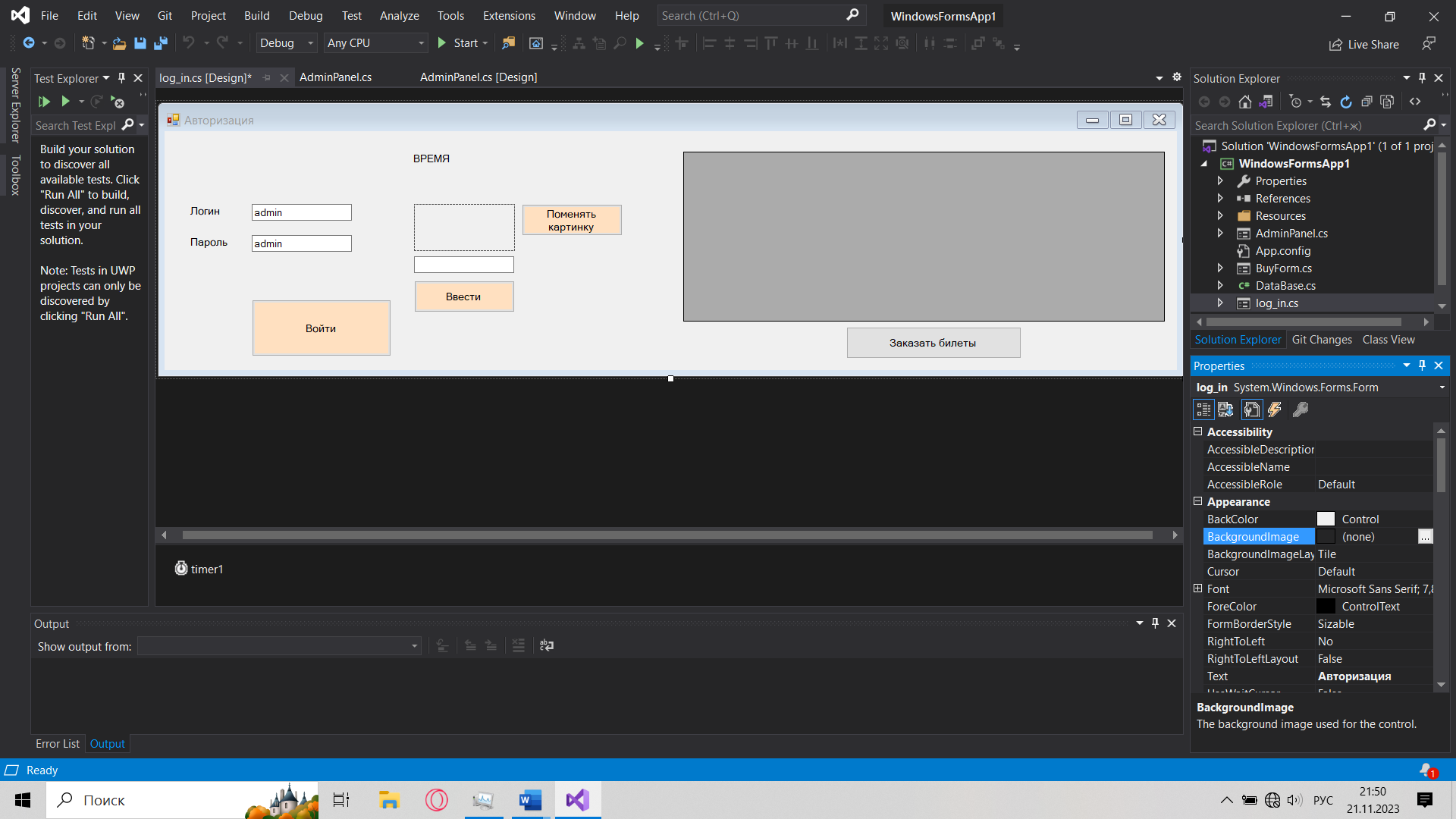


Рисунок 24 – Макет формы входа(wireframes)

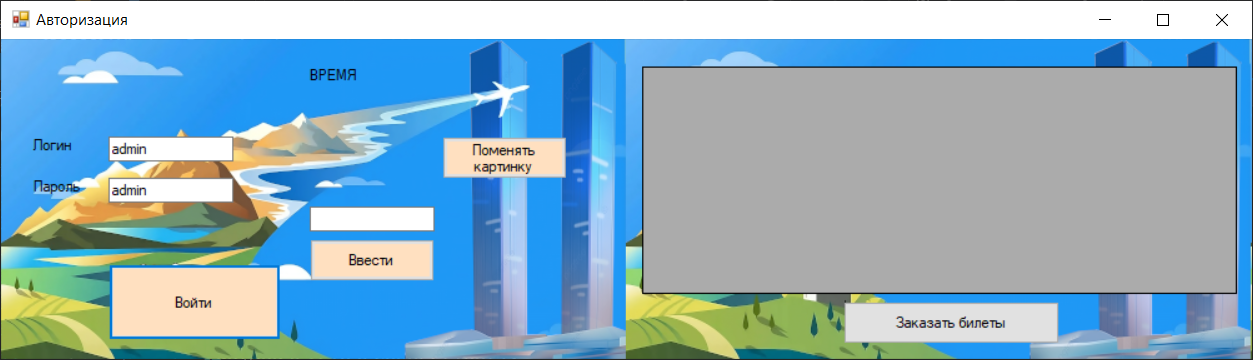


Рисунок 25 – Макет формы входа

* 1. **Создание базы данных и заполнение таблиц данными**

Скрипт для таблиц базы данных находится в приложении А.

Результат заполнения таблиц данными (Рисунок 26-30).

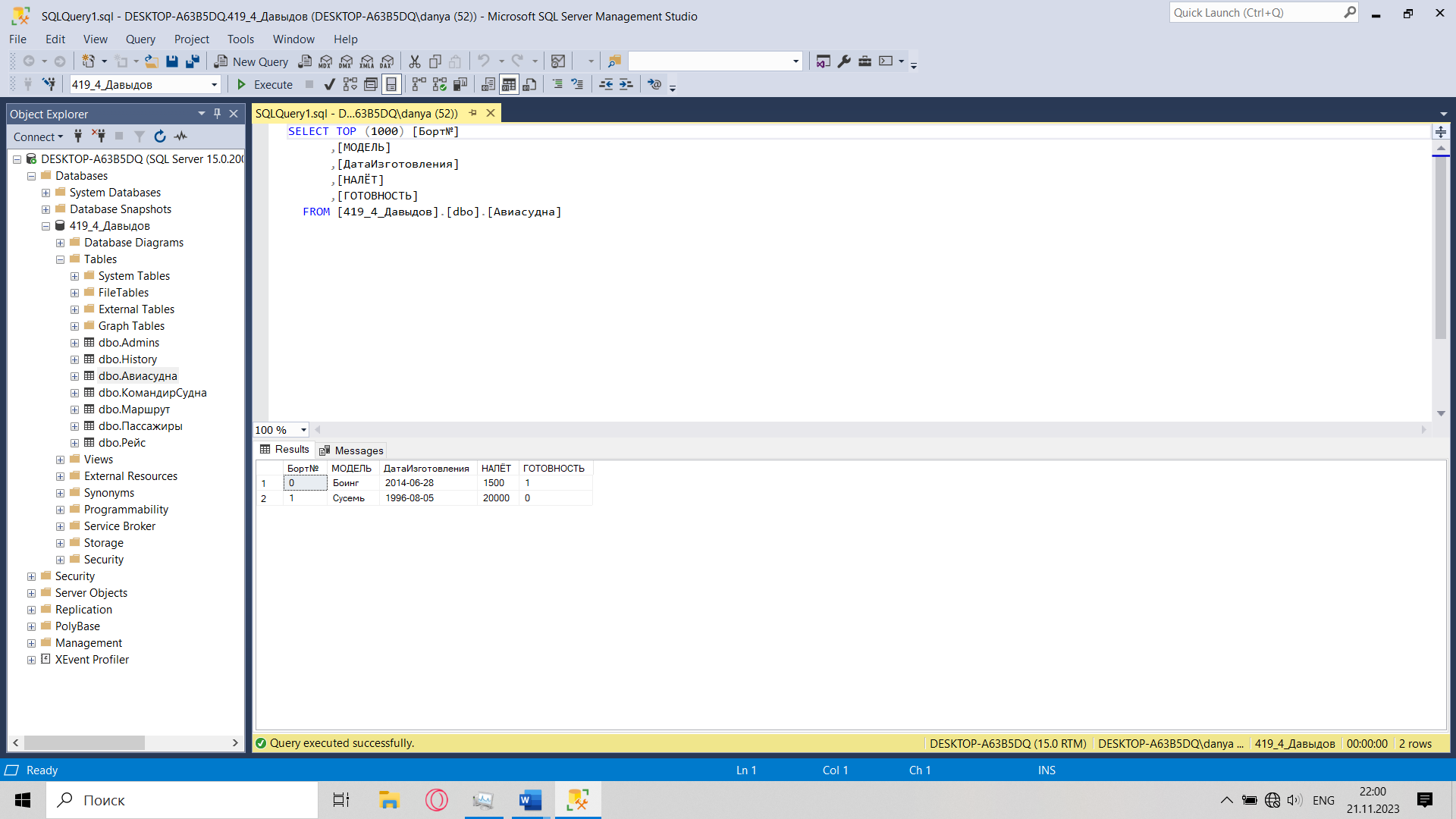


Рисунок 26 – Таблица «Авиасудна»

