МИНОБРНАУКИ РОССИИ

федеральное государственное автономное образовательное учреждение высшего образования

**«Омский государственный университет им. Ф.М. Достоевского»**

Институт среднего профессионального образования

и довузовской подготовки

**Методические рекомендации по выполнению демонстрационного экзамена для специальности 09.02.07**

**СОДЕРЖАНИЕ**

[Рекомендованный хронометраж этапов выполнения задания 3](#_dj0j922375is)

[АЛГОРИТМ ВЫПОЛНЕНИЯ ЗАДАНИЙ 4](#_jwor2vpgin1g)

[Модуль №1 Разработка БД 4](#_l7t4dzonj6hb)

[Модуль №2 Соадминистрирование баз данных и серверов 12](#_1oyxbs3fdpbs)

[Модуль №3 Проектирование и разработка информационных систем 15](#_810v0jvankkn)

[Модуль №3.1 Описание системы 23](#_5vgiysl8kwd0)

[Модуль №4 Осуществление интеграции программных модулей 25](#_6738u0lutp6j)

# Рекомендованный хронометраж этапов выполнения задания

| № | **Время** (3ч. 30мин) | Действие |
| --- | --- | --- |
| 1 | 40 мин | Разработка БД: ЕR – диаграмма (разработать модель, не заполняя данными), сохранить в ее в pdf, экспортировать в физическую БД, которую заполнить данными, предоставленными в задании |
| 2 | 40 мин | Разработка графического интерфейса (экранные формы в MS VS) |
| 3 | 50 мин | Связать графический интерфейс с БД (connect)  Выполнить обработку окна авторизации в соответствии с заданием  Выполнить загрузку таблиц БД в таблицы экранных форм  Выполнить SQL-запросы из задания |
| 4 | 20 мин | Тестирование (рассмотреть 2 или 3 тестовые случая, лучше представить в виде таблицы) |
| 5 | 20 мин | Описание системы (титульный лист, описание интерфейса) |
| 6 | 40 мин | Доделать, все что недоделано |

# АЛГОРИТМ ВЫПОЛНЕНИЯ ЗАДАНИЙ

## Модуль №1 Разработка БД

Текст задания: На основании описания брифинга и документов, представленных заказчиком, необходимо спроектировать ER-диаграмму для информационной системы. Обязательна 3 нормальная форма с обеспечением ссылочной целостности. При разработке диаграммы обратите внимание на согласованную осмысленную схему именования, создайте необходимые первичные и внешние ключи.

**ER – диаграмма должна быть представлена в формате .pdf** и содержать таблицы, связи между ними, атрибуты и ключи (типами данных на данном этапе можно пренебречь).

**Необходимые приложения:** Текст брифинга.pdf, Документы заказчика.zip

**Выполнение:**

1. Создаём соединение MySQL:

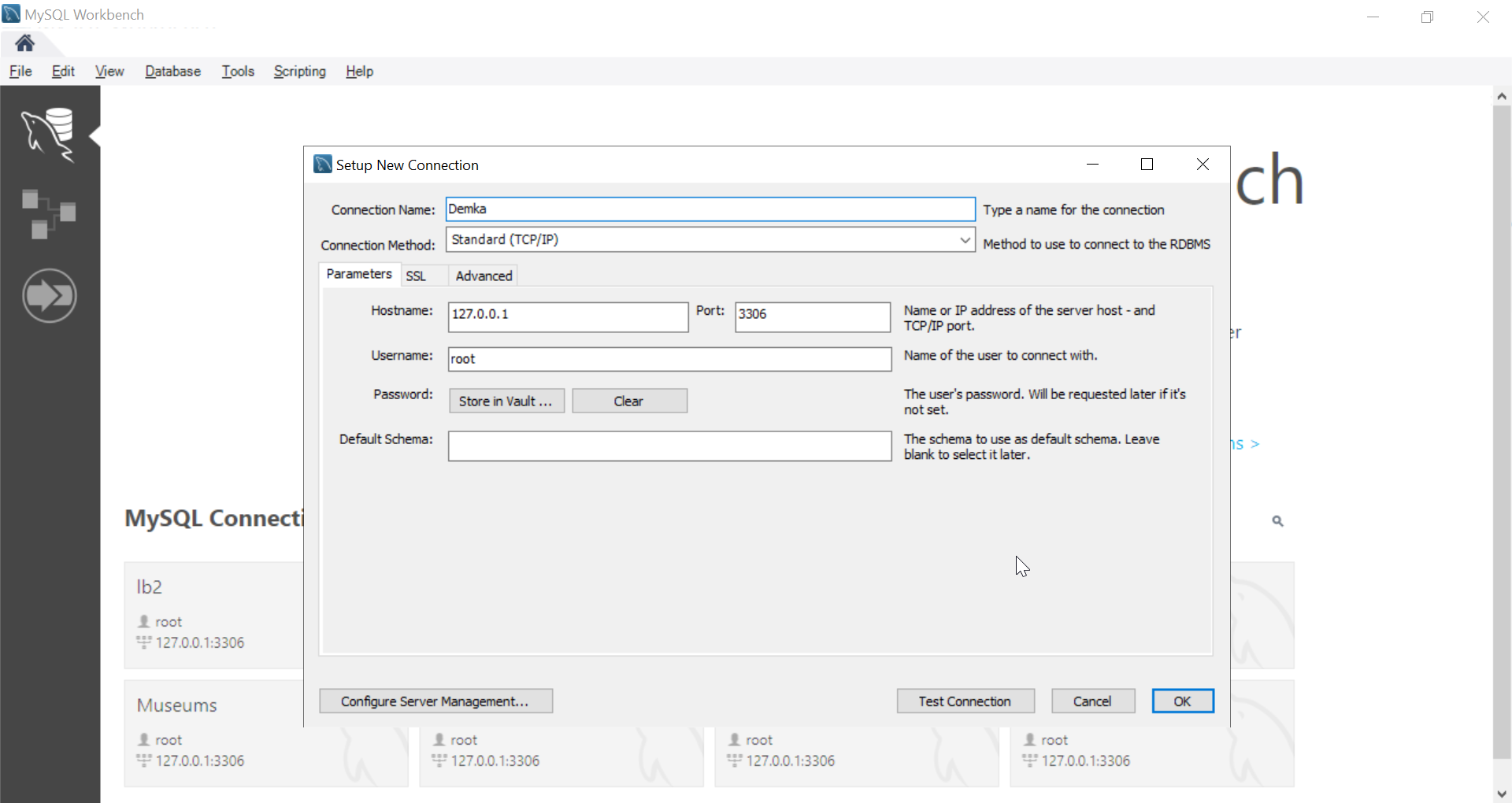


Рисунок 1 – Соединение MySQL

1. Создаём новую модель:

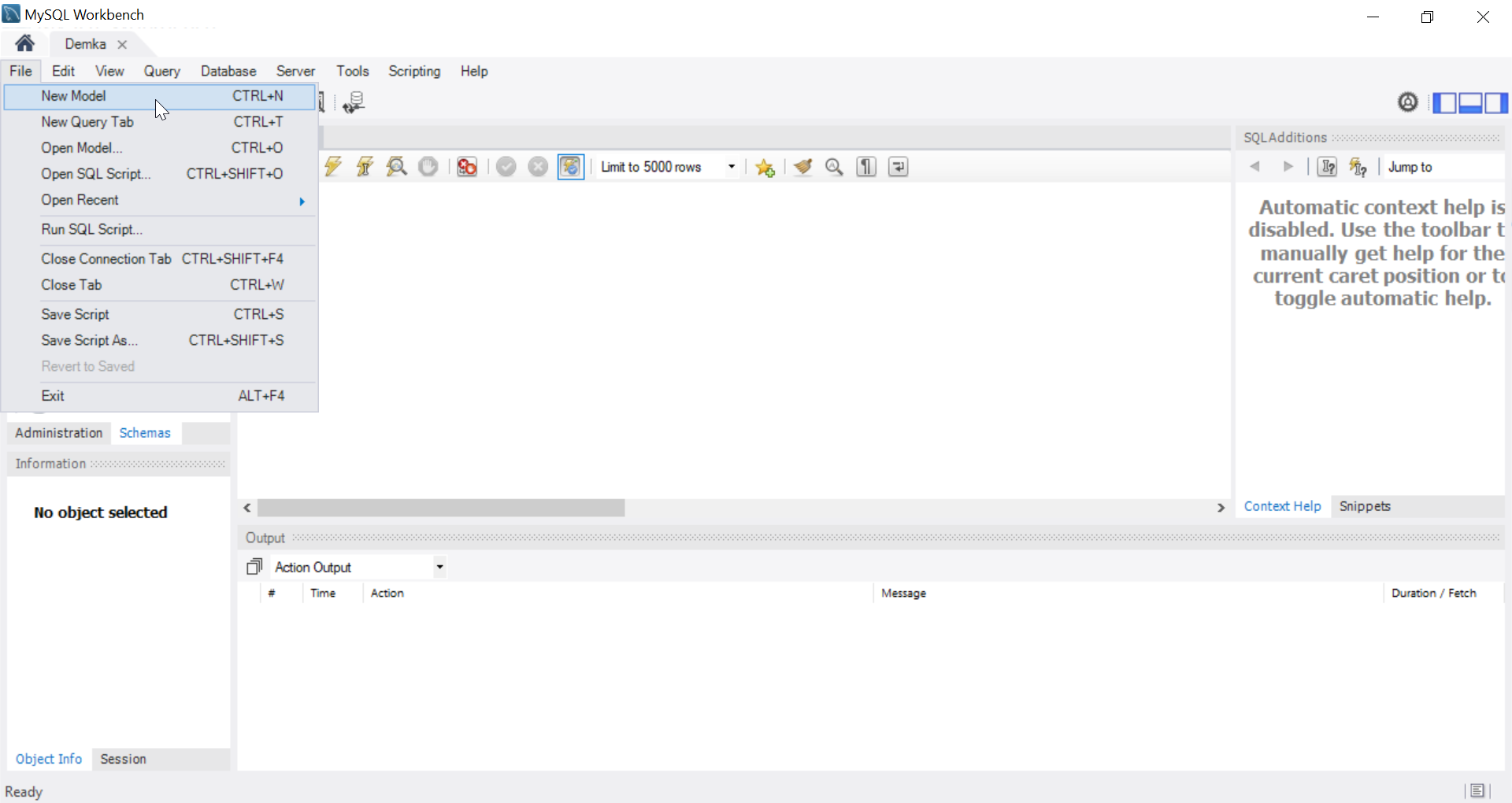


Рисунок 2 – Создание модели

1. Создаем диаграмму:

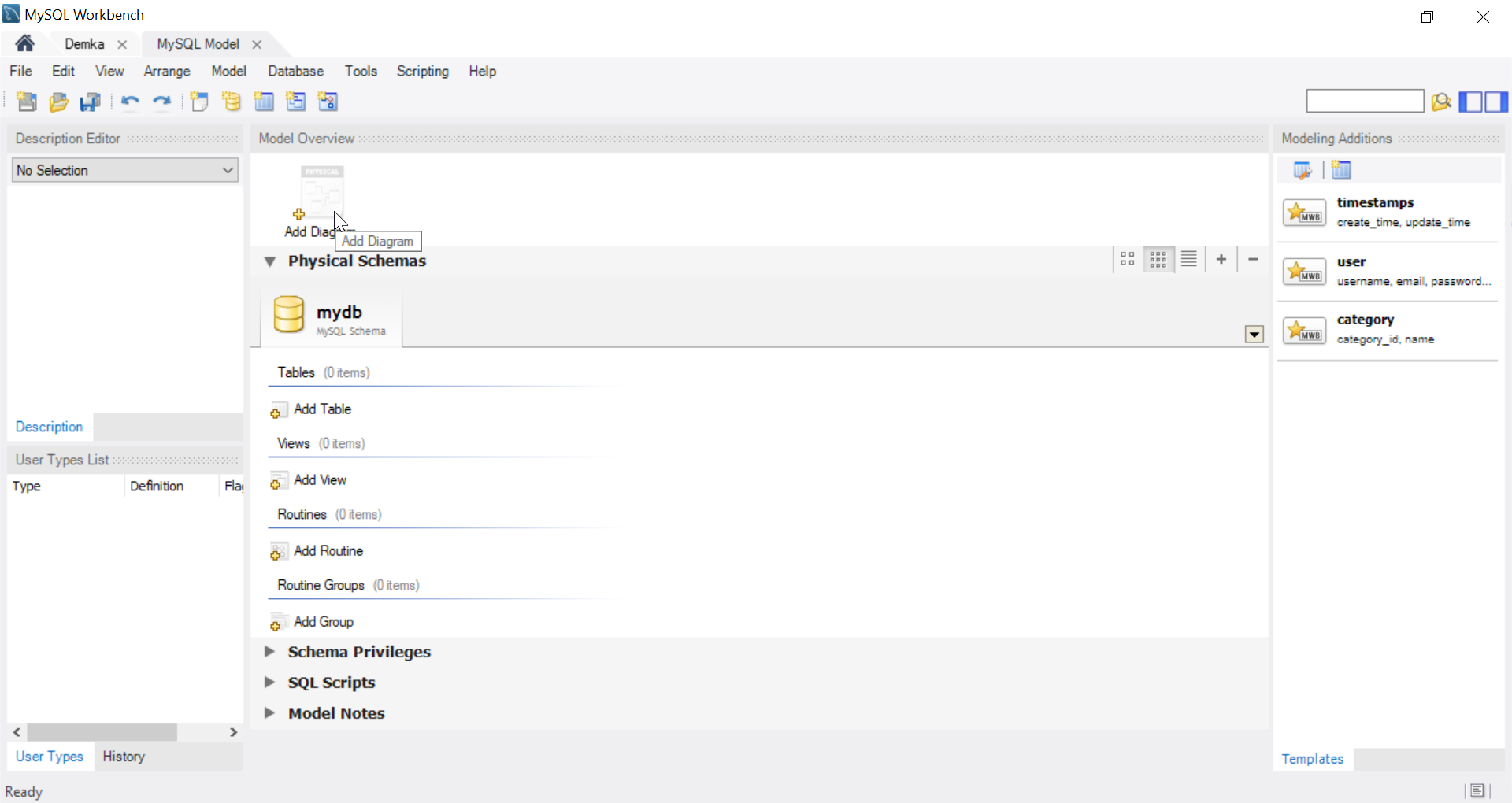


Рисунок 3 – Создание диаграммы

1. ER-моделирование (выделить сущности (таблицы), заполнить атрибуты (поля)). Таблицы Room\_stock, Report, Guests, Guests\_currently\_living\_in\_the\_hotel – делаются на основе Текста брифинга и Документам заказчика (по xlsx таблицам). Таблица Users обязательная и отличатся не будет. Она включает в себя ID пользователя, логин, пароль, роль (для разграничения доступа по ролям: администратор, пользователь), статус (для отслеживания статуса блокировки пользователя), попытки входа (для отслеживания количества попыток неудачного входа пользователя, 3 неудачи = блокировка), время последнего входа (для отслеживания последней попытки входа, пользователь неактивен 30 дней = блокировка, первая попытка входа (для отслеживания первой попытки входа и перенаправления пользователя на форму смены пароля).