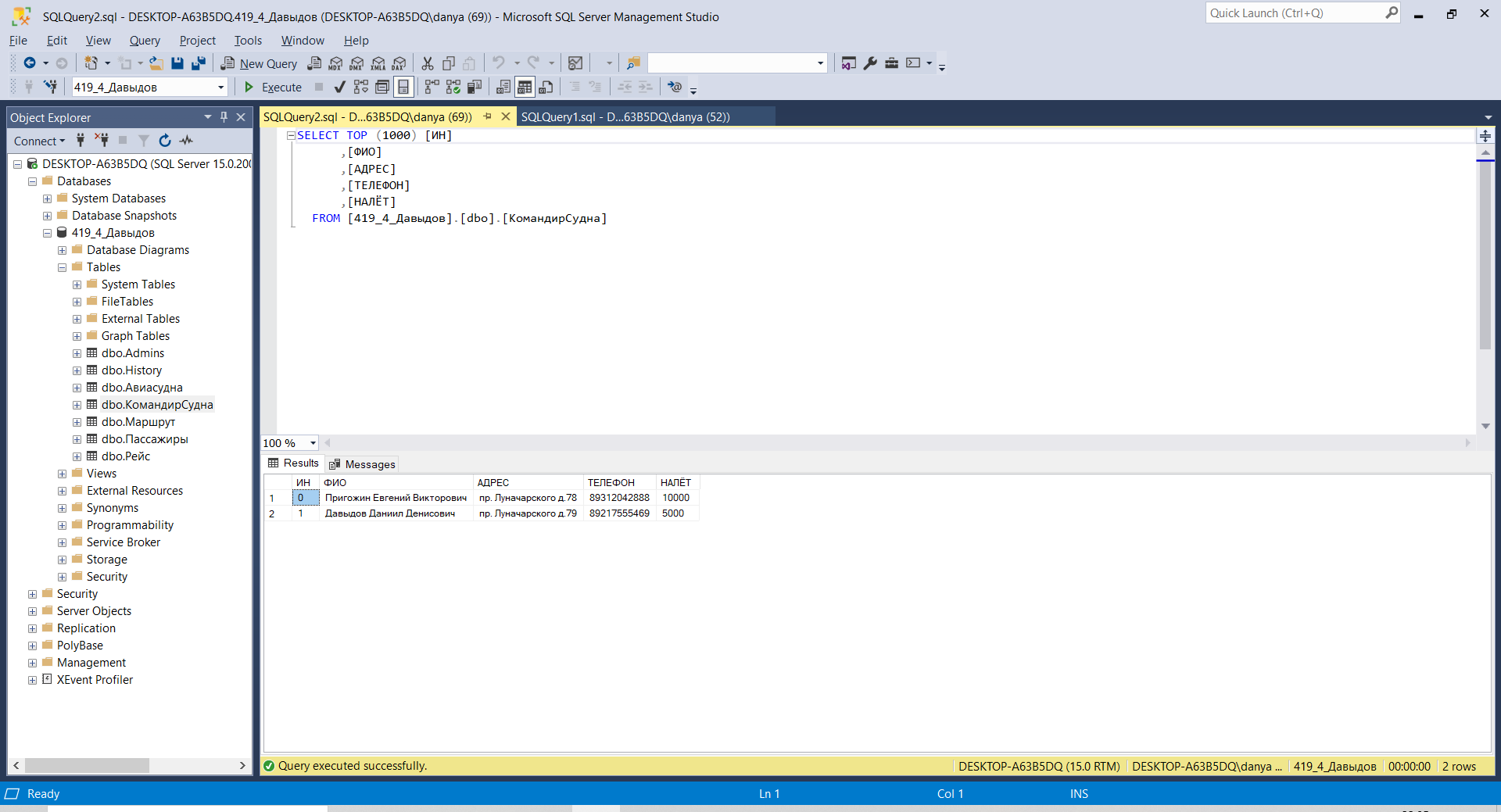


Рисунок 27 – Таблица «КомандирСудна»

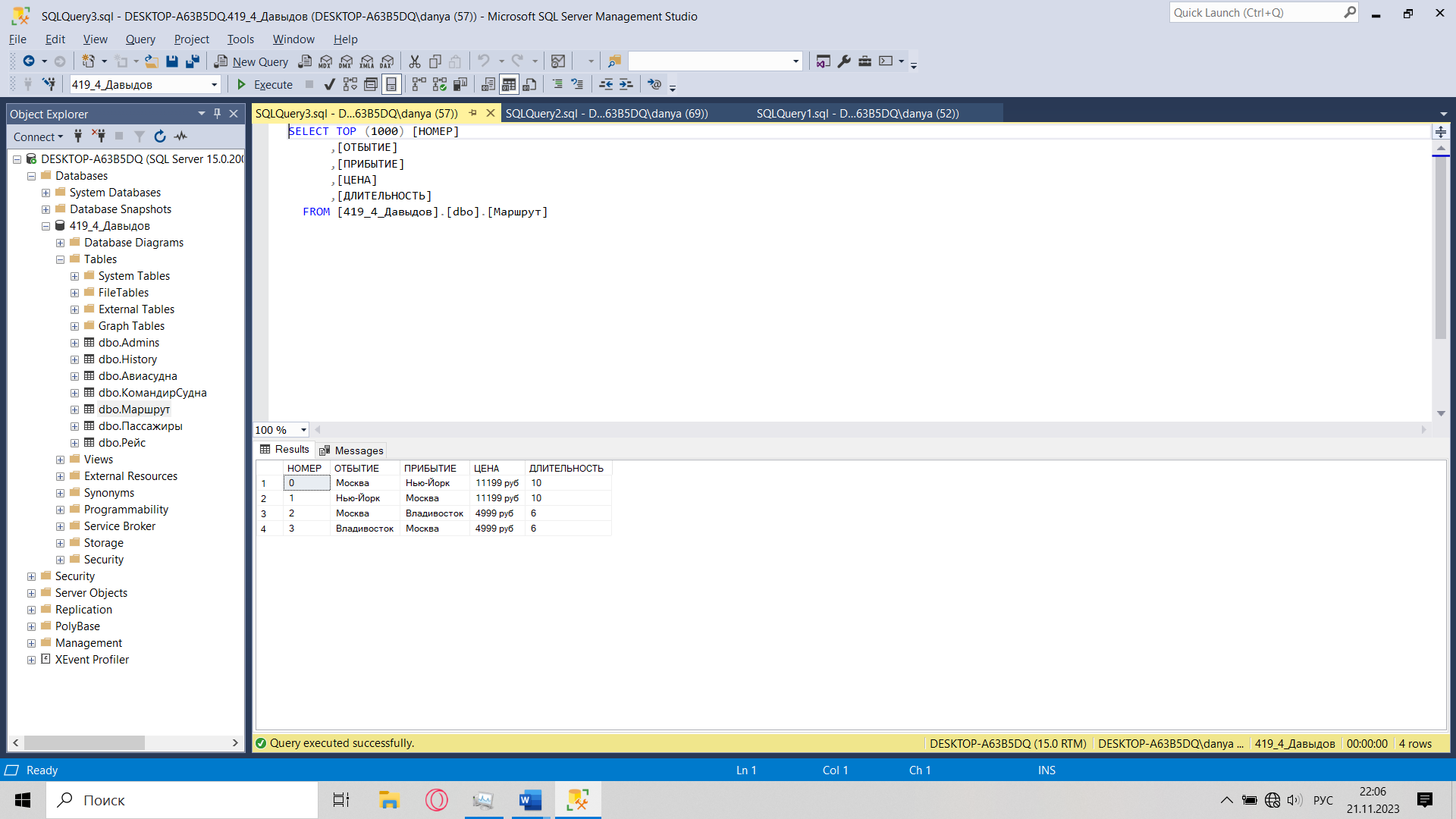


Рисунок 28 – Таблица «Маршрут»

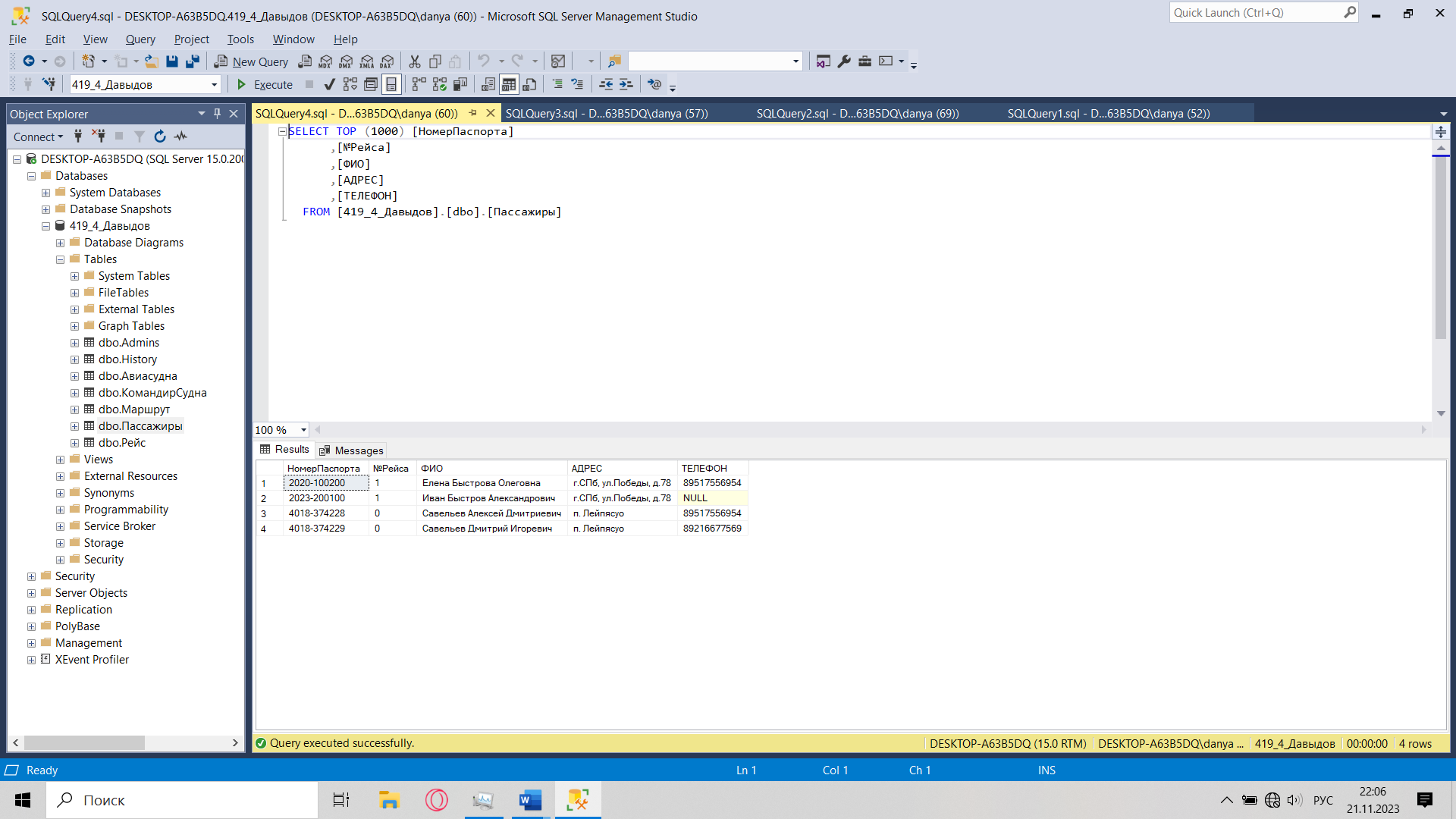


Рисунок 29 – Таблица «Пассажиры»

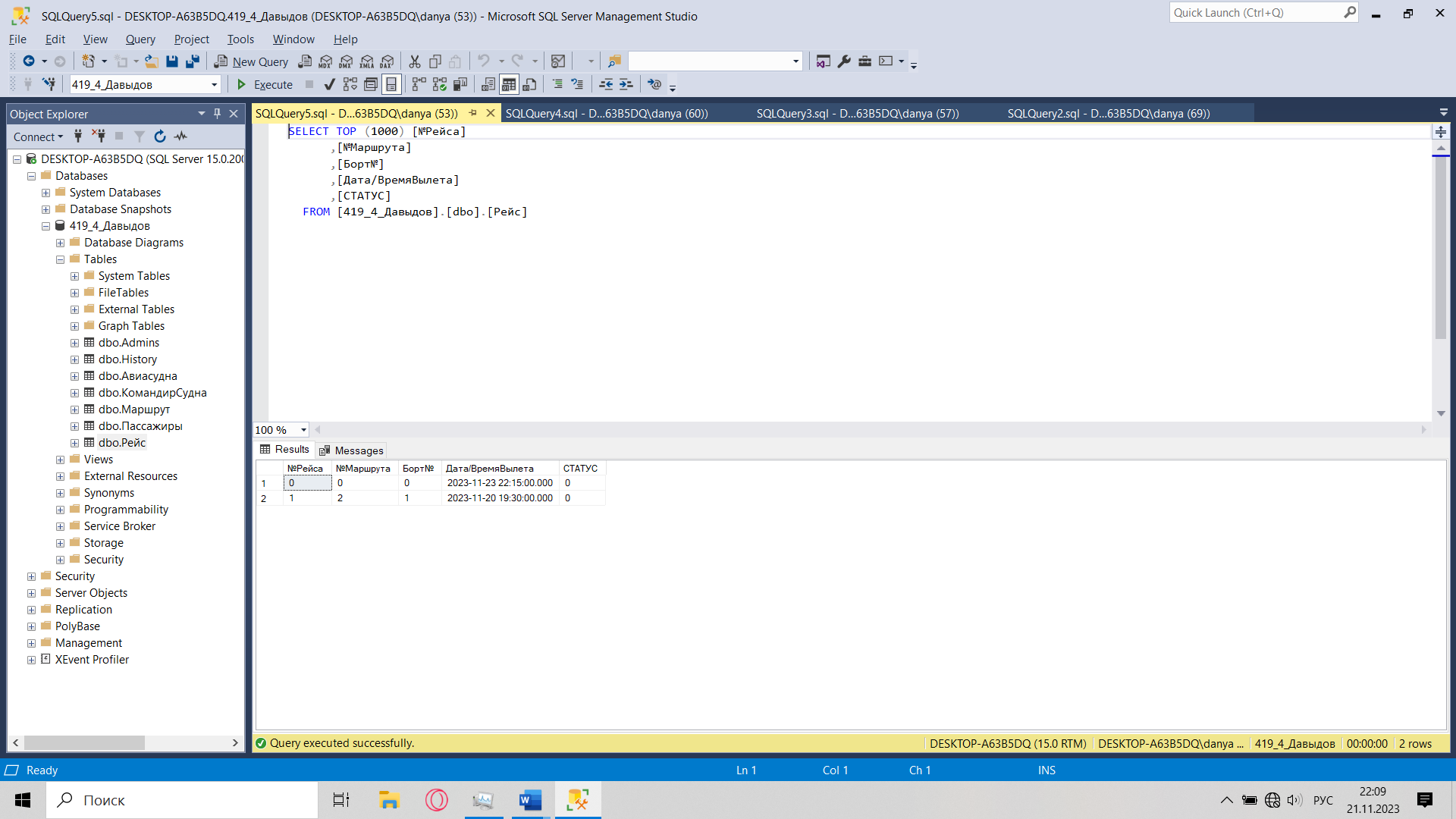


Рисунок 30 – Таблица «Рейс»

* 1. **Разработка библиотеки и подключение её к проекту**

Создание библиотеки

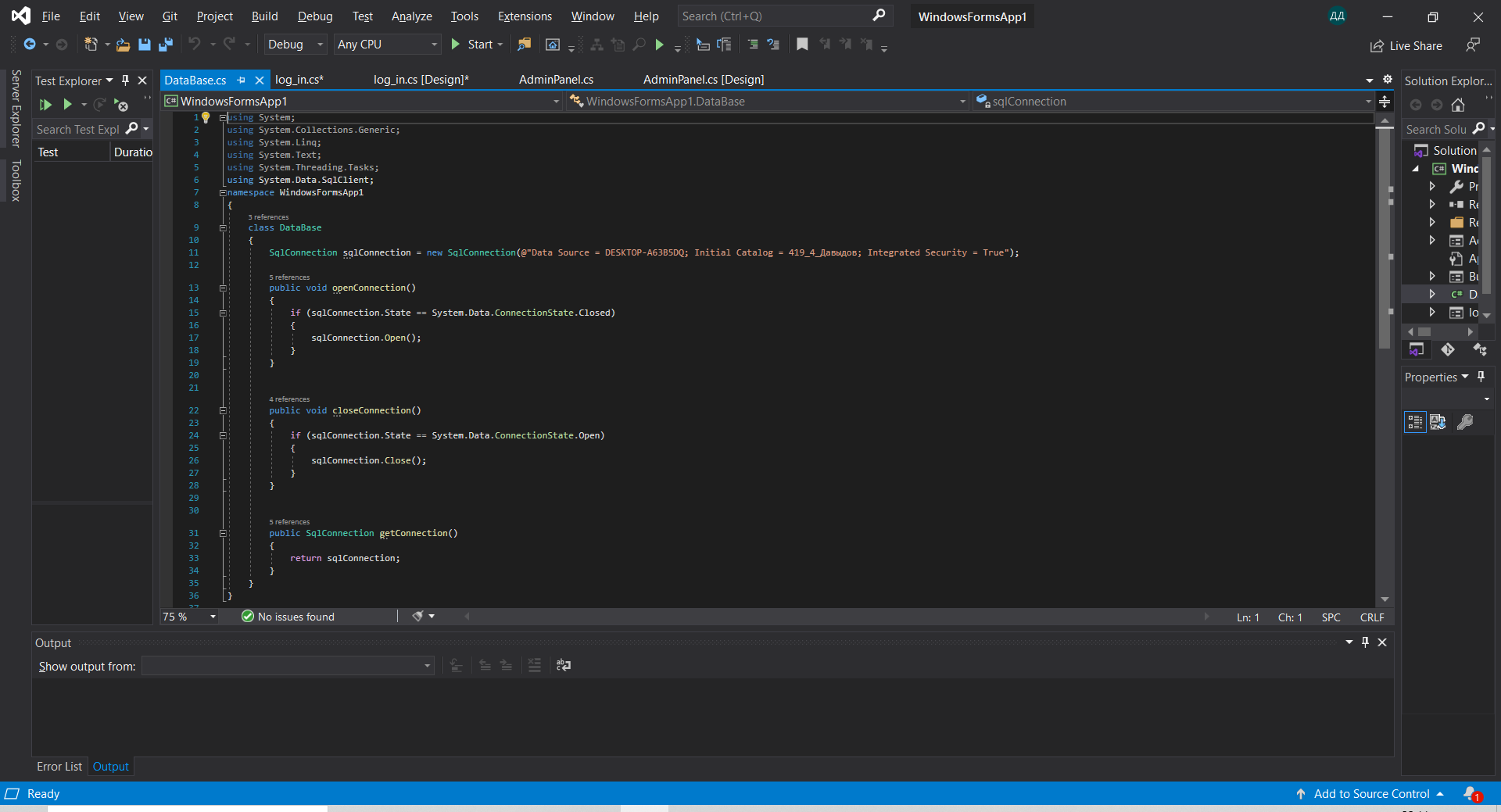


Рисунок 31 – Создание библиотеки для работы с БД

Код библиотеки находится в приложении Б.

* 1. **Разработка приложения**

Код приложения находится в приложении В.

При запуске программы открывается окно авторизации (Рисунок 32). При вводе пароль скрыт маской. При нажатии кнопки «Войти» данные проверяются на валидность. Так же здесь можно посмотреть доступные для покупки рейсы и кнопка заказа билетов.

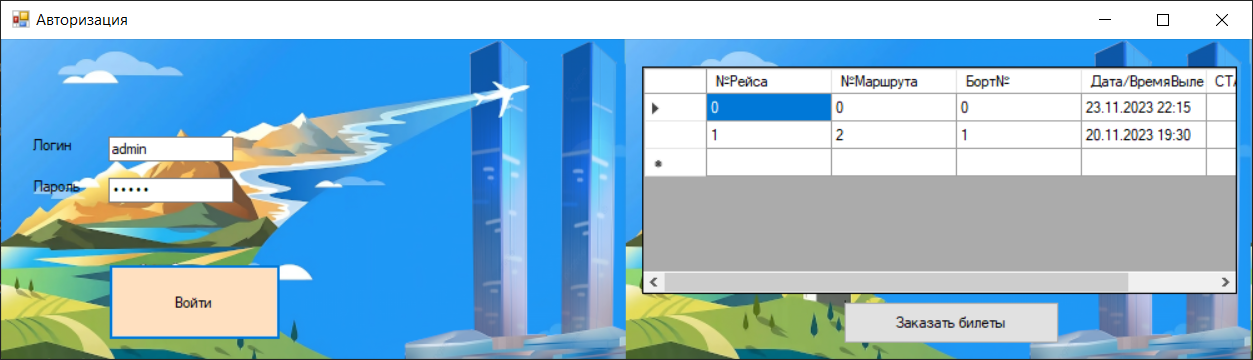


Рисунок 32 – Окно авторизации

При неудачной попытке авторизации система выводит сообщение «Не получилось». После этого пользователю необходимо ввести капчу состоящей из 4 символов и поля для ее ввода (Рисунок 33).

Изображение выглядит как текст, Графическое программное обеспечение, Мультимедийное программное обеспечение, программное обеспечение

Автоматически созданное описание

Рисунок 33 – Ввод капчи

При неправильном вводе капчи система блокируется на 180 секунд. Все элементы формы становятся недоступны (Рисунок 32).

Изображение выглядит как Графическое программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 34 – Блокировка системы

После верного ввода капчи пользователю предоставляется возможность повторно ввести логин и пароль. После успешной авторизации открывается главная форма (Рисунок 35).

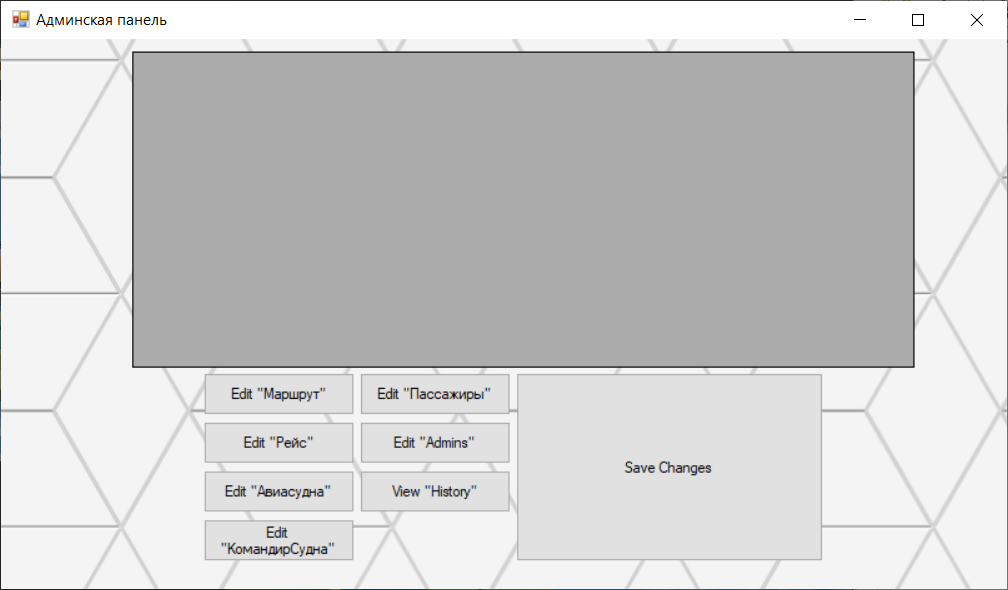


Рисунок 35 – Главная форма администратора

На главной форме есть 7 кнопок с названиями таблиц и 1 кнопка сохранения изменений.