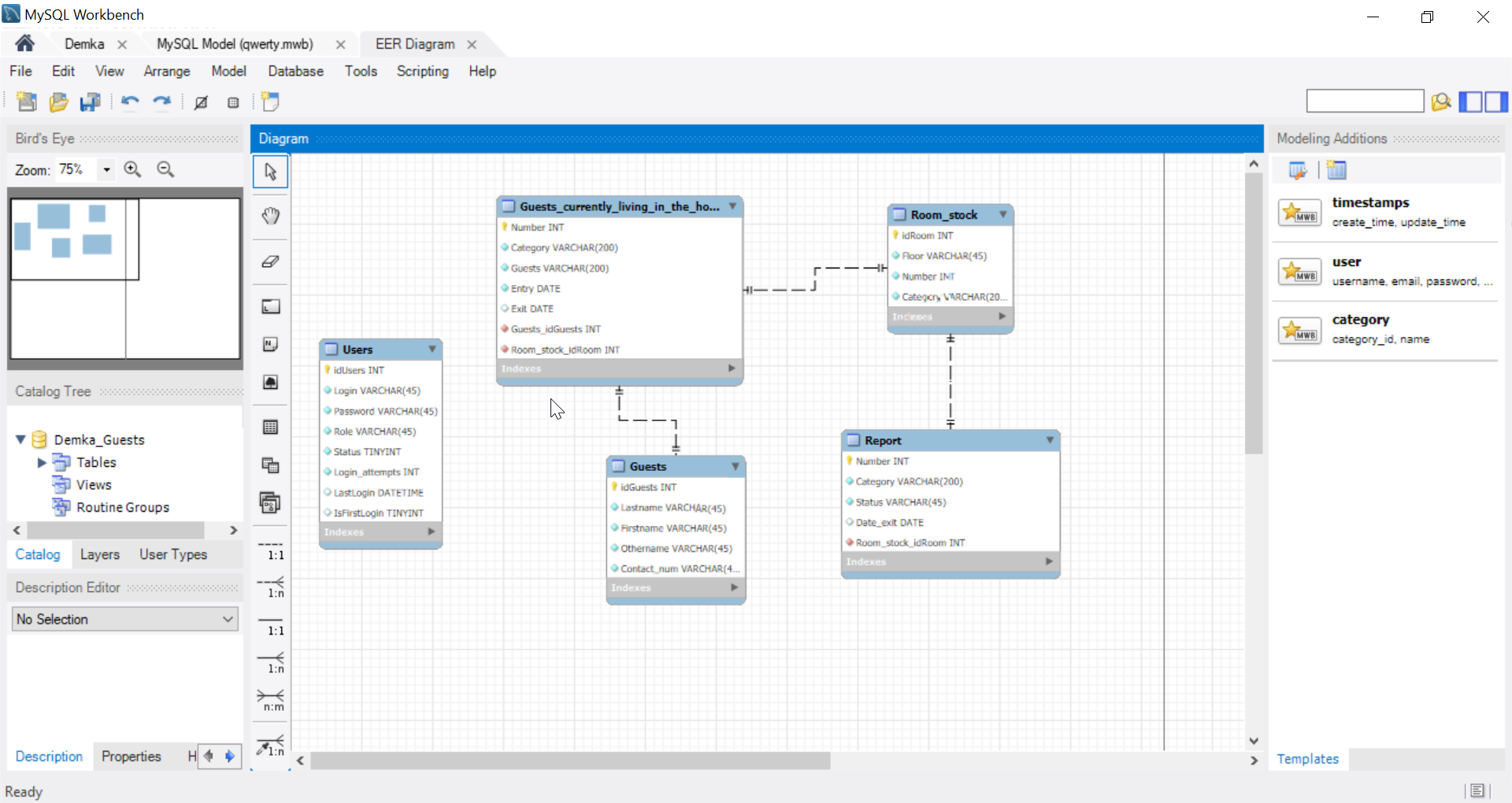


Рисунок 4 – ER-диаграмма

TINYINT = BOOLEAN

DATETIME(0)

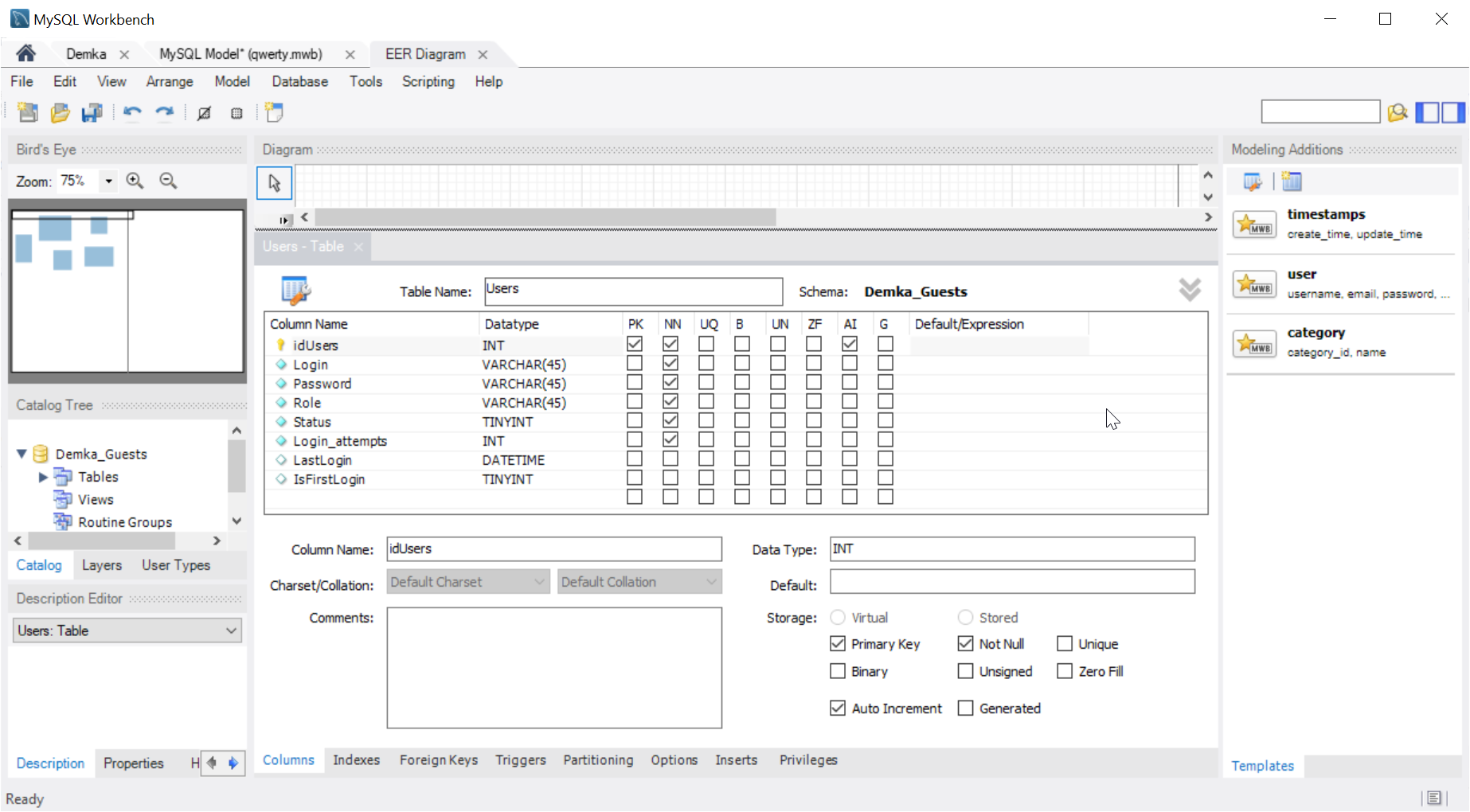


Рисунок 5 – Таблица Users

НЕ ЗАБЫТЬ ПЕРЕИМЕНОВАТЬ ТАБЛИЦУ СЛЕВА

1. По требованиям задания сохраним ER-диаграмму в формате .pdf:

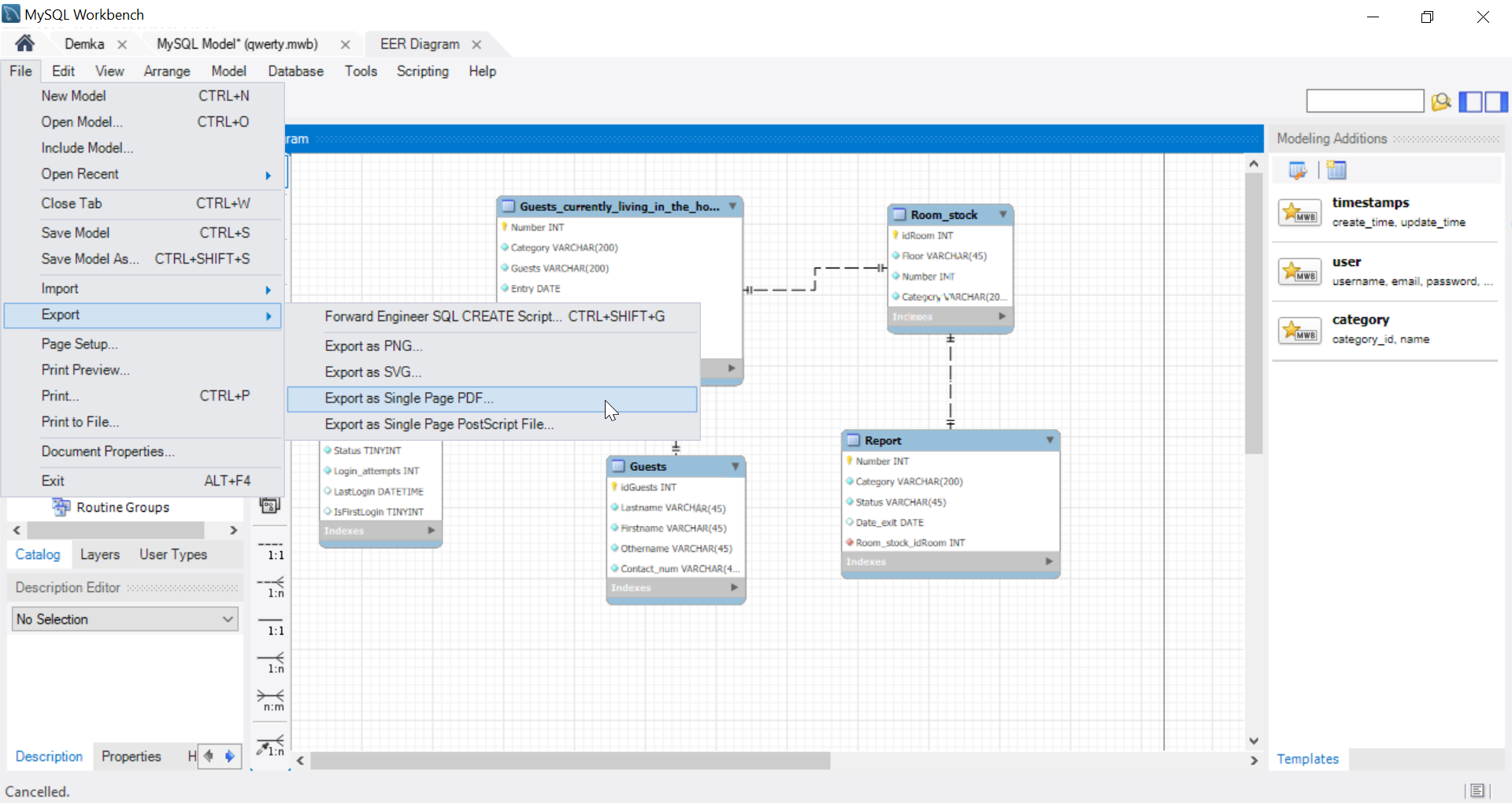


Рисунок 6 – Экспорт в pdf

1. Создаем SQL-скрипт для импортирования в физическую БД

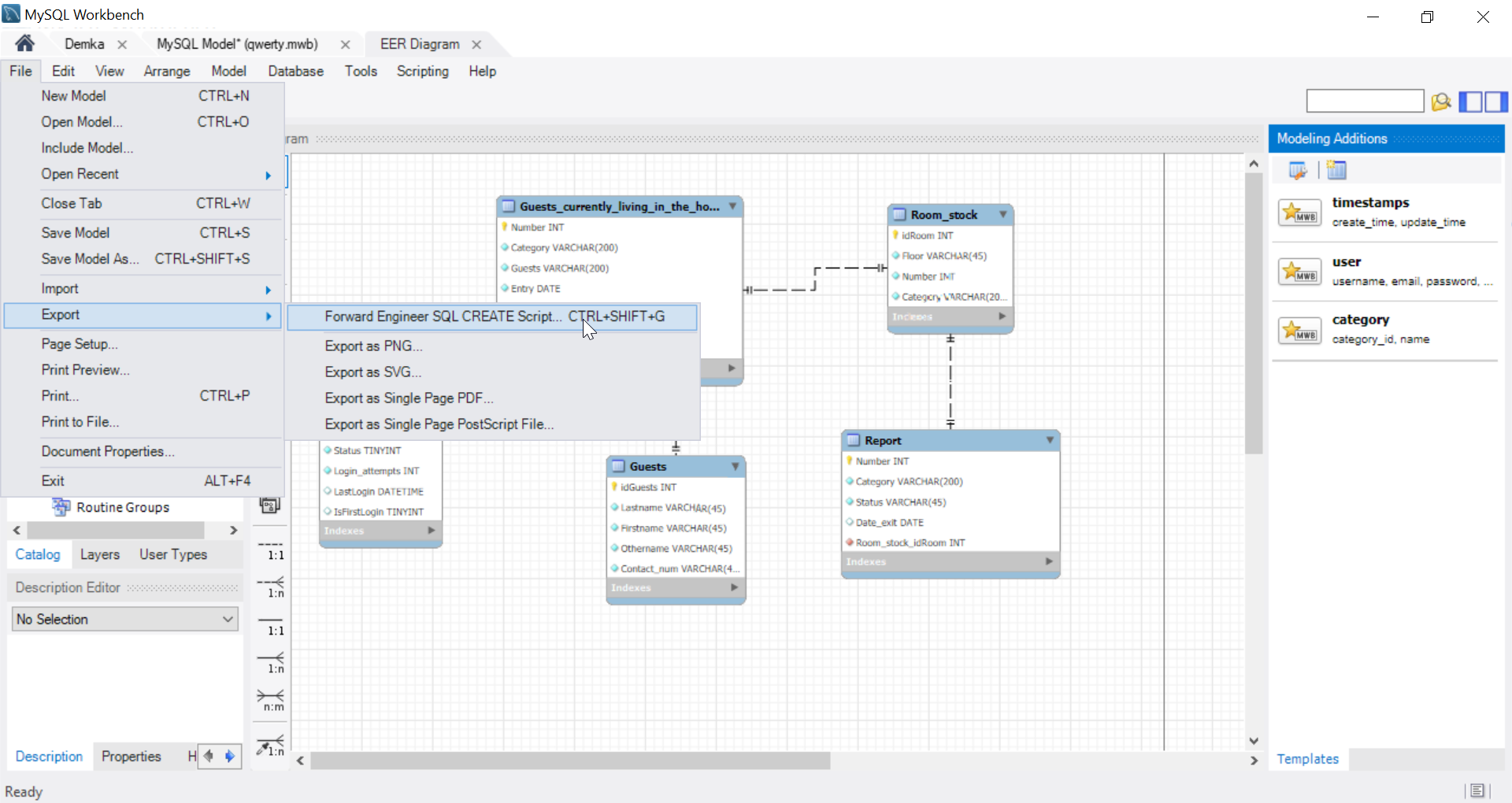


Рисунок 7 – Экспорт в физическую БД

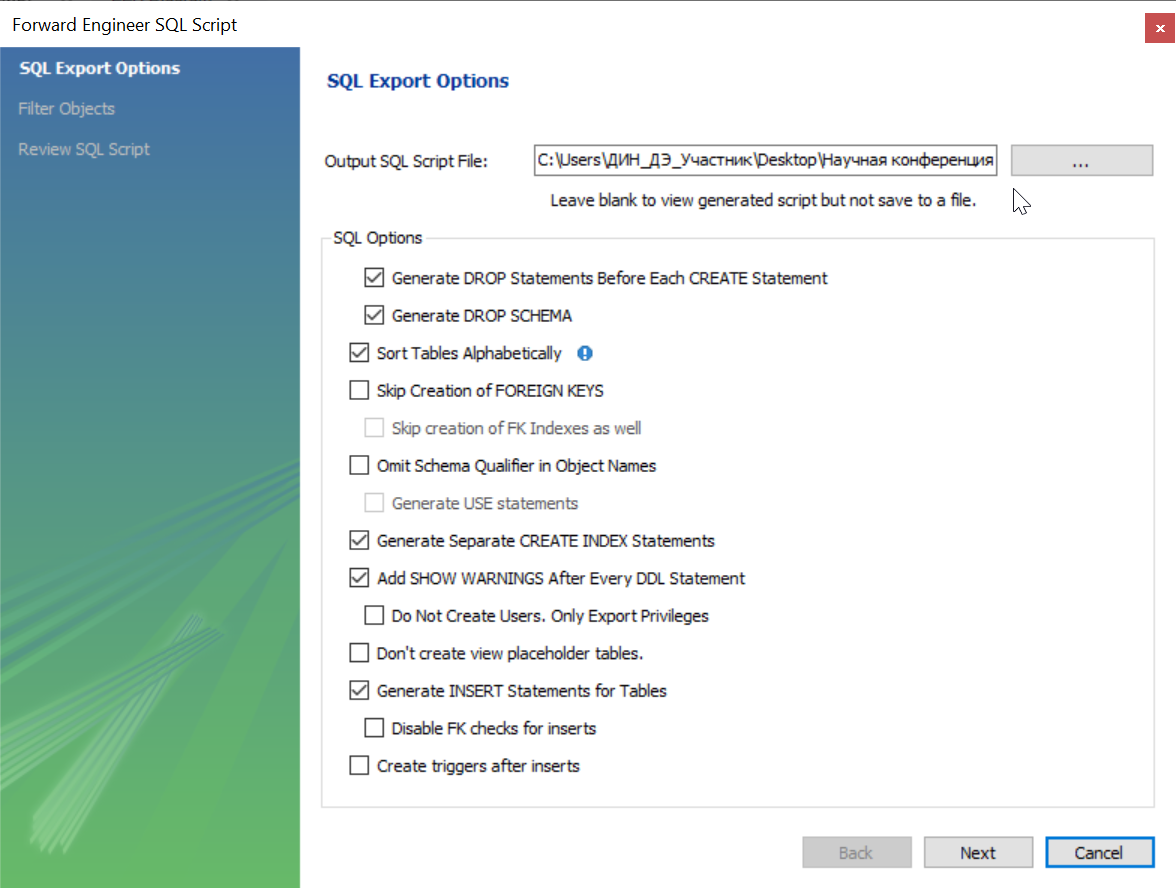


Рисунок 8 – Настройка экспорта

Листинг 1. SQL скрипт диаграммы:

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147 | -- MySQL Script generated by MySQL Workbench  -- Thu May 29 16:56:10 2025  -- Model: New Model Version: 1.0  -- MySQL Workbench Forward Engineering  SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;  SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;  SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';  -- -----------------------------------------------------  -- Schema Demka\_Guests  -- -----------------------------------------------------  DROP SCHEMA IF EXISTS `Demka\_Guests` ;  -- -----------------------------------------------------  -- Schema Demka\_Guests  -- -----------------------------------------------------  CREATE SCHEMA IF NOT EXISTS `Demka\_Guests` DEFAULT CHARACTER SET utf8 ;  SHOW WARNINGS;  USE `Demka\_Guests` ;  -- -----------------------------------------------------  -- Table `Demka\_Guests`.`Guests`  -- -----------------------------------------------------  DROP TABLE IF EXISTS `Demka\_Guests`.`Guests` ;  SHOW WARNINGS;  CREATE TABLE IF NOT EXISTS `Demka\_Guests`.`Guests` (  `idGuests` INT NOT NULL,  `Lastname` VARCHAR(45) NOT NULL,  `Firstname` VARCHAR(45) NOT NULL,  `Othername` VARCHAR(45) NOT NULL,  `Contact\_num` VARCHAR(45) NOT NULL,  PRIMARY KEY (`idGuests`))  ENGINE = InnoDB;  SHOW WARNINGS;  -- -----------------------------------------------------  -- Table `Demka\_Guests`.`Guests\_currently\_living\_in\_the\_hotel`  -- -----------------------------------------------------  DROP TABLE IF EXISTS `Demka\_Guests`.`Guests\_currently\_living\_in\_the\_hotel` ;  SHOW WARNINGS;  CREATE TABLE IF NOT EXISTS `Demka\_Guests`.`Guests\_currently\_living\_in\_the\_hotel` (  `Number` INT NOT NULL,  `Category` VARCHAR(200) NOT NULL,  `Guests` VARCHAR(200) NOT NULL,  `Entry` DATE NOT NULL,  `Exit` DATE NULL,  `Guests\_idGuests` INT NOT NULL,  `Room\_stock\_idRoom` INT NOT NULL,  PRIMARY KEY (`Number`),  CONSTRAINT `fk\_Guests\_currently\_living\_in\_the\_hotel\_Guests`  FOREIGN KEY (`Guests\_idGuests`)  REFERENCES `Demka\_Guests`.`Guests` (`idGuests`)  ON DELETE NO ACTION  ON UPDATE NO ACTION,  CONSTRAINT `fk\_Guests\_currently\_living\_in\_the\_hotel\_Room\_stock1`  FOREIGN KEY (`Room\_stock\_idRoom`)  REFERENCES `Demka\_Guests`.`Room\_stock` (`idRoom`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  SHOW WARNINGS;  CREATE INDEX `fk\_Guests\_currently\_living\_in\_the\_hotel\_Guests\_idx` ON `Demka\_Guests`.`Guests\_currently\_living\_in\_the\_hotel` (`Guests\_idGuests` ASC) VISIBLE;  SHOW WARNINGS;  CREATE INDEX `fk\_Guests\_currently\_living\_in\_the\_hotel\_Room\_stock1\_idx` ON `Demka\_Guests`.`Guests\_currently\_living\_in\_the\_hotel` (`Room\_stock\_idRoom` ASC) VISIBLE;  SHOW WARNINGS;  -- -----------------------------------------------------  -- Table `Demka\_Guests`.`Report`  -- -----------------------------------------------------  DROP TABLE IF EXISTS `Demka\_Guests`.`Report` ;  SHOW WARNINGS;  CREATE TABLE IF NOT EXISTS `Demka\_Guests`.`Report` (  `Number` INT NOT NULL,  `Category` VARCHAR(200) NOT NULL,  `Status` VARCHAR(45) NOT NULL,  `Date\_exit` DATE NULL,  `Room\_stock\_idRoom` INT NOT NULL,  PRIMARY KEY (`Number`),  CONSTRAINT `fk\_Report\_Room\_stock1`  FOREIGN KEY (`Room\_stock\_idRoom`)  REFERENCES `Demka\_Guests`.`Room\_stock` (`idRoom`)  ON DELETE NO ACTION  ON UPDATE NO ACTION)  ENGINE = InnoDB;  SHOW WARNINGS;  CREATE INDEX `fk\_Report\_Room\_stock1\_idx` ON `Demka\_Guests`.`Report` (`Room\_stock\_idRoom` ASC) VISIBLE;  SHOW WARNINGS;  -- -----------------------------------------------------  -- Table `Demka\_Guests`.`Room\_stock`  -- -----------------------------------------------------  DROP TABLE IF EXISTS `Demka\_Guests`.`Room\_stock` ;  SHOW WARNINGS;  CREATE TABLE IF NOT EXISTS `Demka\_Guests`.`Room\_stock` (  `idRoom` INT NOT NULL,  `Floor` VARCHAR(45) NOT NULL,  `Number` INT NOT NULL,  `Category` VARCHAR(200) NOT NULL,  PRIMARY KEY (`idRoom`))  ENGINE = InnoDB;  SHOW WARNINGS;  -- -----------------------------------------------------  -- Table `Demka\_Guests`.`users`  -- -----------------------------------------------------  DROP TABLE IF EXISTS `Demka\_Guests`.`users` ;  SHOW WARNINGS;  CREATE TABLE IF NOT EXISTS `Demka\_Guests`.`users` (  `idUsers` INT NOT NULL AUTO\_INCREMENT,  `Login` VARCHAR(45) NOT NULL,  `Password` VARCHAR(45) NOT NULL,  `Role` VARCHAR(45) NOT NULL,  `Status` TINYINT NOT NULL,  `Login\_attempts` INT NOT NULL,  `LastLogin` DATETIME NULL,  `IsFirstLogin` TINYINT NULL,  PRIMARY KEY (`idUsers`))  ENGINE = InnoDB;  SHOW WARNINGS;  SET SQL\_MODE=@OLD\_SQL\_MODE;  SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;  SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS; |
| --- | --- |

## Модуль №2 Соадминистрирование баз данных и серверов

Текст задания: Создайте базу данных на основании разработанной ER диаграммы, используя предпочтительную платформу, на сервере баз данных, который вам предоставлен. Создайте таблицы основных сущностей, атрибуты, отношения и необходимые ограничения.

После создания базы данных требуется импортировать данные из файла "Номерной фонд.xlsx".

Создайте запрос вычисляющий процент загрузки номерного фонда – это отношение количества проданных ночей к общему количеству номеров в отеле.

**Необходимые приложения:** Текст брифинга.pdf, Документы заказчика.zip

**Выполнение:**

1. Импортируем файл с моделью базы данных, для создания физической базы данных

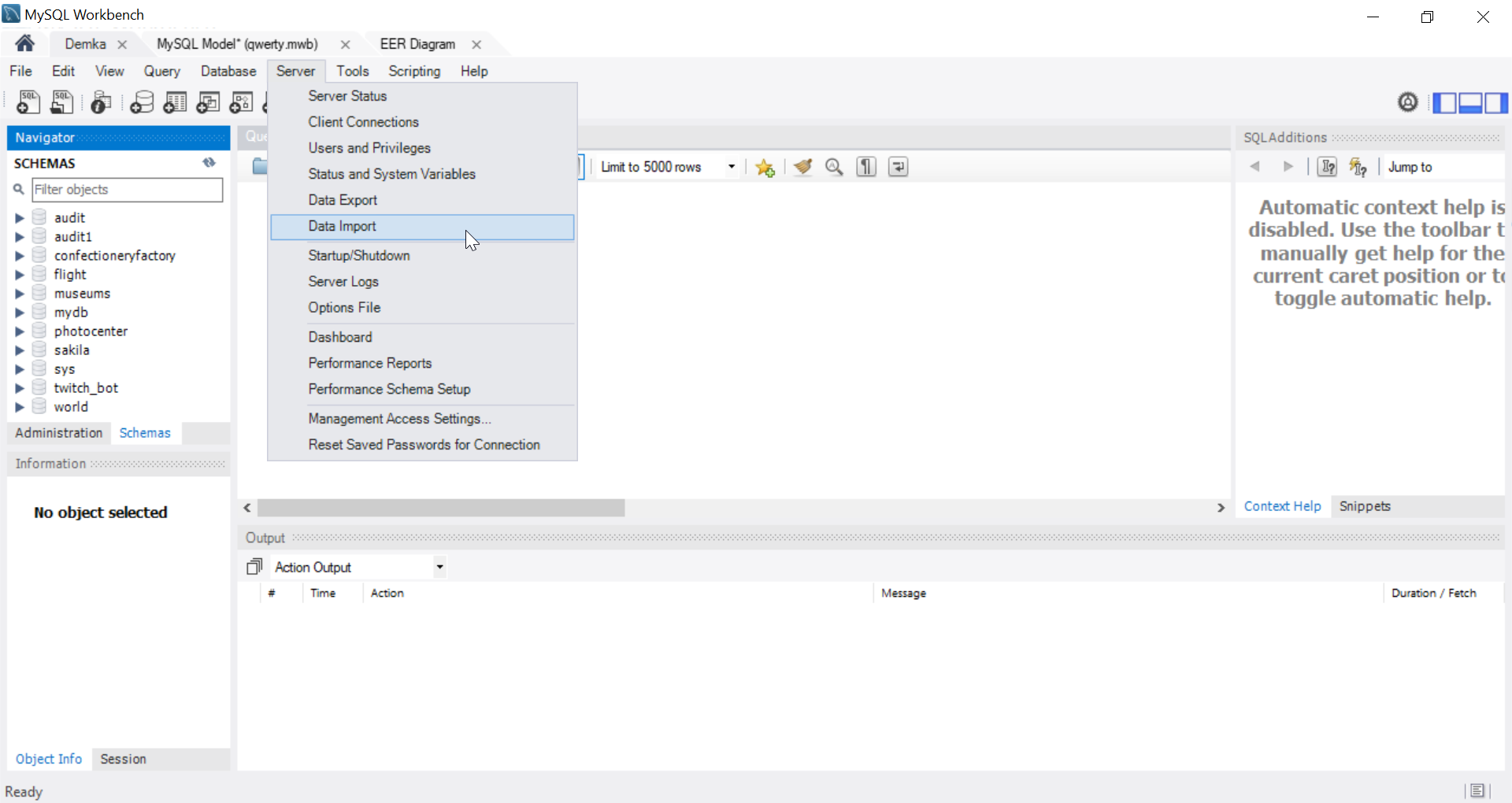


Рисунок 9 – Импорт ER-диаграммы

Далее выбираем место хранения sql-скрипта, который мы создали ранее.

## 

Рисунок 10 – Импорт ER-диаграммы

Переходим во вкладку Import Progress и нажимаем Start Import

## 

Рисунок 11 – Импорт ER-диаграммы

Обновляем список баз данных и находим нашу БД.

## 

Рисунок 12 – Импорт ER-диаграммы

Далее заполняем таблицы, т.е. импортируем данные из файла "Номерной фонд.xlsx".

**Необязательное задание (выполнять в последнюю очередь)**

Создаем запрос вычисляющий процент загрузки номерного фонда – это отношение количества проданных ночей к общему количеству номеров в отеле.

Если можно нейросеть:

Листинг 2. SQL запрос

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | Use demka\_guests;  -- Основной запрос для расчета текущей загрузки по статусам  SELECT  -- Общее количество номеров в отеле  COUNT(rs.idRoom) AS 'Всего номеров',    -- Количество занятых номеров (где статус = 'Занят')  SUM(CASE WHEN r.Status = 'Занят' THEN 1 ELSE 0 END) AS 'Занято номеров',    -- Расчет процента загрузки: (занятые номера / всего номеров) \* 100  -- Округляем до 1 знака после запятой  ROUND(  SUM(CASE WHEN r.Status = 'Занят' THEN 1 ELSE 0 END) / COUNT(rs.idRoom) \* 100,  1  ) AS 'Процент загрузки (%)'  FROM  -- Берем данные из таблицы номерного фонда  Room\_stock rs  LEFT JOIN  -- Соединяем с таблицей отчетов по номеру комнаты  -- LEFT JOIN чтобы включить все номера, даже без записей в отчетах  Report r ON rs.idRoom = r.Room\_stock\_idRoom  -- Можно добавить условие для актуальных данных (например, не выехавшие гости)  -- WHERE r.Date\_exit IS NULL OR r.Date\_exit >= CURDATE()  ; |
| --- | --- |



Листинг 2.1. SQL-запрос

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95 | using MySql.Data.MySqlClient;  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 Demka  {  public partial class SQL : Form  {  private readonly MySqlConnection \_conn;    public SQL()  {  InitializeComponent();  \_conn = new MySqlConnection("Server=localhost;Database=demka\_guests;Uid=root;Pwd=root;");    }  private void SQL\_Load(object sender, EventArgs e)  {  }  private void LoadData()  {  try  {  // Получаем данные о загрузке номеров из БД  DataTable table = GetOccupancyData();  // Устанавливаем источник данных для DataGridView  dataGridView1.DataSource = table;  // Настраиваем автоматическое изменение ширины столбцов  dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;  // Запрещаем редактирование данных  dataGridView1.ReadOnly = true;  // Скрываем заголовки строк  dataGridView1.RowHeadersVisible = false;  }  catch (Exception ex)  {  // Выводим сообщение об ошибке, если что-то пошло не так  MessageBox.Show("Ошибка: " + ex.Message);  }  }  // Метод для получения данных о загрузке номеров из базы данных  private DataTable GetOccupancyData()  {  // SQL-запрос для получения статистики:  // - общее количество номеров  // - количество занятых номеров  // - процент загрузки (с округлением до 1 знака)  string sql = @"SELECT  COUNT(rs.idRoom) AS 'Всего номеров',  SUM(CASE WHEN r.Status = 'Занят' THEN 1 ELSE 0 END) AS 'Занято',  ROUND(SUM(CASE WHEN r.Status = 'Занят' THEN 1 ELSE 0 END) /  COUNT(rs.idRoom) \* 100, 1) AS 'Загрузка (%)'  FROM Room\_stock rs  LEFT JOIN Report r ON rs.idRoom = r.Room\_stock\_idRoom";  // Создаем и открываем подключение к БД  using (var conn = new MySqlConnection(\_conn.ConnectionString))  {  // Создаем команду для выполнения SQL-запроса  MySqlCommand cmd = new MySqlCommand(sql, conn);  // Создаем адаптер для работы с данными  MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);  // Создаем таблицу для хранения результатов  DataTable table = new DataTable();  // Открываем соединение с БД  conn.Open();  // Заполняем таблицу данными из БД  adapter.Fill(table);  // Возвращаем заполненную таблицу  return table;  }  }  private void button1\_Click(object sender, EventArgs e)  {  LoadData();  }  }  } |
| --- | --- |

## Модуль №3 Проектирование и разработка информационных систем

Текст задания: Для выполнения задания рекомендуется создать в базе данных таблицу "Пользователи". Если такая таблица уже существует, необходимо внести некоторые изменения для реализации дальнейшего функционала приложения.

Разработайте форму для авторизации зарегистрированных пользователей с ролями "Администратор" и "Пользователь". Форма должна содержать поля текстовые поля логин, пароль и кнопку "Войти". Поля "Логин" и "Пароль" должны быть обязательными для заполнения. При неверно введенных данных, пользователь должен получить сообщение об ошибке "Вы ввели неверный логин или пароль. Пожалуйста проверьте ещё раз введенные данные".

После успешной авторизации пользователь должен получить сообщение "Вы успешно авторизовались".

При аутентификации связка «логин/пароль» должна совпадать с одной из записей в таблице "Пользователи".

При первой успешной авторизации по выданному паролю администратором должна выводится форма для смены пароля. Форма должна включать текущий пароль, новый пароль, подтверждение нового пароля. Все поля обязательные для заполнения. После заполнения формы и нажатия кнопки "Изменить пароль", система должна проверить правильность введенного текущего пароля и совпадение нового пароля с подтверждением.

В случае ошибок при заполнении формы пользователю должно выводиться сообщение об ошибке. В случае успешного изменения пароля, пользователю должно выводиться сообщение об успешной смене пароля.

Если в течении 3-х раз подряд был неверно введен логин/пароль, то учетная запись блокируется и при повторном авторизации должно появляться сообщение "Вы заблокированы. Обратитесь к администратору".

Также учетная запись должна блокироваться если пользователь не авторизовался в течении 1 месяца.

На рабочем столе пользователя с ролью "Администратор" предусмотрите функционал для добавления новых пользователей, изменения данных текущих пользователей (включая снятие блокировки). При добавлении нового пользователя следует проверять его наличие в базе данных. В случае, если пользователь с указанным логином уже существует, должно выводиться соответствующее сообщение.

**Графический интерфейс необходимо разработать в соответствии с требованиями к разработке.**

Разработайте проектную документацию на разработанный функционал. Включите описание функционального назначения, используемые методы с указанием параметров **(см. Модуль №3.1 Описание программы)**.