При нажатии на кнопку «View History» открывается форма для просмотра истории входа в приложение (Рисунок 36)

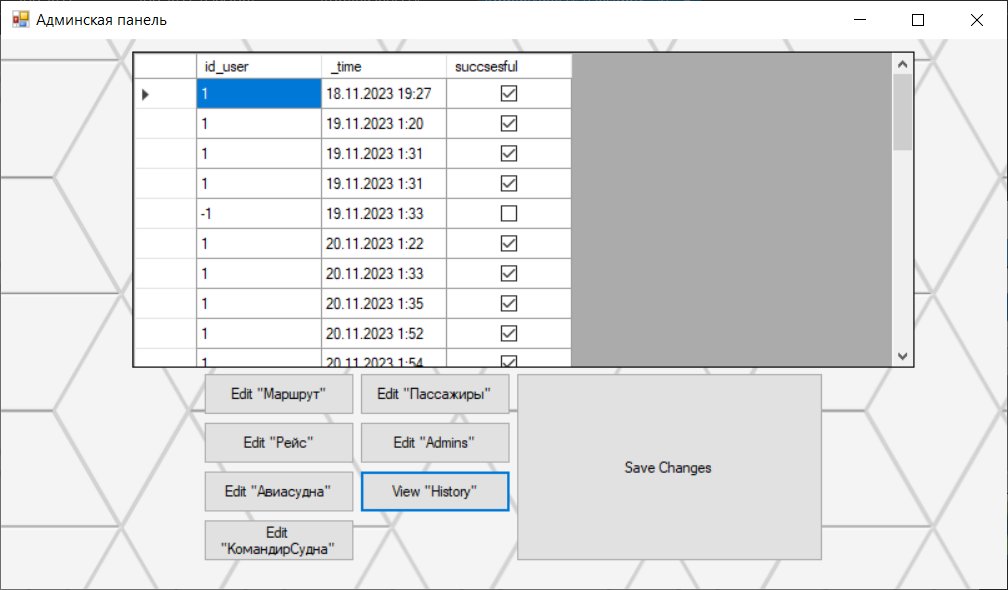


Рисунок 36 – Просмотр истории входа

При удалении строки и нажатии кнопки «Save Changes» удаляется выбранная запись из таблицы (Рисунок 37).

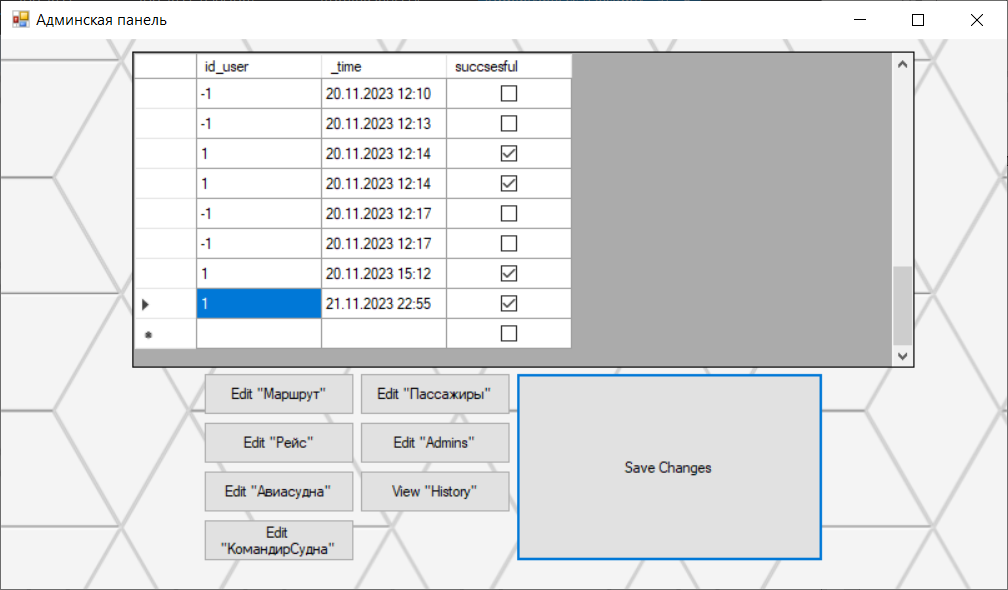


Рисунок 37 – Удаление записи из таблицы «History»

При нажатии на кнопку «Edit Admins» открывается форма для просмотра списка пользователей (Рисунок 38).

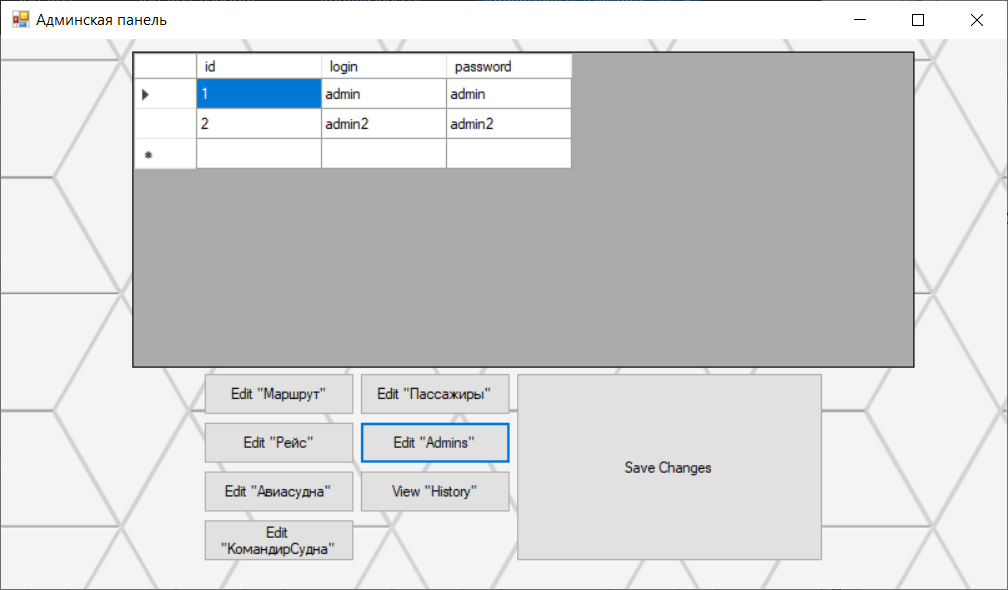


Рисунок 38 – Просмотр списка пользователей

На форме находятся есть лишь 1 кнопка: «Save Changes» – она сохраняет все изменения внесённые в dataGridWiev1 и это относиться к любой вызванной таблице: создание записи (Рисунок 39), удаление записи (Рисунок 40), изменение записи (Рисунок 41). Всё это сохраняется в БД (Рисунок 42– 43).