Необходимые приложения: Требования к разработке.pdf

**Выполнение:**

**Разработка экранных форм**

1. В MS VS создать проект на WinForms и .NET

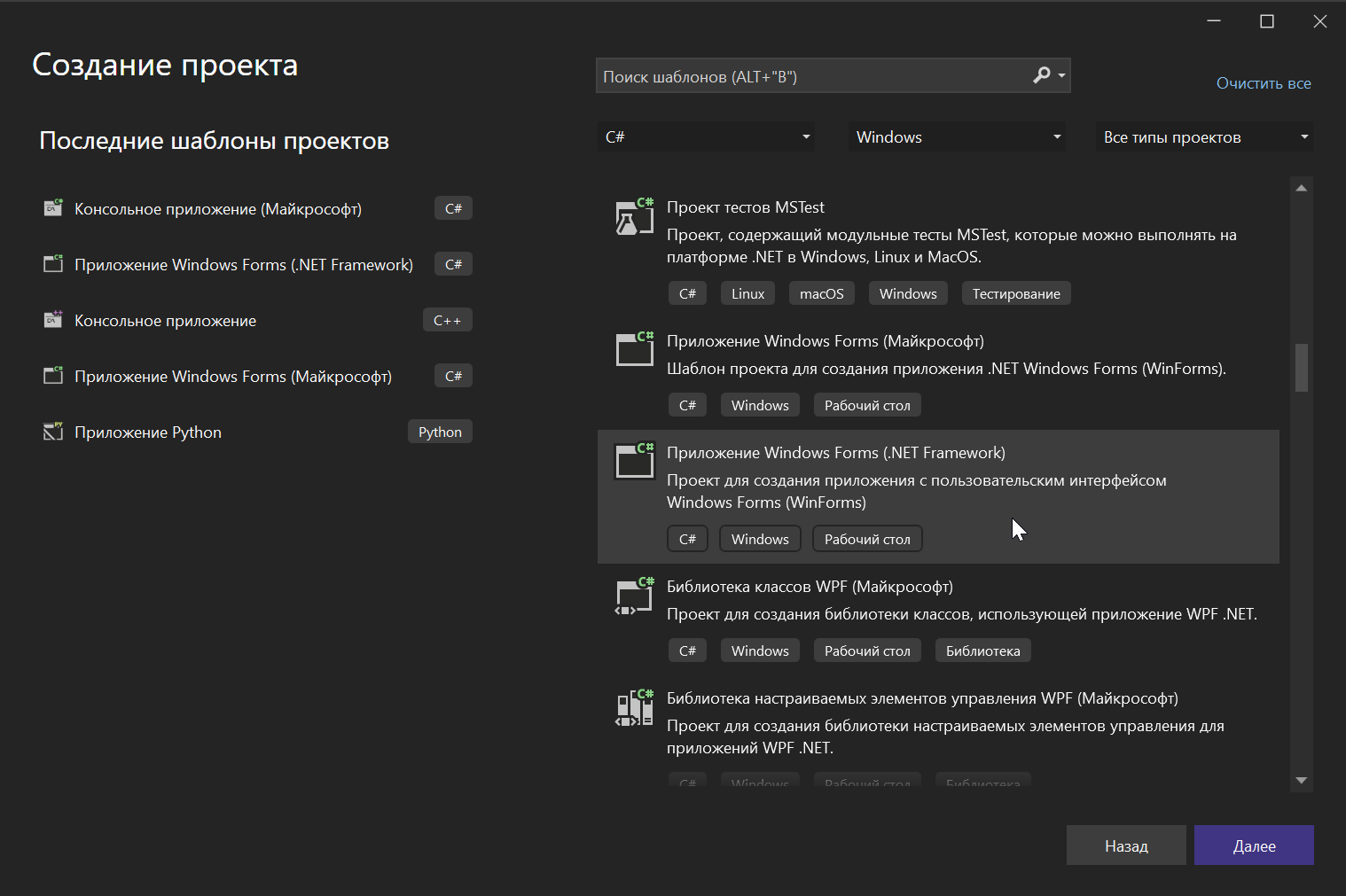


Рисунок 13 – Создание проекта

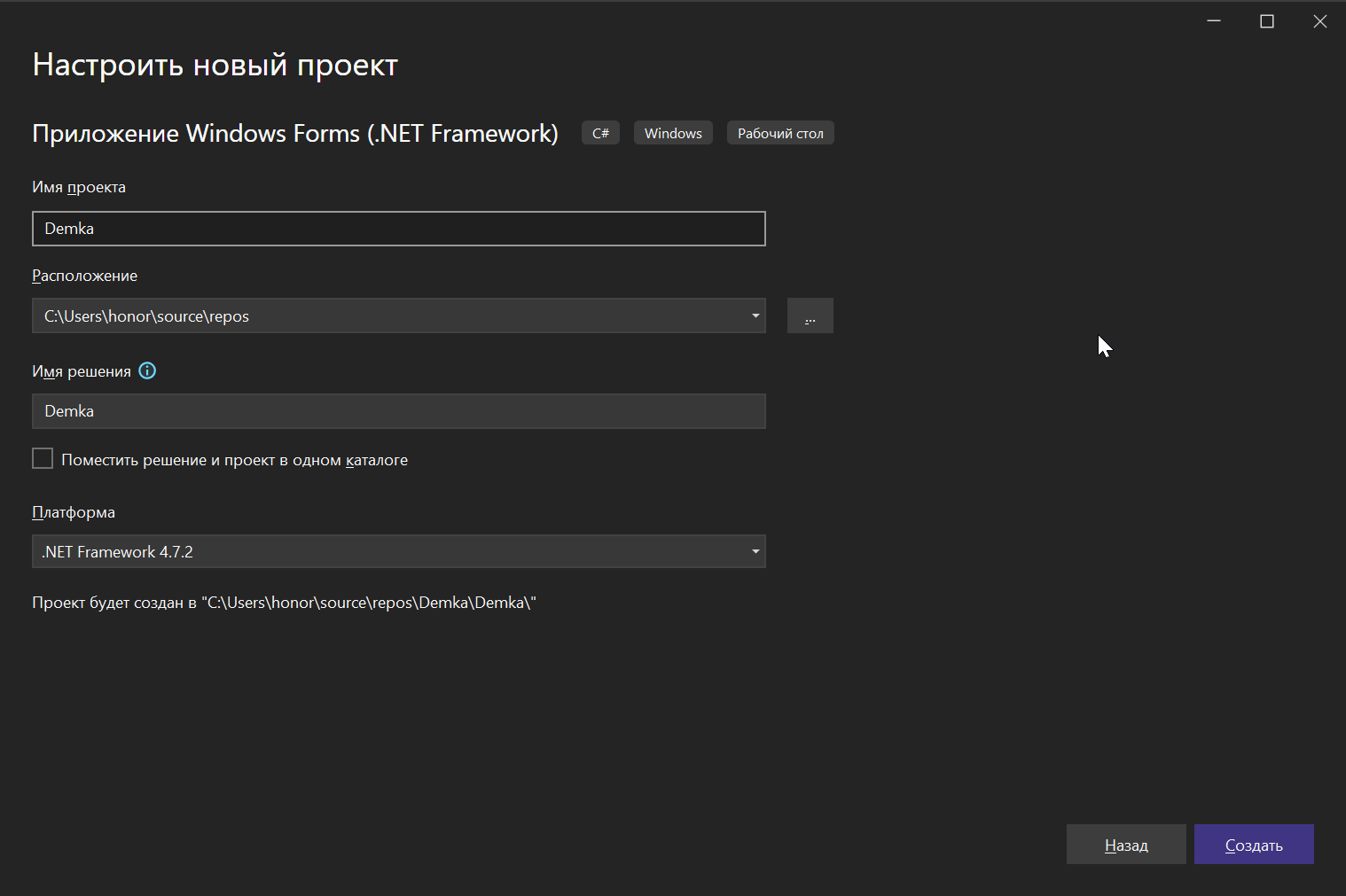


Рисунок 14 – Настройка проекта

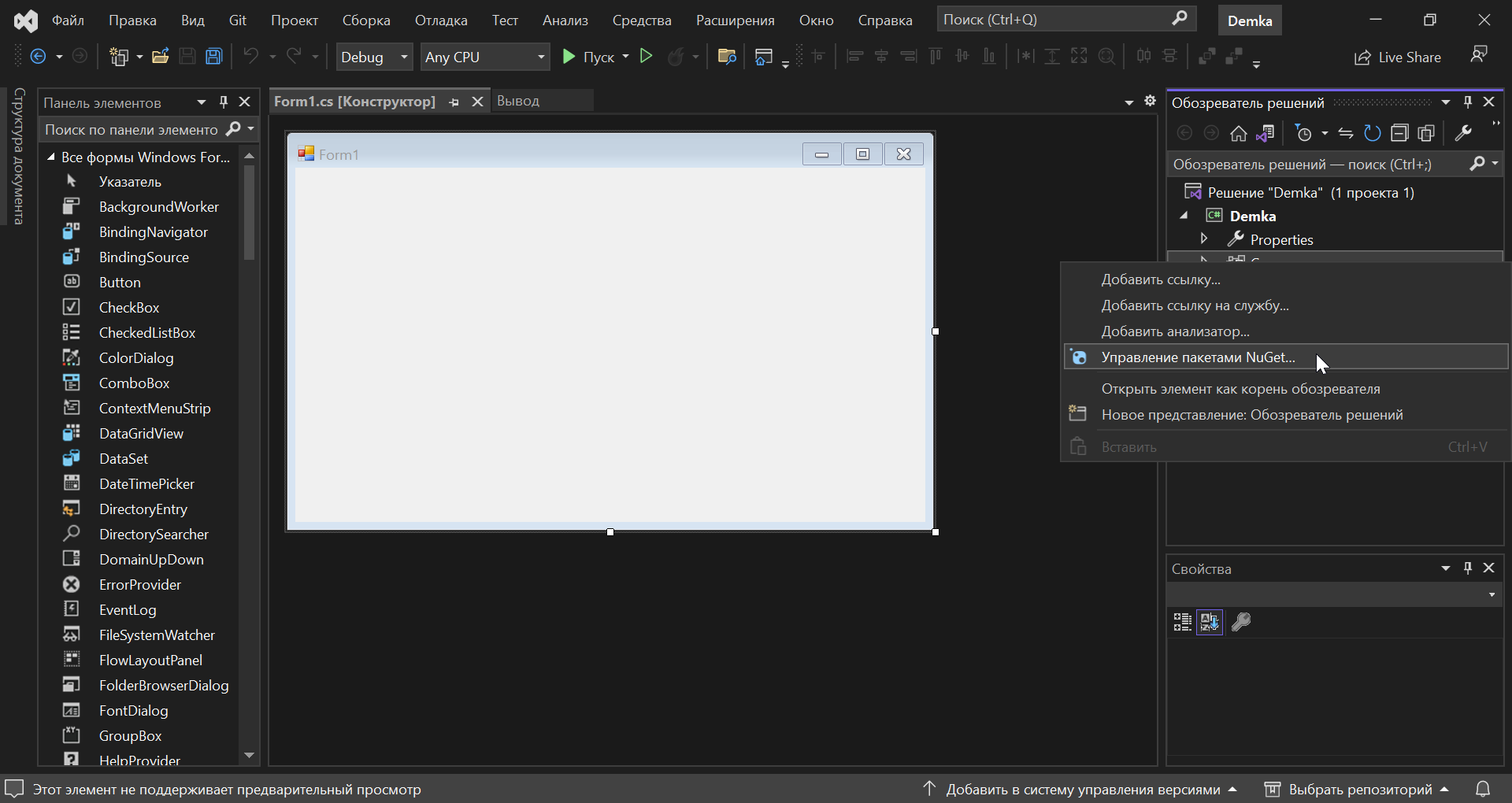


Рисунок 15 – Проверка настройки пакетов MySQL

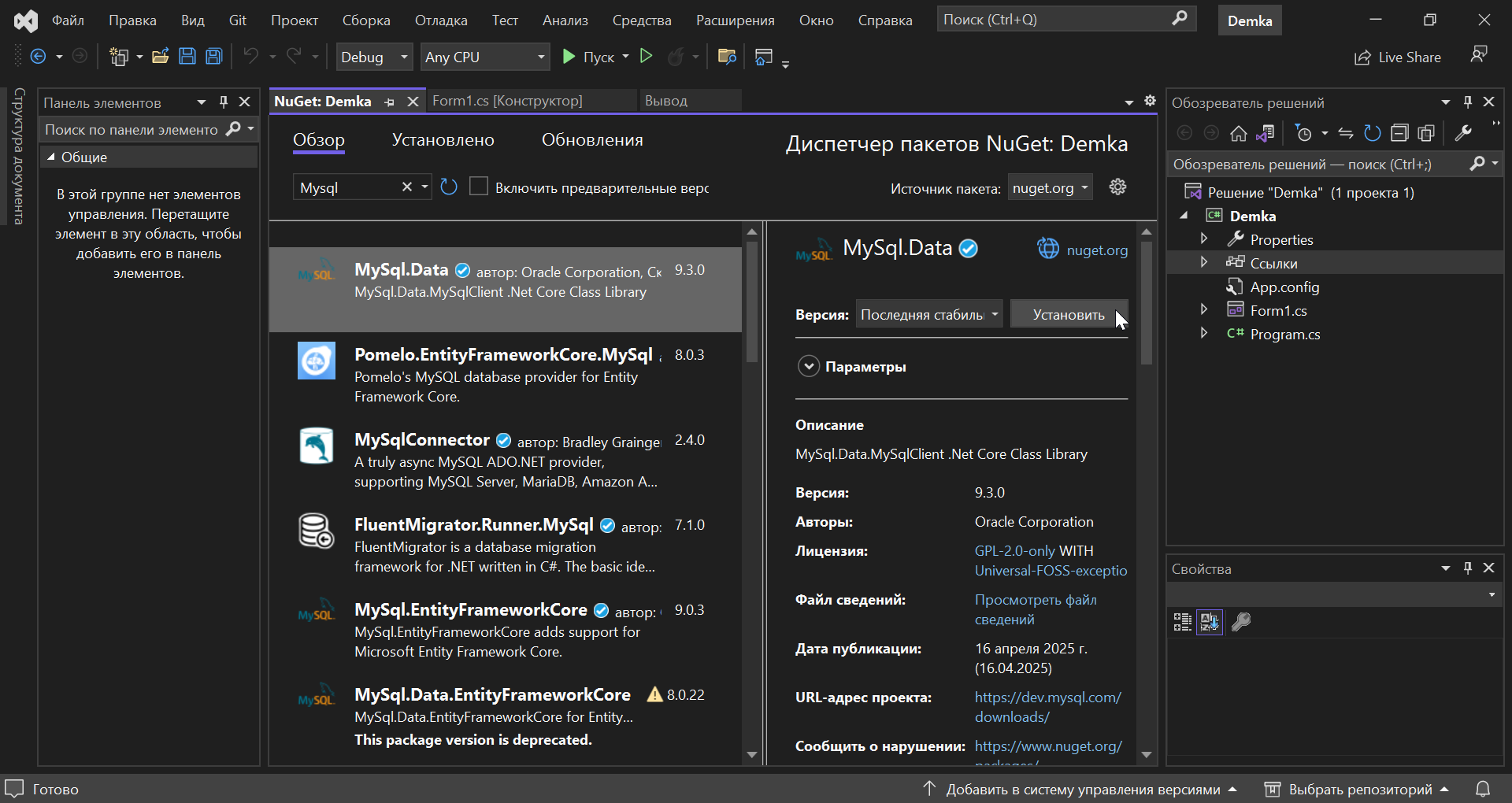


Рисунок 16 – Проверка настройки пакетов MySQL

1. Создать главное окно и меню вызова форм(menustrip)

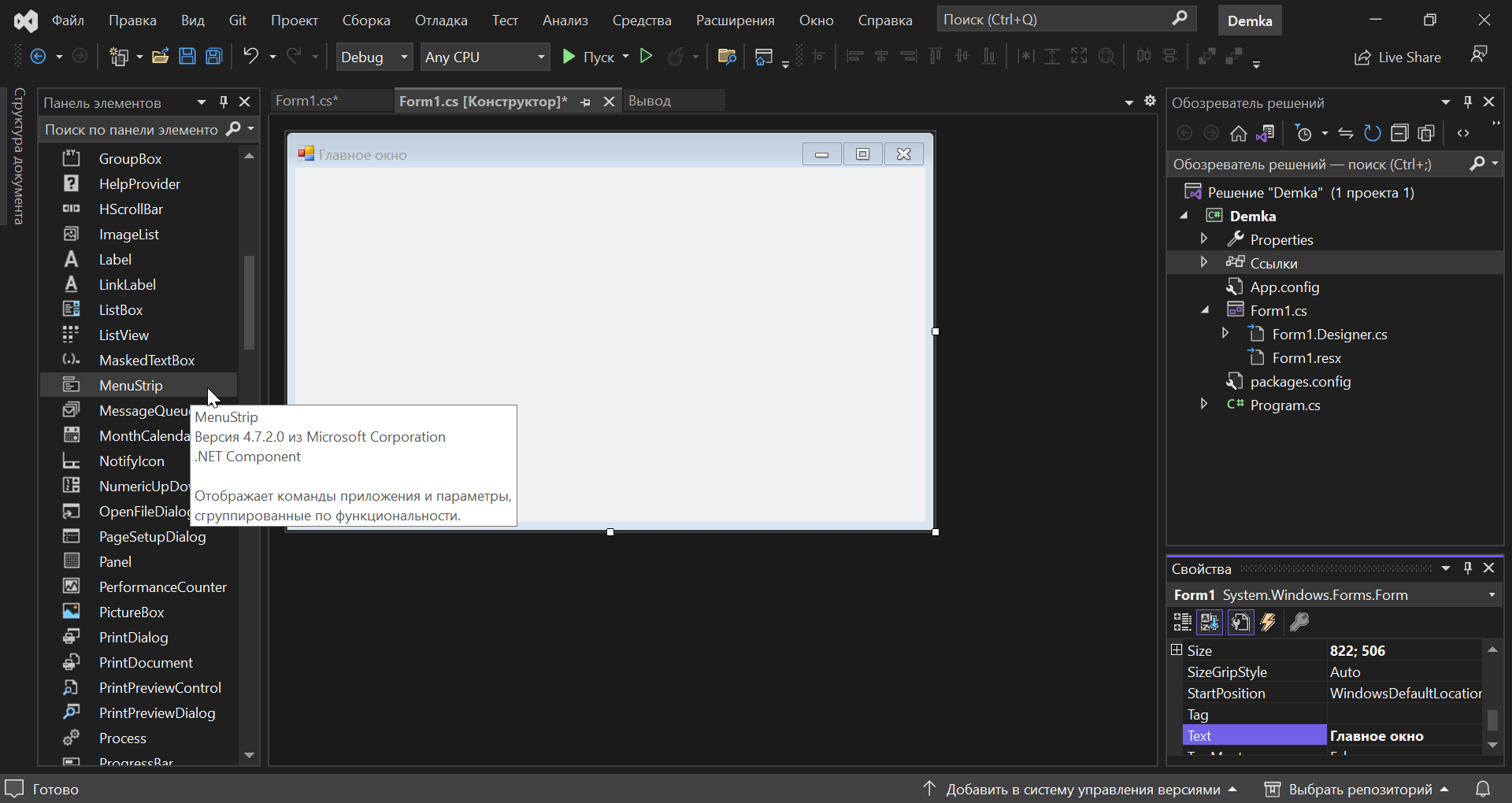


Рисунок 17 – Создание главного меню

1. В соответствии с заданной предметной областью разработать формы (с таблицами DataGrid соответствующими таблицам БД)

**Разработка ИС**

**НАЗВАНИЯ ДЛЯ ФОРМ:**

**Authorization -** авторизация

**ChangePasswordForm -** смена пароля

**Adminpanel -** форма администратора

**Mainmenu -** главное меню пользователя

**Validation** - тестирование

**SQL** - запрос

**RoomStock** - вывод таблицы

1. Связать графический интерфейс с БД.