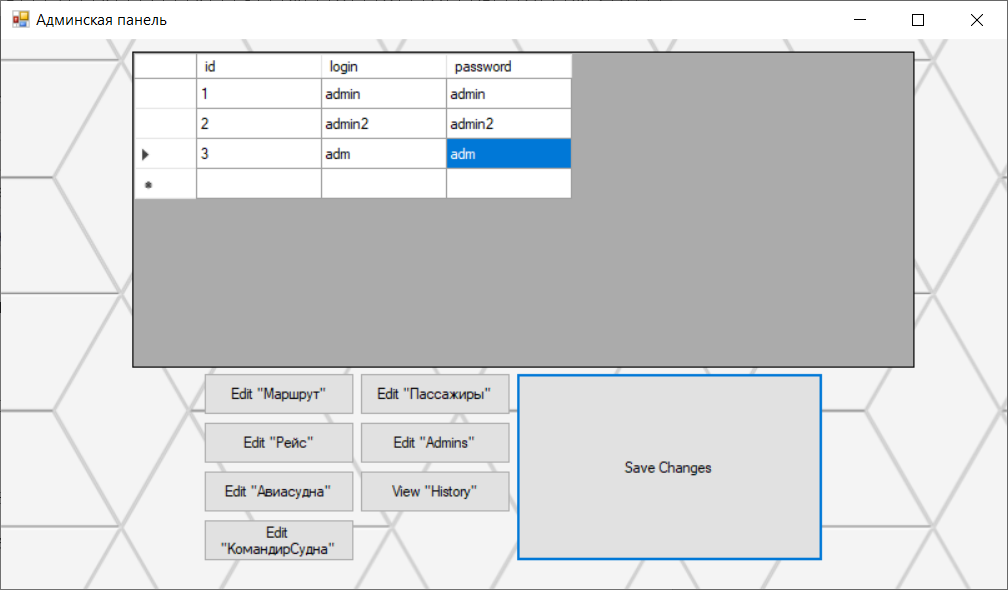


Рисунок 39 – Добавление нового работника

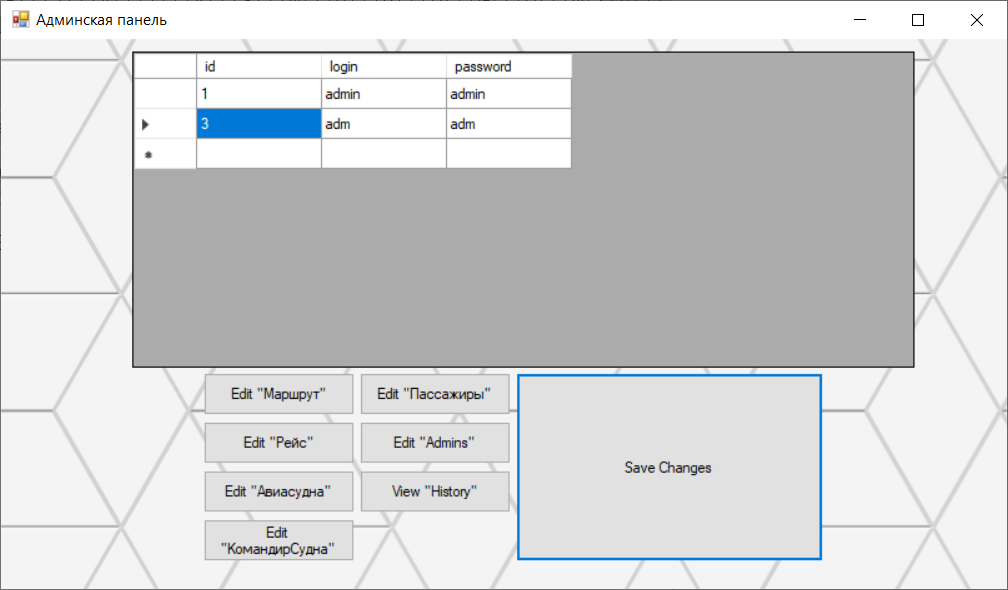


Рисунок 40 – Удаление записи

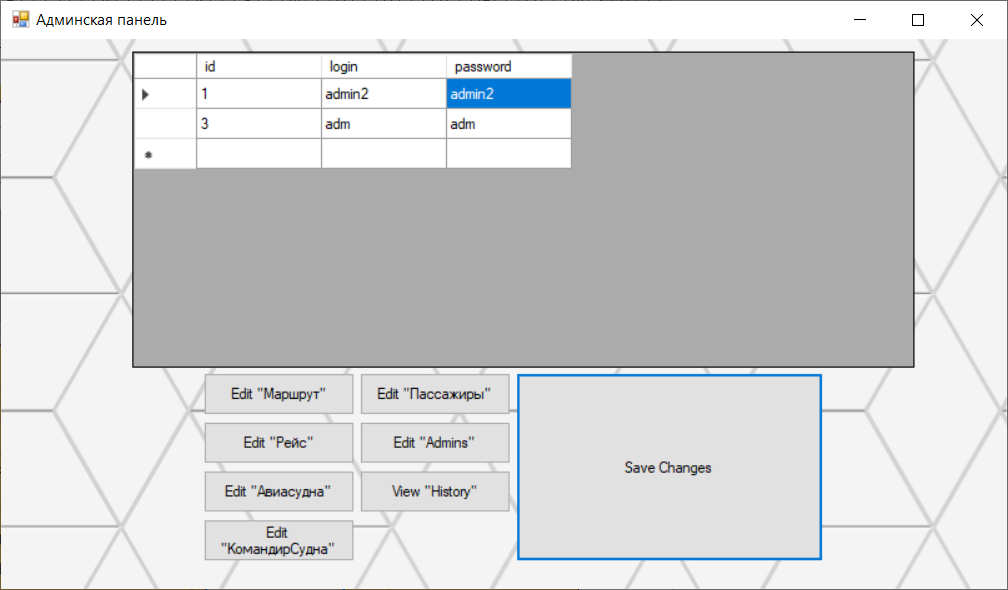


Рисунок 41 – Изменение записи из таблицы

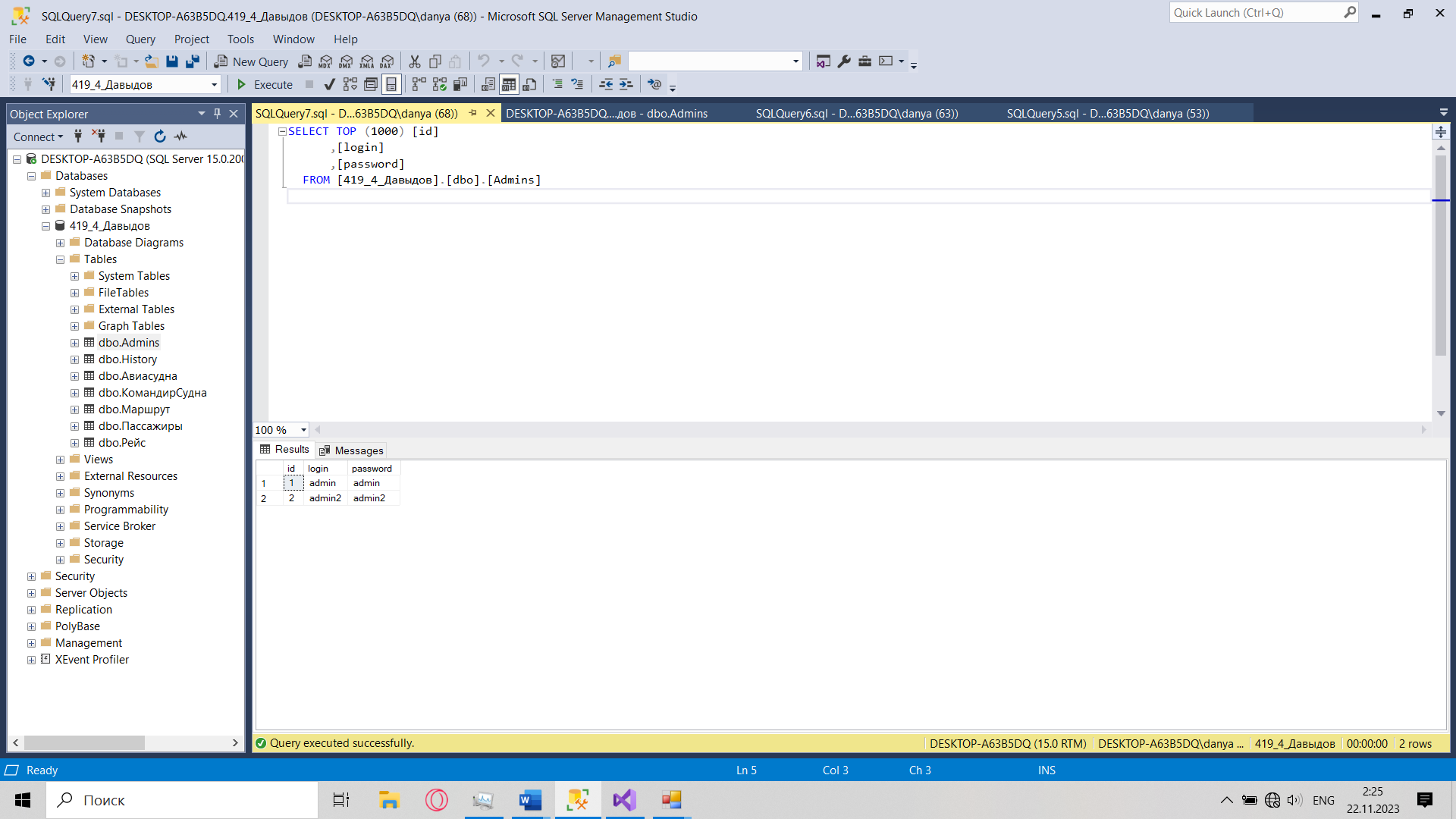


Рисунок 42 – Сохранение записи в БД до изменений

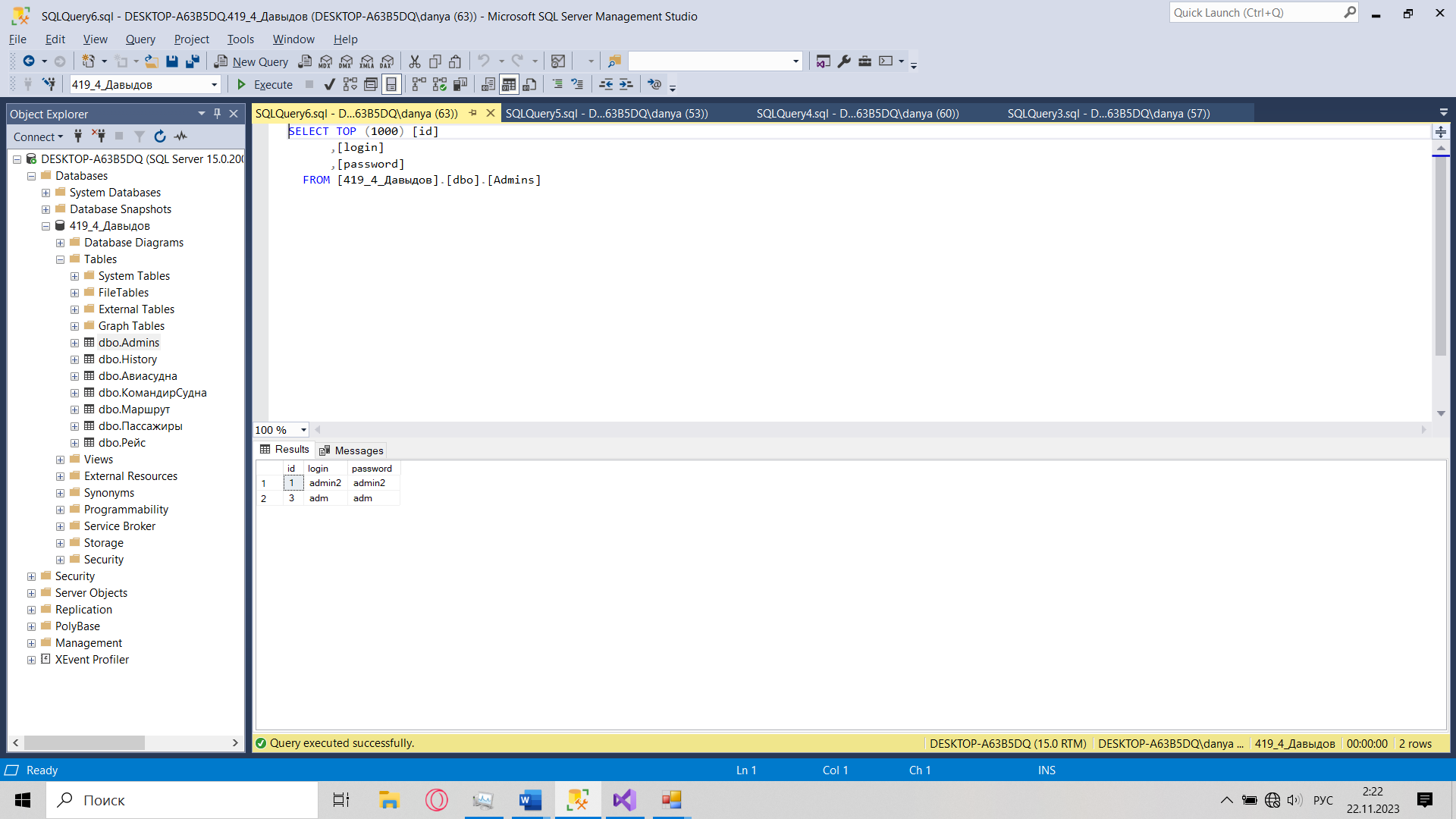


Рисунок 43 – Сохранение записи в БД после изменений

Также на форме входа есть кнопка заказа билетов. Она открывает соответствующую форму (Рисунок 44 – 45).

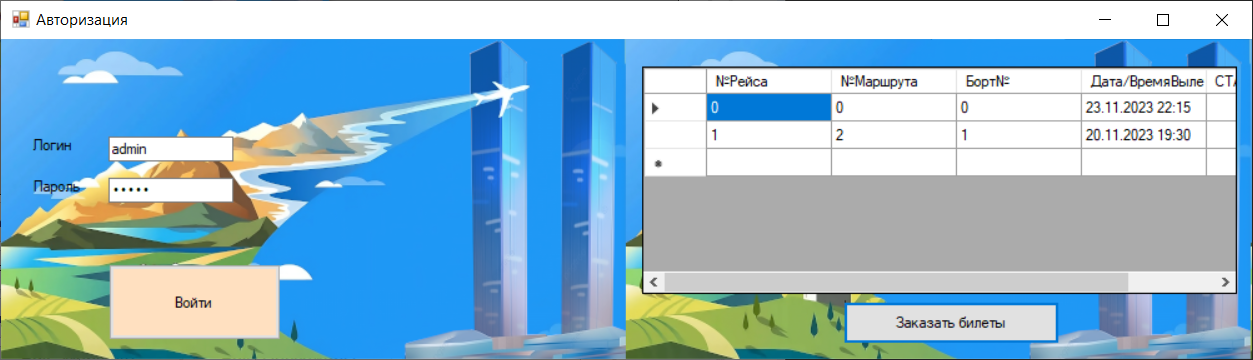


Рисунок 44 – Кнопка вызова формы



Рисунок 45 – Форма заказа билетов

На форме находится 1 кнопка: «Заказать билет» – добавление новой записи в таблицу «Пассажиры» (Рисунок 46 – 47)

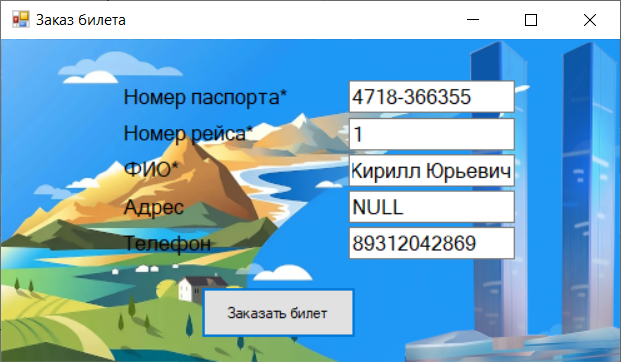


Рисунок 46 – Форма добавления новой записи

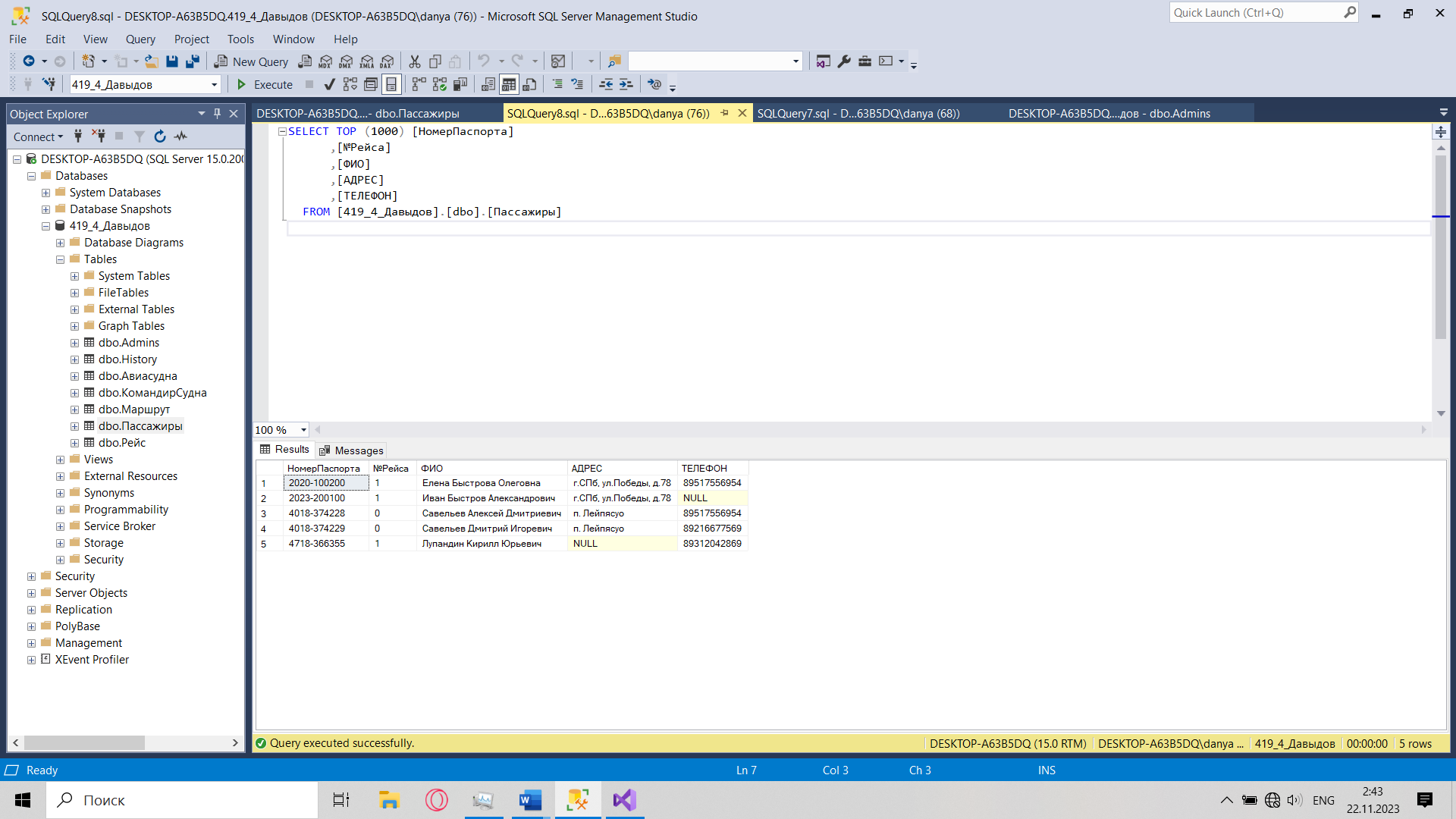


Рисунок 47 – Результат добавления новой записи

* 1. **Тестирование приложения**
     1. Создание тестовых случаев

Тестовые случае представлены в приложении Г.

* + 1. Модульное тестирование

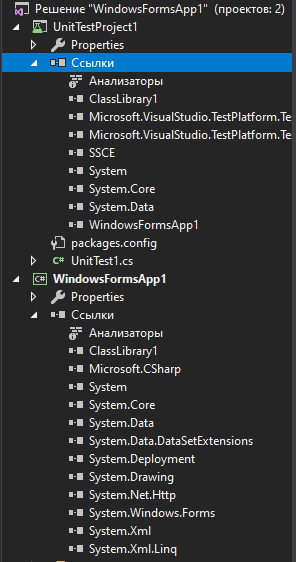


Рисунок 48 – Создание проекта UnitTest, добавление ссылки на библиотеку

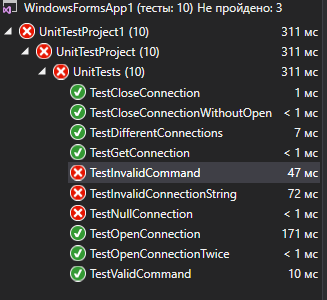


Рисунок 49 – UnitTest библиотеки

Код модульных тестов представлены в приложении Д.

* 1. **Выгрузка готового проекта в репозиторий Git**

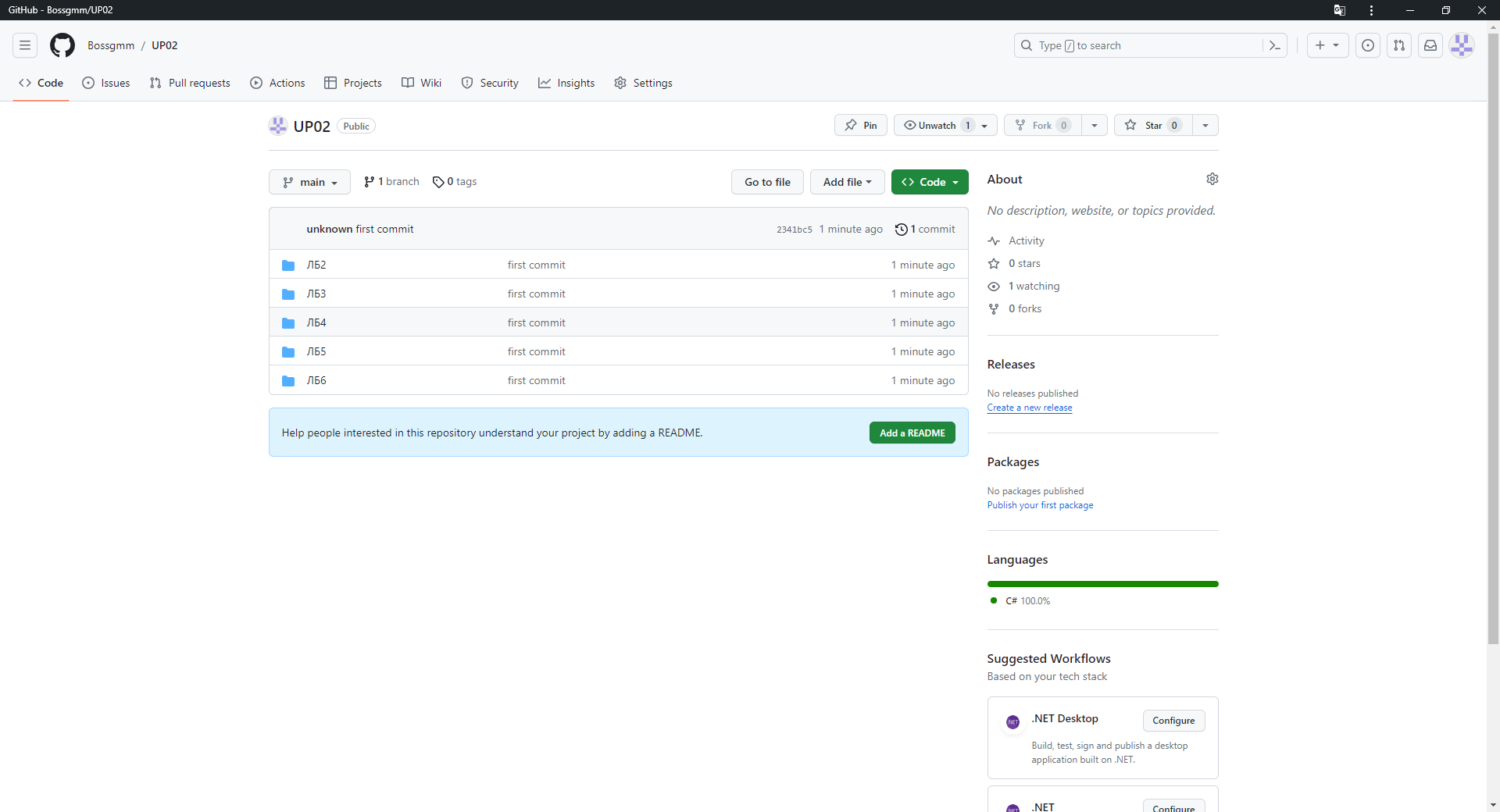


Рисунок 50 – Проект выгружен в репозиторий Git

# **Заключение**

Результатом выполнения работы стала база данных и приложение, выполняющее с ней операции.

Были рассмотрены и опробованы основные методы работы с базой данных. Получены практические навыки работы с базой данных, а также изучены новые возможности среды программирования С#.