Пример кода, обязательно использовать директиву using MySql.Data.MySqlClient;:

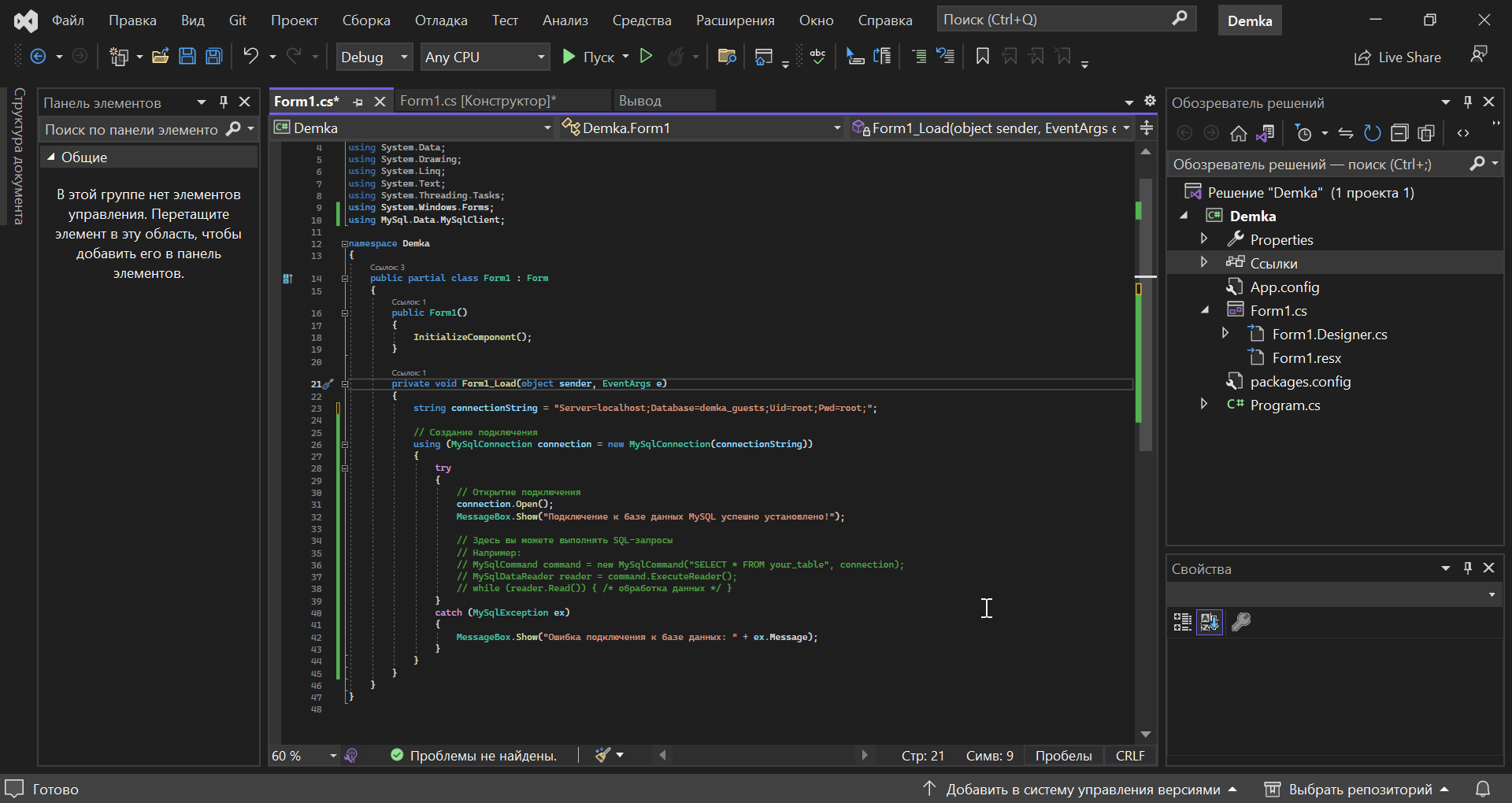


Рисунок 18 – Пример кода

Листинг 3. Подключение БД

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | private void Form1\_Load(object sender, EventArgs e)  {  string connectionString = "Server=localhost;Database=demka\_guests;Uid=root;Pwd=root;";  // Создание подключения  using (MySqlConnection connection = new MySqlConnection(connectionString))  {  try  {  // Открытие подключения  connection.Open();  MessageBox.Show("Подключение к базе данных MySQL успешно установлено!");  // Здесь вы можете выполнять SQL-запросы  // Например:  // MySqlCommand command = new MySqlCommand("SELECT \* FROM your\_table", connection);  // MySqlDataReader reader = command.ExecuteReader();    }  catch (MySqlException ex)  {  MessageBox.Show("Ошибка подключения к базе данных: " + ex.Message);  }  }  } |
| --- | --- |

1. Окно авторизации:

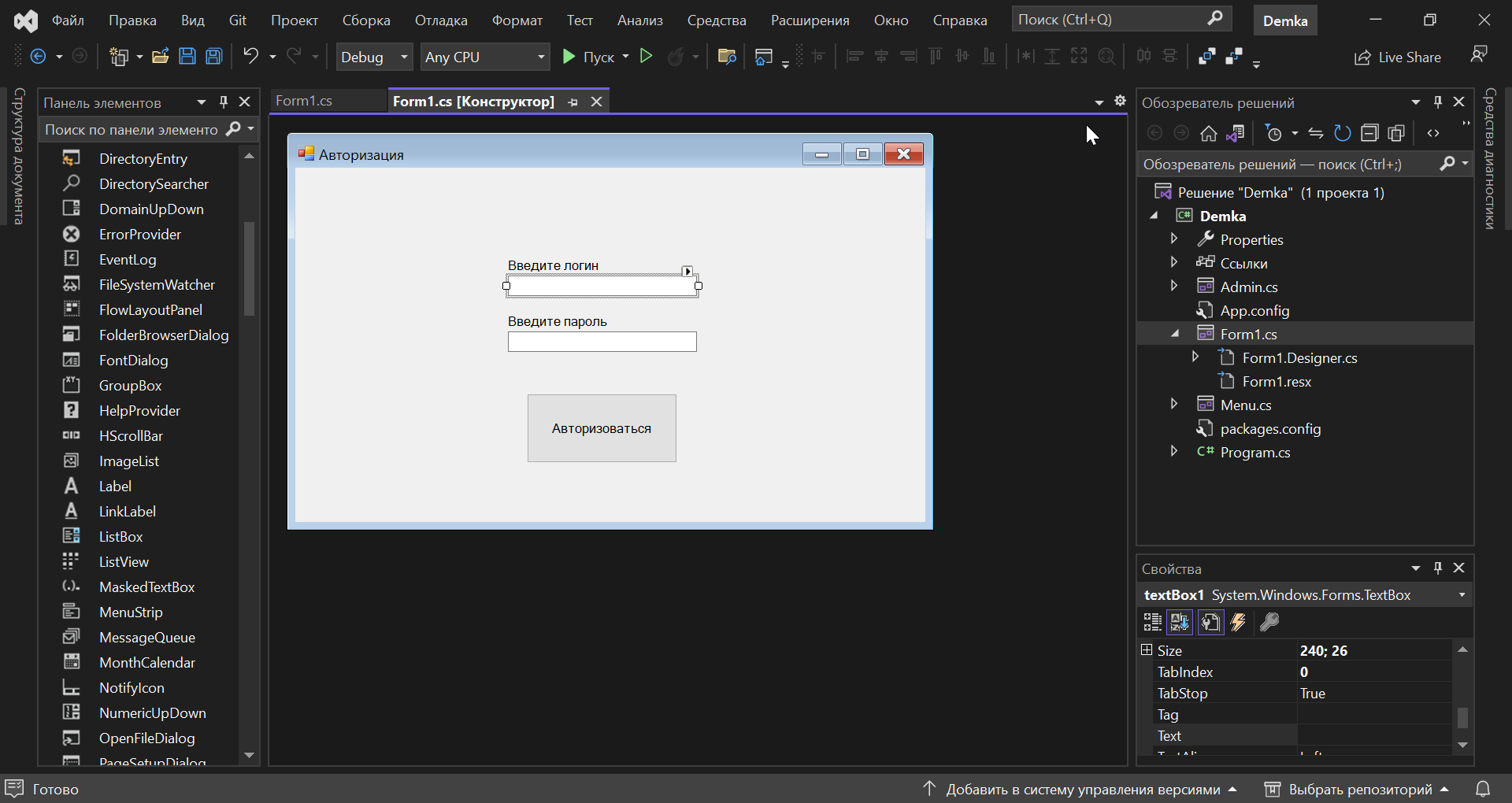


Рисунок 19 – Окно авторизации

Листинг 4. Пример кода для обработки окна авторизации

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204 | 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 MySql.Data.MySqlClient;  using static System.Windows.Forms.VisualStyles.VisualStyleElement;  namespace Demka  {  public partial class Form1 : Form  {  private readonly MySqlConnection \_conn = new MySqlConnection("Server=localhost;Database=demka\_guests;Uid=root;Pwd=root;");  private readonly MySqlDataAdapter \_adapter;  private readonly DataTable \_table = new DataTable();  public Form1()  {  InitializeComponent();  \_conn.Open();  \_adapter = new MySqlDataAdapter("SELECT \* FROM demka\_guests.users", \_conn);  RefreshData();  }  private void Form1\_Load(object sender, EventArgs e)  {  // Инициализация при загрузке формы  }  // Обработчик нажатия кнопки входа  private void button1\_Click(object sender, EventArgs e)  {  string login = txtlogin.Text;  string password = txtpassword.Text;  if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))  {  MessageBox.Show("Пожалуйста, заполните все поля.");  return;  }  if (AuthenticateUser(login, password, out bool isAdmin))  {  MessageBox.Show("Вы успешно авторизовались.");  CheckFirstLogin(login);  if (isAdmin)  {  ShowAdminPanel();  }  else  {  ShowMainForm();  }  }    }  // Аутентификация пользователя  private bool AuthenticateUser(string login, string password, out bool isAdmin)  {  isAdmin = false;  string query = "SELECT Role, Status, LastLogin FROM users WHERE Login = @Login AND Password = @Password";  using (MySqlConnection readConn = new MySqlConnection("Server=localhost;Database=demka\_guests;Uid=root;Pwd=root;"))  {  readConn.Open();  using (MySqlCommand command = new MySqlCommand(query, readConn))  {  command.Parameters.AddWithValue("@Login", login);  command.Parameters.AddWithValue("@Password", password);  using (MySqlDataReader reader = command.ExecuteReader())  {  if (reader.Read())  {  string role = reader.GetString("Role");  bool isBlocked = reader.GetBoolean("Status");  DateTime lastLogin = reader.GetDateTime("LastLogin");  if (isBlocked)  {  MessageBox.Show("Вы заблокированы. Обратитесь к администратору.");  return false;  }  if (lastLogin < DateTime.Now.AddMonths(-1))  {  BlockUser(login);  MessageBox.Show("Ваш аккаунт был заблокирован из-за длительного отсутствия активности. Обратитесь к администратору.");  return false;  }  isAdmin = (role == "Администратор");  UpdateLastLogin(login);  return true;  }  else  {  MessageBox.Show("Вы ввели неверный логин или пароль. Пожалуйста, проверьте ещё раз введенные данные.");  IncrementLoginAttempts(login);  return false;  }  }  }  }  }  private void IncrementLoginAttempts(string login)  {  string query = "UPDATE users SET Login\_attempts = Login\_attempts + 1 WHERE Login = @Login";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@Login", login);  command.ExecuteNonQuery();  }  if (GetLoginAttempts(login) >= 3)  {  BlockUser(login);  }  }  private int GetLoginAttempts(string login)  {  string query = "SELECT Login\_attempts FROM users WHERE Login = @Login";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@Login", login);  return Convert.ToInt32(command.ExecuteScalar());  }  }  private void BlockUser(string login)  {  string query = "UPDATE users SET Status = 1 WHERE Login = @Login";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@Login", login);  command.ExecuteNonQuery();  }  }  private void CheckFirstLogin(string login)  {  string query = "SELECT IsFirstLogin FROM users WHERE Login = @Login";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@Login", login);  bool isFirstLogin = Convert.ToBoolean(command.ExecuteScalar());  if (isFirstLogin)  {  ShowChangePasswordForm(login);  }  }  }  private void ShowChangePasswordForm(string login)  {  // Открыть форму для смены пароля  ChangePasswordForm changePasswordForm = new ChangePasswordForm(login, \_conn);  changePasswordForm.ShowDialog();  }  private void RefreshData()  {  \_table.Clear();  \_adapter.Fill(\_table);  }  private void ShowAdminPanel()  {  // Открыть форму администратора  Adminpanel adminPanel = new Adminpanel();  adminPanel.ShowDialog();  }  private void ShowMainForm()  {  // Открыть главное окно для пользователя  Mainmenu mainForm = new Mainmenu();  mainForm.ShowDialog();  }  private void UpdateLastLogin(string login)  {  string query = "UPDATE users SET LastLogin = @LastLogin, Login\_attempts = 0 WHERE Login = @Login";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@LastLogin", DateTime.Now);  command.Parameters.AddWithValue("@Login", login);  command.ExecuteNonQuery();  }  }  }    } |
| --- | --- |

1. Смена пароля:

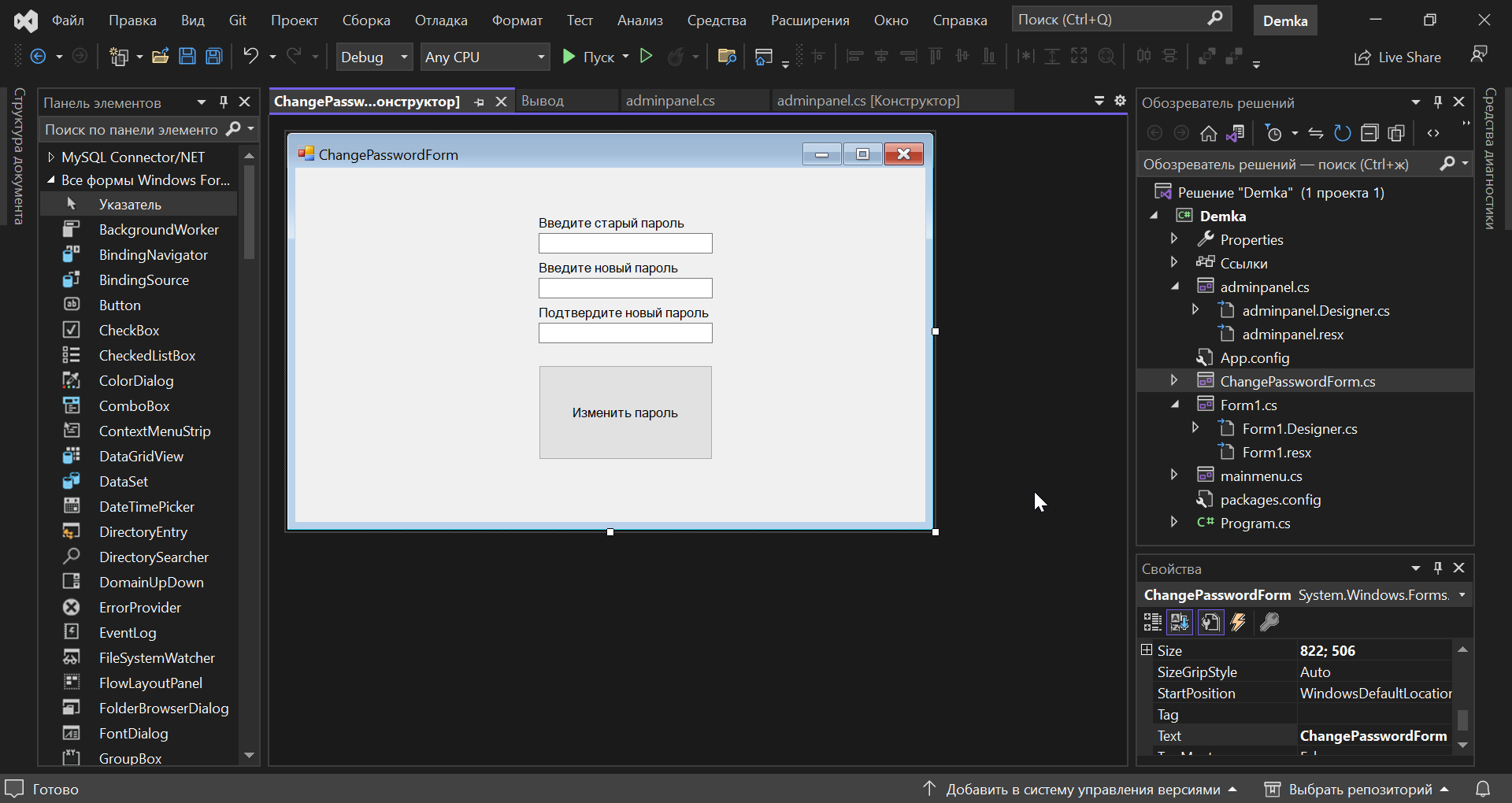


Рисунок 20 – Смена пароля

Листинг 5. Код для формы смена пароля

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 20  21 22  23  24  25  26  27  28  29 30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89 | using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using MySql.Data.MySqlClient;  using System.Threading.Tasks;  using System.Windows.Forms;  using static System.Windows.Forms.VisualStyles.VisualStyleElement;  namespace Demka  {  public partial class ChangePasswordForm : Form  {  private readonly string \_login;  private readonly MySqlConnection \_conn;  public ChangePasswordForm(string login, MySqlConnection conn)  {  InitializeComponent();  \_login = login;  \_conn = conn;  }  private void ChangePasswordForm\_Load(object sender, EventArgs e)  {  }  private void button1\_Click(object sender, EventArgs e)  {  string currentPassword = textBox1.Text;  string newPassword = textBox2.Text;  string confirmPassword = textBox3.Text;  if (string.IsNullOrEmpty(currentPassword) || string.IsNullOrEmpty(newPassword) || string.IsNullOrEmpty(confirmPassword))  {  MessageBox.Show("Пожалуйста, заполните все поля.");  return;  }  if (newPassword != confirmPassword)  {  MessageBox.Show("Новый пароль и его подтверждение не совпадают.");  return;  }  if (VerifyCurrentPassword(currentPassword))  {  UpdatePassword(newPassword);  MessageBox.Show("Пароль успешно изменен.");  this.Close();  }  else  {  MessageBox.Show("Текущий пароль введен неверно.");  }  }  private bool VerifyCurrentPassword(string currentPassword)  {  string query = "SELECT COUNT(\*) FROM users WHERE Login = @Login AND Password = @Password";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@Login", \_login);  command.Parameters.AddWithValue("@Password", currentPassword);  return Convert.ToInt32(command.ExecuteScalar()) > 0;  }  }  private void UpdatePassword(string newPassword)  {  string query = "UPDATE users SET Password = @Password, IsFirstLogin = 0 WHERE Login = @Login";  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  command.Parameters.AddWithValue("@Password", newPassword);  command.Parameters.AddWithValue("@Login", \_login);  command.ExecuteNonQuery();  }  }  }  } |
| --- | --- |

1. Форма администратора:

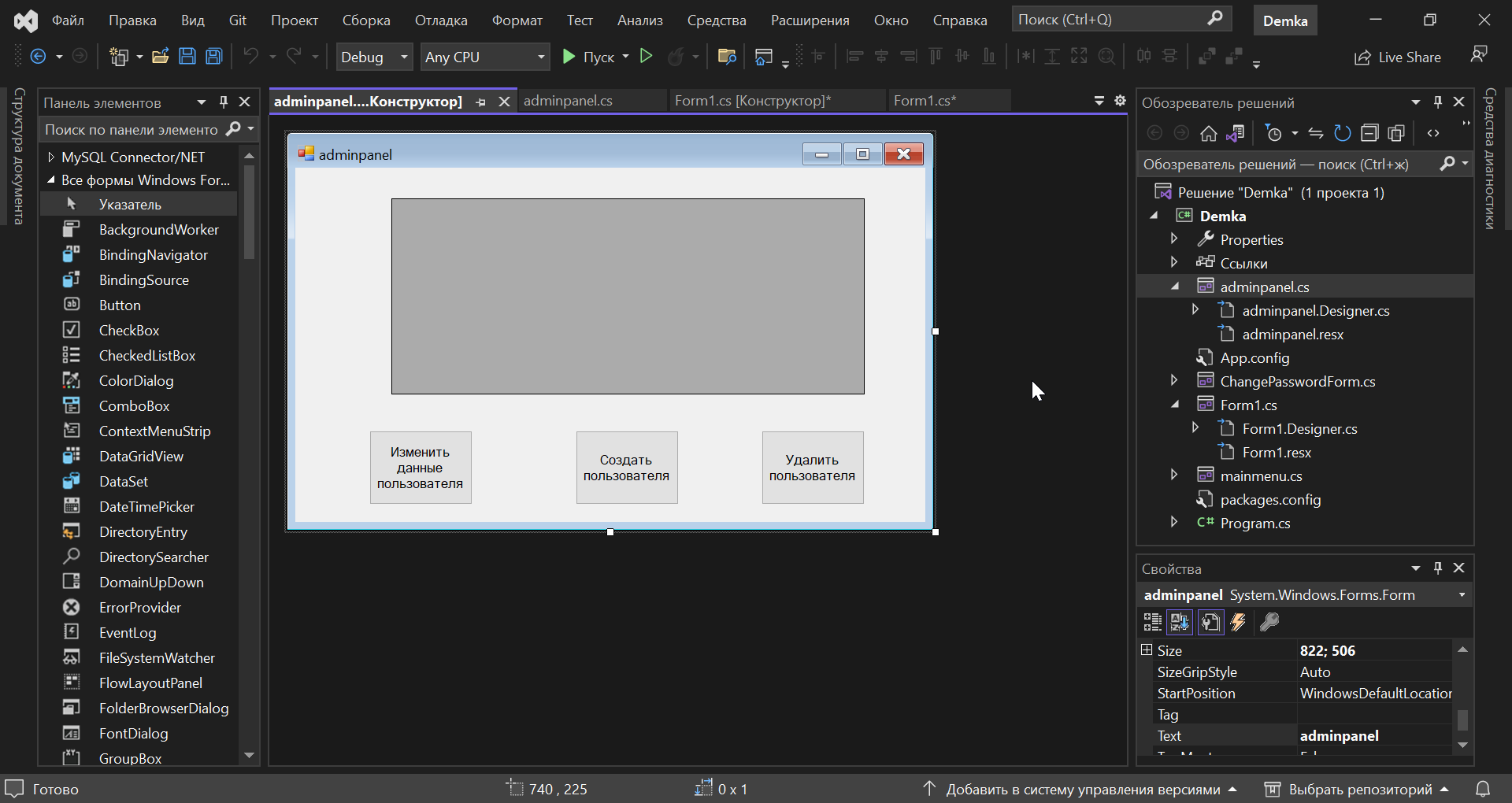


Рисунок 21 – Форма администратора

Листинг 6. Код формы администратора

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 20  21 22  23  24  25  26  27  28  29 30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140141  142  143  144  145  146  147  148  149 | using MySql.Data.MySqlClient;  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 Demka  {  public partial class adminpanel : Form  {  private readonly MySqlConnection \_conn = new MySqlConnection("Server=localhost;Database=demka\_guests;Uid=root;Pwd=root;");  private readonly MySqlDataAdapter \_adapter;  private readonly DataTable \_table = new DataTable();  public adminpanel()  {  InitializeComponent();  \_conn.Open();  \_adapter = new MySqlDataAdapter("SELECT \* FROM demka\_guests.users", \_conn);  dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;  RefreshData();  }  private bool LoginExists(string login)  {  using (MySqlCommand command = new MySqlCommand("SELECT COUNT(\*) FROM users WHERE Login = @Login", \_conn))  {  command.Parameters.AddWithValue("@Login", login);  int count = Convert.ToInt32(command.ExecuteScalar());  return count > 0;  }  }  private void UpdateUser(int idUsers, string login, string password, string role, bool status, int loginAttempts, DateTime lastLogin, bool isFirstLogin)  {  ExecuteCommand("UPDATE users SET Login = @Login, Password = @Password, Role = @Role, Status = @Status, Login\_attempts = @Login\_attempts, LastLogin = @LastLogin, IsFirstLogin = @IsFirstLogin WHERE idUsers = @idUsers",  new[] { "@Login", "@Password", "@Role", "@Status", "@Login\_attempts", "@LastLogin", "@IsFirstLogin", "@idUsers" },  new object[] { login, password, role, status, loginAttempts, lastLogin, isFirstLogin, idUsers });  }  private void AddUser(string login, string password, string role, bool status, int loginAttempts, DateTime lastLogin, bool isFirstLogin)  {  if (LoginExists(login))  {  MessageBox.Show("Пользователь с таким логином уже существует.");  return;  }  ExecuteCommand("INSERT INTO users (Login, Password, Role, Status, Login\_attempts, LastLogin, IsFirstLogin) VALUES (@Login, @Password, @Role, @Status, @Login\_attempts, @LastLogin, @IsFirstLogin)",  new[] { "@Login", "@Password", "@Role", "@Status", "@Login\_attempts", "@LastLogin", "@IsFirstLogin" },  new object[] { login, password, role, status, loginAttempts, lastLogin, isFirstLogin });  }  private void DeleteUser(int idUsers)  {  ExecuteCommand("DELETE FROM users WHERE idUsers = @idUsers", new[] { "@idUsers" }, new object[] { idUsers });  }  private void ExecuteCommand(string query, string[] parameterNames, object[] parameterValues)  {  using (MySqlCommand command = new MySqlCommand(query, \_conn))  {  for (int i = 0; i < parameterNames.Length; i++)  {  command.Parameters.AddWithValue(parameterNames[i], parameterValues[i]);  }  command.ExecuteNonQuery();  RefreshData();  }  }  private void RefreshData()  {  \_table.Clear();  \_adapter.Fill(\_table);  dataGridView1.DataSource = \_table;  }  private void adminpanel\_Load(object sender, EventArgs e)  {  }  private void button1\_Click(object sender, EventArgs e)  {  if (dataGridView1.SelectedRows.Count > 0)  {  var row = dataGridView1.SelectedRows[0];  UpdateUser(  Convert.ToInt32(row.Cells["idUsers"].Value),  row.Cells["Login"].Value?.ToString(),  row.Cells["Password"].Value?.ToString(),  row.Cells["Role"].Value?.ToString(),  Convert.ToBoolean(row.Cells["Status"].Value),  Convert.ToInt32(row.Cells["Login\_attempts"].Value),  Convert.ToDateTime(row.Cells["LastLogin"].Value),  Convert.ToBoolean(row.Cells["IsFirstLogin"].Value)  );  }  }  private void button2\_Click(object sender, EventArgs e)  {  var newRow = dataGridView1.Rows[dataGridView1.NewRowIndex - 1];  if (newRow.IsNewRow) return;  var login = newRow.Cells["Login"].Value?.ToString();  var password = newRow.Cells["Password"].Value?.ToString();  if (!string.IsNullOrEmpty(login) && !string.IsNullOrEmpty(password))  {  AddUser(  login,  password,  newRow.Cells["Role"].Value?.ToString(),  Convert.ToBoolean(newRow.Cells["Status"].Value),  Convert.ToInt32(newRow.Cells["Login\_attempts"].Value),  Convert.ToDateTime(newRow.Cells["LastLogin"].Value),  Convert.ToBoolean(newRow.Cells["IsFirstLogin"].Value)  );  }  }  private void button3\_Click(object sender, EventArgs e)  {  if (dataGridView1.SelectedRows.Count > 0)  {  DeleteUser((int)dataGridView1.SelectedRows[0].Cells["idUsers"].Value);  }  }  }  } |
| --- | --- |