База данных была разработана с использованием СУБД SQL Server. Для разработки пользовательского приложения была использована среда Microsoft Visual Studio 2022 Community, язык программирования C# и библиотека для разработки WindowsForms.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. РАБОТА С НЕСТРУКТУРИРОВАННЫМИ ДАННЫМИ // WorldSkillsRussia URL: https://nationalteam.worldskills.ru/skills/rabota-s-nestrukturirovannymi-dannymi-obrabotka-i-import-v-bazu-dannykh/ (дата обращения: 07.11.2023).
2. Импорт xml файла в базу данных MSSQl // Admin Day URL: https://blogadminday.ru/import-xml-fayla-v-bazu-dannyh-mssql/ (дата обращения: 07.11.2023).
3. Импорт CSV в SQL Server // Professor Web URL: https://professorweb.ru/my/it/blog/net/csv\_sql\_server.php (дата обращения: 08.11.2023).
4. Руководство по MS SQL Server 2022 // METANIT.COM URL: https://metanit.com/sql/sqlserver/?ysclid=lp3ofcoswx406598108 (дата обращения: 10.11.2023).
5. Учебное пособие по диаграмма прецедентов (Руководство с примерами) // CREATELY URL: https://creately.com/blog/ru/%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0/%D1%83%D1%87%D0%B5%D0%B1%D0%BD%D0%BE%D0%B5-%D0%BF%D0%BE%D1%81%D0%BE%D0%B1%D0%B8%D0%B5-%D0%BF%D0%BE-%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0-%D0%BF%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5/ (дата обращения: 15.11.2023).

# **ПРИЛОЖЕНИЕ А**

USE [419/4 Давыдов]

GO

CREATE TABLE "Командир судна" (

ИН int,

ФИО nvarchar not null,

АДРЕС nvarchar,

ТЕЛЕФОН nvarchar,

НАЛЁТ int not null,

PRIMARY KEY(ИН)

);

INSERT INTO "Командир судна" (ИН, ФИО, АДРЕС, ТЕЛЕФОН, НАЛЁТ)

VALUES (0, 'Пригожин Евгений Викторович', 'пр. Луначарского д.78', '89312042888', 10000);

INSERT INTO "Командир судна" (ИН, ФИО, АДРЕС, ТЕЛЕФОН, НАЛЁТ)

VALUES (1, 'Давыдов Даниил Денисович', 'пр. Луначарского д.79', '89217555469', 5000);

CREATE TABLE Авиасудна(

"БОРТ №" int,

"МОДЕЛЬ" nvarchar not null,

"ДАТА ИЗГОТОВЛЕНИЯ" date not null,

"НАЛЁТ" int not null,

"ГОТОВНОСТЬ" bit not null,

PRIMARY KEY("БОРТ №","ИН КМД")

);

INSERT INTO Авиасудна ("БОРТ №", "ИН КМД", "МОДЕЛЬ", "ДАТА ИЗГОТОВЛЕНИЯ", "НАЛЁТ", "ГОТОВНОСТЬ")

VALUES (0, 0, 'Боинг', '2014-06-28', 1500, 1);

INSERT INTO Авиасудна ("БОРТ №", "ИН КМД", "МОДЕЛЬ", "ДАТА ИЗГОТОВЛЕНИЯ", "НАЛЁТ", "ГОТОВНОСТЬ")

VALUES (1, 1, 'Сусемь', '1996-08-05', 20000, 0);

CREATE TABLE Пассажиры(

"НОМЕР ПАСПОРТА" nvarchar(11),

"№ РЕЙСА" int not null,

"ФИО" nvarchar not null,

"АДРЕС" nvarchar,

"ТЕЛЕФОН" nvarchar,

PRIMARY KEY("НОМЕР ПАСПОРТА")

);

INSERT INTO Пассажиры ("НОМЕР ПАСПОРТА", "№ РЕЙСА", "ФИО", "АДРЕС", "ТЕЛЕФОН")

VALUES ('4018-374228', 0, 'Савельев Алексей Дмитриевич', 'п. Лейпясуо', '89517556954');

INSERT INTO Пассажиры ("НОМЕР ПАСПОРТА", "№ РЕЙСА", "ФИО", "АДРЕС", "ТЕЛЕФОН")

VALUES ('4018-374229', 0, 'Савельев Дмитрий Игоревич', 'п. Лейпясуо', '89216677569');

INSERT INTO Пассажиры ("НОМЕР ПАСПОРТА", "№ РЕЙСА", "ФИО", "АДРЕС", "ТЕЛЕФОН")

VALUES ('2020-100200', 1, 'Елена Быстрова Олеговна', 'г.СПб, ул.Победы, д.78', '89517556954');

INSERT INTO Пассажиры ("НОМЕР ПАСПОРТА", "№ РЕЙСА", "ФИО", "АДРЕС", "ТЕЛЕФОН")

VALUES ('2023-200100', 1, 'Иван Быстров Александрович', 'г.СПб, ул.Победы, д.78', NULL);

CREATE TABLE Рейс(

"№ РЕЙСА" int,

"№ МАРШРУТА" int not null,

"БОРТ №" int not null,

"ДАТА/ВРЕМЯ ВЫЛЕТА" smalldatetime not null,

"СТАТУС" bit not null,

PRIMARY KEY("№ РЕЙСА")

);

INSERT INTO Рейс ("№ РЕЙСА", "№ МАРШРУТА", "БОРТ №", "ДАТА/ВРЕМЯ ВЫЛЕТА", "ОТМЕНЁН?")

VALUES (0, 0, 0, '2023-11-23 22:15:00', 0);

INSERT INTO Рейс ("№ РЕЙСА", "№ МАРШРУТА", "БОРТ №", "ДАТА/ВРЕМЯ ВЫЛЕТА", "ОТМЕНЁН?")

VALUES (1, 1, 1, '2023-11-20 19:30:00', 1);

CREATE TABLE Маршрут(

"НОМЕР" int,

"ОТБЫТИЕ" nvarchar not null,

"ПРИБЫТИЕ" nvarchar not null,

"ЦЕНА" nvarchar not null,

"ДЛИТЕЛЬНОСТЬ" int not null,

PRIMARY KEY (НОМЕР)

);

INSERT INTO Маршрут(НОМЕР, ОТБЫТИЕ, ПРИБЫТИЕ, ЦЕНА, ДЛИТЕЛЬНОСТЬ)

VALUES(0, 'Москва','Нью-Йорк','11199 руб',10);

INSERT INTO Маршрут(НОМЕР, ОТБЫТИЕ, ПРИБЫТИЕ, ЦЕНА, ДЛИТЕЛЬНОСТЬ)

VALUES(1, 'Нью-Йорк','Москва','11199 руб',10);

INSERT INTO Маршрут(НОМЕР, ОТБЫТИЕ, ПРИБЫТИЕ, ЦЕНА, ДЛИТЕЛЬНОСТЬ)

VALUES(2, 'Москва','Владивосток','4999 руб',6);

INSERT INTO Маршрут(НОМЕР, ОТБЫТИЕ, ПРИБЫТИЕ, ЦЕНА, ДЛИТЕЛЬНОСТЬ)

VALUES(3, 'Владивосток', 'Москва', '4999 руб',6);

# **ПРИЛОЖЕНИЕ Б**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

class DataBase

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source = DESKTOP-A63B5DQ; Initial Catalog = 419\_4\_Давыдов; Integrated Security = True");

public void openConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Closed)

{

sqlConnection.Open();

}

}

public void closeConnection()

{

if (sqlConnection.State == System.Data.ConnectionState.Open)

{

sqlConnection.Close();

}

}

public SqlConnection getConnection()

{

return sqlConnection;

}

}

}

# **ПРИЛОЖЕНИЕ В**

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;

using System.Data.SqlClient;

//using System.Threading;

namespace WindowsFormsApp1

{

public partial class log\_in : Form

{

DataBase dataBase = new DataBase();

private string text = String.Empty;

private int count = 0;

int time= 180;

DataTable table;

SqlDataAdapter adapter;

SqlCommandBuilder builder;

public void CreateTable(string sqlCommand)

{

dataBase.openConnection();

table = new DataTable("dataGridView1");

adapter = new SqlDataAdapter();

adapter.SelectCommand = new SqlCommand(sqlCommand, dataBase.getConnection());

builder = new SqlCommandBuilder(adapter);

adapter.Fill(table);

dataGridView1.DataSource = table;

dataBase.closeConnection();

}

public log\_in()

{

InitializeComponent();

//pictureBox1.Image = this.CreateImage(pictureBox1.Width, pictureBox1.Height);

StartPosition = FormStartPosition.CenterScreen;

button1.Visible = false;

textBox1.Visible = false;

button2.Visible = false;

label4.Visible = false;

textBox\_pwd.PasswordChar = '•';

CreateTable("SELECT \* FROM Рейс");

}

private void button\_Enter\_Click(object sender, EventArgs e)

{

count++;

var login = textBox\_login.Text;

var password = textBox\_pwd.Text;

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable table = new DataTable();

string queryString = $"select id, login, password FROM Admins " +

$"where login = '{login}' and password = '{password}'";

SqlCommand command = new SqlCommand(queryString, dataBase.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

//var a = table.Columns;

//if(a == "Администратор")

dataBase.openConnection();

string history = $"insert into History(id\_user, \_time, succsesful)";

if (table.Rows.Count != 1)

{

history += $" values ('-1', '{DateTime.Now.ToString("yyyy-dd-MM HH:mm:ss")}', 'false')";

}

else

{

history += $" values ({table.Rows[0].ItemArray[0]}, '{DateTime.Now.ToString("yyyy-dd-MM HH:mm:ss")}', 'true')";

}

SqlCommand his = new SqlCommand(history, dataBase.getConnection());

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter();

sqlDataAdapter.SelectCommand = his;

his.ExecuteNonQuery();

if (table.Rows.Count == 1)

{

MessageBox.Show("Вы успешно вошли", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

AdminPanel adm = new AdminPanel();

adm.Show();

}

else

{

MessageBox.Show("Не получилось", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

label3.Text = $"Количество попыток {count}/3";

if (count == 1)

{

button1.Visible = true;

button\_Enter.Visible = false;

textBox1.Visible = true;

textBox\_login.Enabled = false;

textBox\_pwd.Enabled = false;

button2.Visible = true;

textBox\_login.Text = "Введите CAPTCHA!!!";

textBox\_pwd.Text = "Введите CAPTCHA!!!";

pictureBox1.Image = this.CreateImage(pictureBox1.Width, pictureBox1.Height);

}

if (count == 2)

{

label4.Visible = true;

timer1.Start();

textBox\_login.Enabled = false;

textBox\_pwd.Enabled = false;

button\_Enter.Enabled = false;

}

if (count == 3)

{

Application.Exit();

}

}

}

private Bitmap CreateImage(int Width, int Height)

{

Random rnd = new Random();

//Создадим изображение

Bitmap result = new Bitmap(Width, Height);

//Вычислим позицию текста

int Xpos = rnd.Next(0, Width - 50);

int Ypos = rnd.Next(15, Height - 15);

//Добавим различные цвета

Brush[] colors = { Brushes.Black,

Brushes.Red,

Brushes.RoyalBlue,

Brushes.Green };

//Укажем где рисовать

Graphics g = Graphics.FromImage(result);

//Пусть фон картинки будет серым

g.Clear(Color.Gray);

//Сгенерируем текст

text = String.Empty;

string ALF = "1234567890QWERTYUIOPASDFGHJKLZXCVBNM";

for (int i = 0; i < 5; ++i)

text += ALF[rnd.Next(ALF.Length)];

//Нарисуем сгенирируемый текст

g.DrawString(text,

new Font("Arial", 15),

colors[rnd.Next(colors.Length)],

new PointF(Xpos, Ypos));

//Добавим немного помех

/////Линии из углов

g.DrawLine(Pens.Black,

new Point(0, 0),

new Point(Width - 1, Height - 1));

g.DrawLine(Pens.Black,

new Point(0, Height - 1),

new Point(Width - 1, 0));

////Белые точки

//for (int i = 0; i < Width; ++i)

// for (int j = 0; j < Height; ++j)

// if (rnd.Next() % 20 == 0)

// result.SetPixel(i, j, Color.White);

return result;

}

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

{

pictureBox1.Image = this.CreateImage(pictureBox1.Width, pictureBox1.Height);

}

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

{

if (textBox1.Text == this.text)

{

MessageBox.Show("Верно!");

textBox\_login.Enabled = true;

textBox\_pwd.Enabled = true;

textBox\_login.Text = "";

textBox\_pwd.Text = "";

button\_Enter.Visible = true;

button1.Visible = false;

textBox1.Visible = false;

button2.Visible = false;

pictureBox1.Visible = false;

}

else

{

MessageBox.Show("ERROR!!!");

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

time--;

label4.Text = $"Осталось: {time}";

if (time == 0)

{

timer1.Stop();

textBox\_login.Enabled = true;

textBox\_pwd.Enabled = true;

button\_Enter.Enabled = true;

MessageBox.Show("Время закончилось!!!");

}

}

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

{

BuyForm buy = new BuyForm();

buy.Show();

}

}

}

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;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class AdminPanel : Form

{

DataBase dataBase = new DataBase();

public DataTable table;

public SqlDataAdapter adapter;

public SqlCommandBuilder builder;

public AdminPanel()

{

InitializeComponent();

}

public void CreateTable(string sqlCommand)

{

dataBase.openConnection();

table = new DataTable("dataGridView1");

adapter = new SqlDataAdapter();

adapter.SelectCommand = new SqlCommand(sqlCommand, dataBase.getConnection());

builder = new SqlCommandBuilder(adapter);

adapter.Fill(table);

dataGridView1.DataSource = table;

dataBase.closeConnection();

}

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

{

CreateTable("SELECT \* FROM Маршрут");

}

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

{

CreateTable("select\* from Рейс");

}

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

{

CreateTable("select \* from Авиасудна");

}

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

{

CreateTable("select \* from КомандирСудна");

}

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

{

CreateTable("select \* from Admins");

}

private void button6\_Click(object sender, EventArgs e) //История

{

CreateTable("select \* from History");

}

private void button8\_Click(object sender, EventArgs e) //Пассажиры

{

CreateTable("SELECT \* FROM Пассажиры");

}

private void button7\_Click(object sender, EventArgs e) //Сохранение

{

dataBase.openConnection();

try

{

if (adapter != null && builder != null)

{

adapter.Update(table);

}

}

catch

{

MessageBox.Show("Упс! Вы наворотили херни с данными, поэтому они не сохранились. Исправте ошибки и попробуйте снова");

}

dataBase.closeConnection();

}

}

}

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;

using System.Data.SqlClient;

namespace WindowsFormsApp1

{

public partial class BuyForm : Form

{

DataBase dataBase = new DataBase();

public BuyForm()

{

InitializeComponent();

}

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

{

if (textBox1.Text == "" || textBox2.Text == "" || textBox3.Text == "")

{

MessageBox.Show("Ошибка! Поля с пометкой \* обязательны к заполнению");

return;

}

if (textBox4.Text == "")

textBox4.Text = "NULL";

if (textBox5.Text == "")

textBox5.Text = "NULL";

dataBase.openConnection();

var addQuery = $"INSERT INTO Пассажиры (НомерПаспорта, [№Рейса], ФИО, АДРЕС, ТЕЛЕФОН) VALUES ('{textBox1.Text}','{textBox2.Text}','{textBox3.Text}',{textBox4.Text},'{textBox5.Text}')";

var command = new SqlCommand(addQuery, dataBase.getConnection());

command.ExecuteNonQuery();

dataBase.closeConnection();

}

}

}

# **ПРИЛОЖЕНИЕ Г**

Test-Cases

|  |  |
| --- | --- |
| **Название проекта** | Авиакомпания |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Давыдов Даниил Денисович |
| **Дата(ы) теста** | 19.11.2023 |

Тестовый пример 1:

|  |  |
| --- | --- |
| **Тестовый пример №1** | TestOpenConnection |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка открытого соединения с БД |
| **Краткое изложение теста** | Открыто ли соединение с БД |
| **Этапы теста** | Открытие соединения с БД  Проверка соединения |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Установление соединения с БД. |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Соединение открыто |
| **Постусловие** | Нет изменений |
| **Примечания** | - |

Тестовый пример 2:

|  |  |
| --- | --- |
| **Тестовый пример №2** | TestCloseConnection |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка закрытого соединения с БД |
| **Краткое изложение теста** | Закрыто ли соединение с БД |
| **Этапы теста** | Закрытие соединения с БД  Проверка соединения |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Закрытие соединения с БД. |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Соединение открыто |
| **Постусловие** | Нет изменений |
| **Примечания** | - |

Тестовый пример 3:

|  |  |
| --- | --- |
| **Тестовый пример №3** | TestGetConnection |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка получения соединения с БД |
| **Краткое изложение теста** | Получено ли соединение с БД |
| **Этапы теста** | Закрытие соединения с БД  Проверка получения соединения |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Получения соединения с БД. |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Соединение получено |
| **Постусловие** | Нет изменений |
| **Примечания** | - |

Тестовый пример 4:

|  |  |
| --- | --- |
| **Тестовый пример №4** | TestOpenConnectionTwice |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка открытого соединения с БД 2 раза |
| **Краткое изложение теста** | Открыто ли соединение с БД |
| **Этапы теста** | Открытие соединения с БД  Проверка получения соединения |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Получения соединения с БД 2 раза. |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Соединение получено |
| **Постусловие** | Нет изменений |
| **Примечания** | - |

Тестовый пример 5:

|  |  |
| --- | --- |
| **Тестовый пример №5** | TestCloseConnectionWithoutOpen |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка закрытие соединения с БД без открытия приложения |
| **Краткое изложение теста** | Закрыто ли соединение с БД |
| **Этапы теста** | Закрытие соединения с БД  Проверка получения соединения |
| **Тестовые данные** | - |
| **Ожидаемый результат** | закрытие соединения с БД без открытия приложения |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Соединение не получено |
| **Постусловие** | Нет изменений |
| **Примечания** | - |

Тестовый пример 6:

|  |  |
| --- | --- |
| **Тестовый пример №6** | TestAdminForm |
| **Приоритет тестирования** | Высокий |
| **Заголовок** | Проверка роли пользователя с логином «admin» |
| **Краткое изложение теста** | Открытие формы администртатора |
| **Этапы теста** | Ввод логина и пароля  Проверка открытия формы администратор |
| **Тестовые данные** | Логин –«admin»  Пароль – «admin» |
| **Ожидаемый результат** | Открытие формы администратора |
| **Фактический результат** | Соответствует ожидаемому результату |
| **Статус** | Зачет |
| **Предварительное условие** | Форма администратора открыта |
| **Постусловие** | Нет изменений |
| **Примечания** | - |

# **ПРИЛОЖЕНИЕ Д**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using ClassLibrary1;

using System.Data.SqlClient;

namespace UnitTestProject

{

[TestClass]

public class UnitTests

{

Class1 class1;

[TestInitialize]

public void TestInitialize()

{

class1 = new Class1();

}

[TestCleanup]

public void TestCleanup()

{

class1.closeConnection();

}

[TestMethod]

public void TestOpenConnection()

{

class1.openConnection();

SqlConnection connection = class1.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Open, connection.State);

}

[TestMethod]

public void TestCloseConnection()

{

class1.openConnection();

class1.closeConnection();

SqlConnection connection = class1.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Closed, connection.State);

}

[TestMethod]

public void TestGetConnection()

{

SqlConnection connection = class1.getConnection();

Assert.IsNotNull(connection);

}

[TestMethod]

public void TestOpenConnectionTwice()

{

class1.openConnection();

class1.openConnection();

SqlConnection connection = class1.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Open, connection.State);

}

[TestMethod]

public void TestCloseConnectionWithoutOpen()

{

class1.closeConnection();

SqlConnection connection = class1.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Closed, connection.State);

}

[TestMethod]

public void TestDifferentConnections()

{

Class1 class2 = new Class1();

class1.openConnection();

class2.openConnection();

SqlConnection connection1 = class1.getConnection();

SqlConnection connection2 = class2.getConnection();

Assert.AreNotSame(connection1, connection2);

}

[TestMethod]

public void TestInvalidConnectionString()

{

// Предполагается что здесь будет неверная строка подключения

Class1 class2 = new Class1();

class2.openConnection();

SqlConnection connection2 = class2.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Closed, connection2.State);

}

[TestMethod]

public void TestInvalidCommand()

{

class1.openConnection();

SqlCommand command = new SqlCommand("INVALID COMMAND", class1.getConnection());

SqlDataReader reader = command.ExecuteReader();

Assert.IsFalse(reader.HasRows);

}

[TestMethod]

public void TestNullConnection()

{

class1.closeConnection();

SqlConnection connection = class1.getConnection();

Assert.IsNull(connection);

}

[TestMethod]

public void TestConnectionTimeout()

{

// Предполагается, что здесь будет некорректная строка подключения с неправильно настроенным таймаутом.

class1.openConnection();

SqlConnection connection = class1.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Closed, connection.State);

}

[TestMethod]

public void TestUnsupportedDatabase()

{

// Предполагается, что здесь будет название базы данных, которая не поддерживается.

class1.openConnection();

SqlConnection connection = class1.getConnection();

Assert.AreEqual(System.Data.ConnectionState.Closed, connection.State);

}

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ (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) IV курса 42919/4 группы

Давыдов Даниил Денисович

(Фамилия, имя, отчество)

Место прохождения практики: Вычислительный центр Институт среднего профессионального образования, пр. Энгельса д.23

(наименование и адрес организации)

Период прохождения практики

с «6» ноября 2023 г. по «18» ноября 2023 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| Участие в выработке требований к программному обеспечению | 9 |  |
| Стадии проектирования программного обеспечения | 21 |  |
| Разработка модулей программного обеспечения | 24 |  |
| Тестирование программных модулей и их интеграции | 9 |  |
| Разработка программной документации и стандарты кодирования | 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 Коннова А.Е. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата «18» ноября 2023 г. (Ф.И.О.) (подпись)