1. Главное меню пользователя

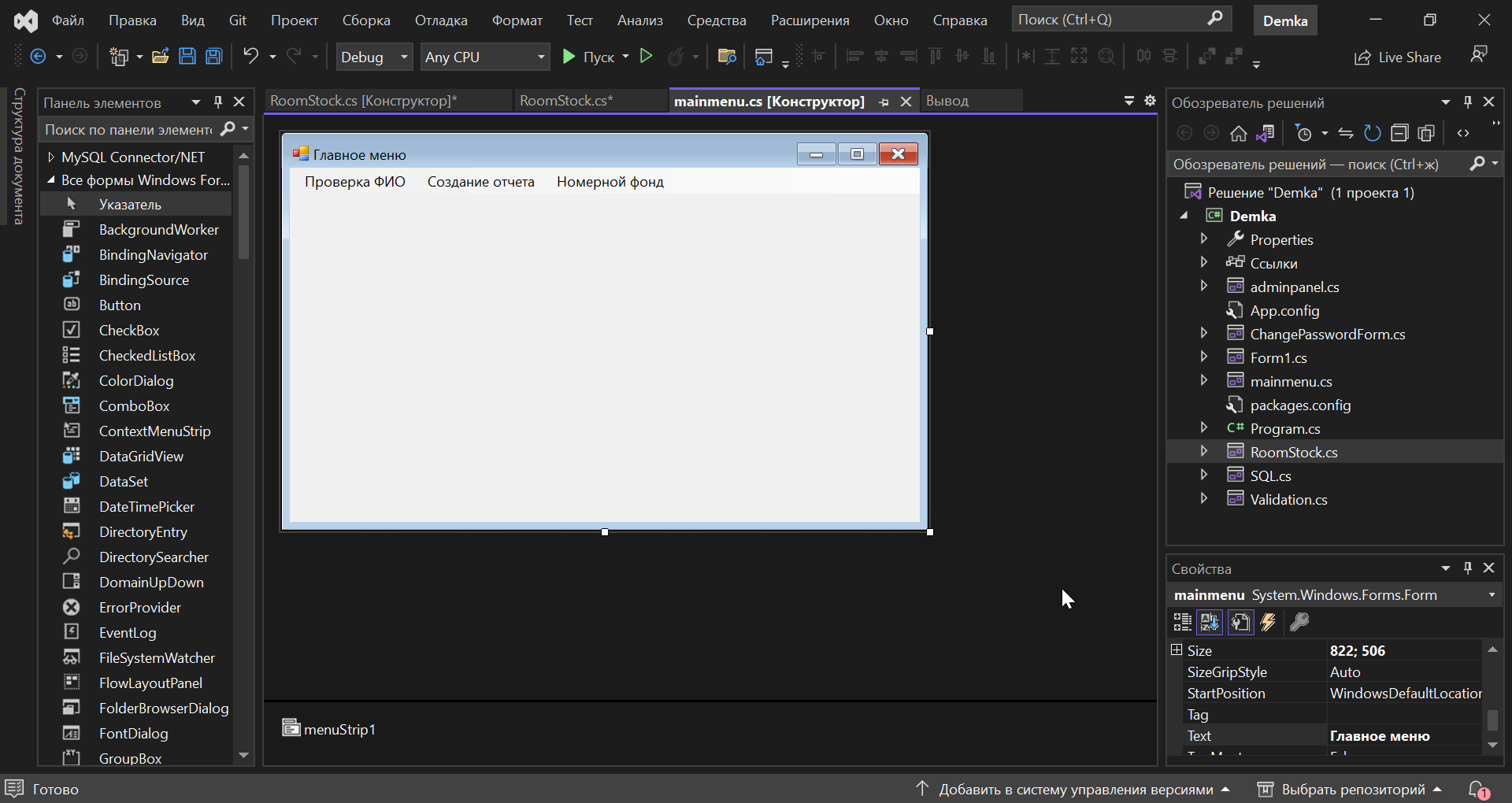
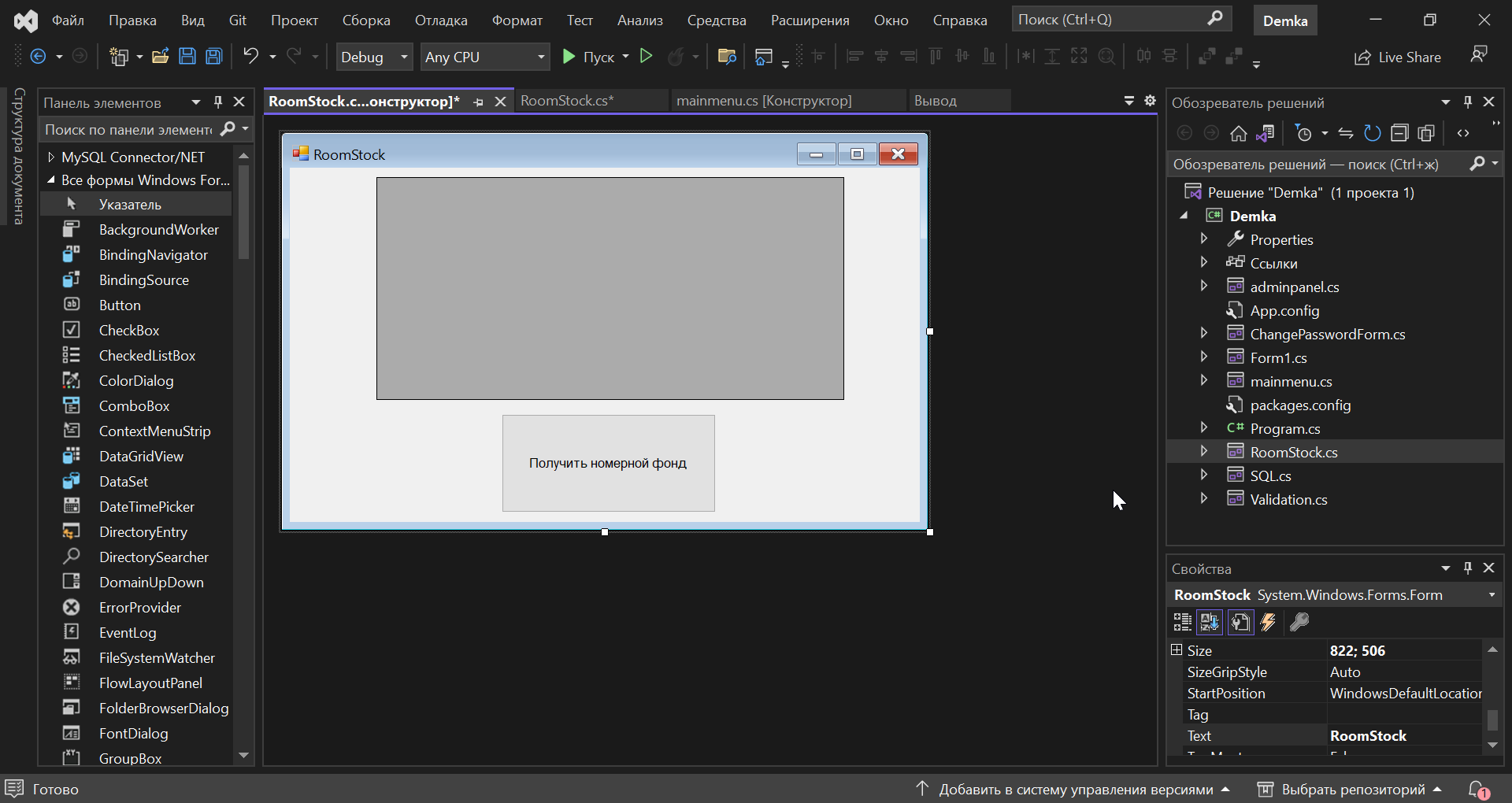


Рисунок 22 – Главное меню пользователя

Листинг 7. Код для формы главного меню

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 20  21 22  23  24  25  26  27  28  29 30  31  32  33  34  35  36  37  38  39  40  41  42  42  44 | 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 Demka  {  public partial class mainmenu : Form  {  public mainmenu()  {  InitializeComponent();  }  private void проверкаФИОToolStripMenuItem\_Click(object sender, EventArgs e) // два раза кликнуть по пункту в MenuStrip  {  Validation mainForm = new Validation();  mainForm.ShowDialog(); // Скопировать вот это для того чтобы сделать переход на форму  }  private void mainmenu\_Load(object sender, EventArgs e)  {  }  private void sQLзапросToolStripMenuItem\_Click(object sender, EventArgs e)  {  SQL sqlform = new SQL();  sqlform.ShowDialog(); // Скопировать вот это для того чтобы сделать переход на форму  }  private void номернойФондToolStripMenuItem\_Click(object sender, EventArgs e)  {  RoomStock room = new RoomStock();  room.ShowDialog();  }  }  } |
| --- | --- |



Листинг 7.1. Меню пользователя

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 20  21 22  23  24  25  26  27  28  29 30  31  32  33  34  35  36  37  38  39  40  41  42  42  44  45  46  47  48  49  50  51  52  53  54  55 | using MySql.Data.MySqlClient;  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 Demka  {  public partial class RoomStock : Form  {  private readonly MySqlConnection \_conn;  public RoomStock()  {  InitializeComponent();  \_conn = new MySqlConnection("Server=localhost;Database=demka\_guests;Uid=root;Pwd=root;");  }  private void LoadData()  {  try  {  // Получаем данные о загрузке номеров из БД  DataTable table = GetData();  // Устанавливаем источник данных для DataGridView  dataGridView1.DataSource = table;  // Настраиваем автоматическое изменение ширины столбцов  dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;  // Запрещаем редактирование данных  dataGridView1.ReadOnly = true;  // Скрываем заголовки строк  dataGridView1.RowHeadersVisible = false;  }  catch (Exception ex)  {  // Выводим сообщение об ошибке, если что-то пошло не так  MessageBox.Show("Ошибка: " + ex.Message);  }  }  // Метод для получения данных о загрузке номеров из базы данных  private DataTable GetData()  {  string sql = @"SELECT Floor AS 'Этаж', Number 'Номер', Category 'Категория' FROM demka\_guests.room\_stock";  // Создаем и открываем подключение к БД  using (var conn = new MySqlConnection(\_conn.ConnectionString))  {  // Создаем команду для выполнения SQL-запроса  MySqlCommand cmd = new MySqlCommand(sql, conn);  // Создаем адаптер для работы с данными  MySqlDataAdapter adapter = new MySqlDataAdapter(cmd);  // Создаем таблицу для хранения результатов  DataTable table = new DataTable();  // Открываем соединение с БД  conn.Open();  // Заполняем таблицу данными из БД  adapter.Fill(table);  // Возвращаем заполненную таблицу  return table;  }  }  private void button1\_Click(object sender, EventArgs e)  {  LoadData();  }    }  } |
| --- | --- |

## 

## Модуль №3.1 Описание системы

Разработайте проектную документацию на разработанный функционал. Включите описание функционального назначения, используемые методы с указанием параметров.

[«Общее описание системы» по ГОСТ 34](https://technicaldocs.ru/%D0%B3%D0%BE%D1%81%D1%8234/%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B/%D0%BE%D0%B1%D1%89%D0%B5%D0%B5_%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%B8%D0%B5_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B#%D1%84%D0%B0%D0%B9%D0%BB%D1%8B)

**Выполнение:**

1. Создать документ в Google или Word
2. Сделать титульник с названием (см. Приложение А)
3. Сделать скрины экранных форм , вставить в док и описать

## 

## Модуль №4 Осуществление интеграции программных модулей

Текст задания: Для проверки данных от клиентов разработайте приложение, которое позволит провести валидацию на корректность данных. Результат проверки необходимо фиксировать в документе ТестКейс.docx. 30 Сначала заполните в документе ТестКейс.docx столбец "Действие" и "Ожидаемый результат" используя предоставленный текстовый редактор. Добавьте закладки в столбец "Результат". Необходимо провести валидацию ФИО клиента на вхождение запрещенных символов. Проверьте два любых критерия. Для эмуляции отправки данных от клиента Вам необходимо запустить приложение TransferSimulator.exe. Методы эмулятора описаны в файле api\_info.pdf. Макет формы представлен на рисунке. Рисунок - Макет окна приложения валидации данных При нажатии на кнопку "Получить данные" данные загружаются с эмулятора и отображаются на форме. После нажатия на кнопку "Отправить результат теста" происходит проверка данных по заполненному шаблону, и результат проверки отображается на форме и в соответствующей строке таблицы в столбце "Результат". Важно: Разрабатывать API Вам не нужно. Используйте предоставленный API из приложения.

Необходимые приложения: api\_info.pdf, TransferSimulator.exe, ТестКейс.docx, Требования к разработке.pdf, Настройка ПК для эмулятора.pdf

**Выполнение:**

**Тестирование**

1. Выбрать вариант тестирования

Функциональное тестирование

Проверяет, выполняет ли система заявленные требования. Фокусируется на внешнем поведении программы: как она реагирует на ввод пользователя, как выполняются определенные операции.

Некоторые цели функционального тестирования:

* Проверка соответствия требованиям. Убедиться, что все функции и компоненты системы работают в соответствии с спецификациями.
* Обнаружение дефектов. Найти и устранить ошибки, которые могут повлиять на работу системы.
* Проверка пользовательского интерфейса. Убедиться, что интерфейс работает корректно и интуитивно понятен пользователям.

1. Создать таблицу см. пример на рисунке ниже (но у вас свои тест- кейсы должны быть). Описать 2-3 тест-кейса

Таблица 1. Тест-кейсы

| Дата | Номер тест-кейса | Заголовок | Предусловие | Шаги проверки | Ожидаемый результат | Фактический результат | Статус |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |



Рисунок 23 – Пример заполненной таблицы

Таблица 2. Готовые тест-кейсы

| Дата | Номер тест-кейса | Заголовок | Предусловие | Шаги проверки | Ожидаемый результат | Фактический результат | Статус |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 17.06.2025 | 1 | Вход с ролью “Администратор” | Проверка входа пользователя с ролью “Администратор” | Ввод корректного логина и пароля. Нажать на кнопку “Авторизация” | Успешная авторизация, переход на главное меню Администратора | Успешная авторизация, переход на главное меню Администратора | Пройден |
| 17.06.2025 | 2 | Вход с ролью “Пользователь” | Проверка входа пользователя с ролью “Пользователь” | Ввод корректного логина и пароля. Нажать на кнопку “Авторизация” | Успешная авторизация, переход на главное меню Пользователя | Успешная авторизация, переход на главное меню Пользователя | Пройден |
| 17.06.2025 | 3 | Нажать кнопку “Удалить пользователя” | Проверка функции администратора “Удалить пользователя” | Войти в систему с ролью Администратор. Выбрать пользователя.  Нажать кнопку “Удалить пользователя” | Успешное удаление пользователя | Успешное удаление пользователя | Пройден |
| 17.06.2025 | 4 |  |  |  |  |  | Пройден |
| 17.06.2025 | 5 |  |  |  |  |  | Пройден |
| 17.06.2025 | 6 |  |  |  |  |  | Пройден |

Статус тест-кейсов:

* Не проверен – Статус по умолчанию, который установлен для всех тест-кейсов и шагов при размещении их в прогоне.
* Пройден – Статус, который означает полностью корректное выполнение шага / тест-кейса.
* Провален – Статус, который означает, что шаг / тест-кейс был проверен, но обнаружилась ошибка, из-за которой не был получен корректный результат.
* Заблокирован – Статус, который означает, что не удалось пройти проверку данного шага / тест-кейса.
* Ретест – Статус, который означает, что необходимо будет вернуться к данному шагу / тест-кейсу - в данный момент это нецелесообразно.

Реализация теста:

Format.OpenXml

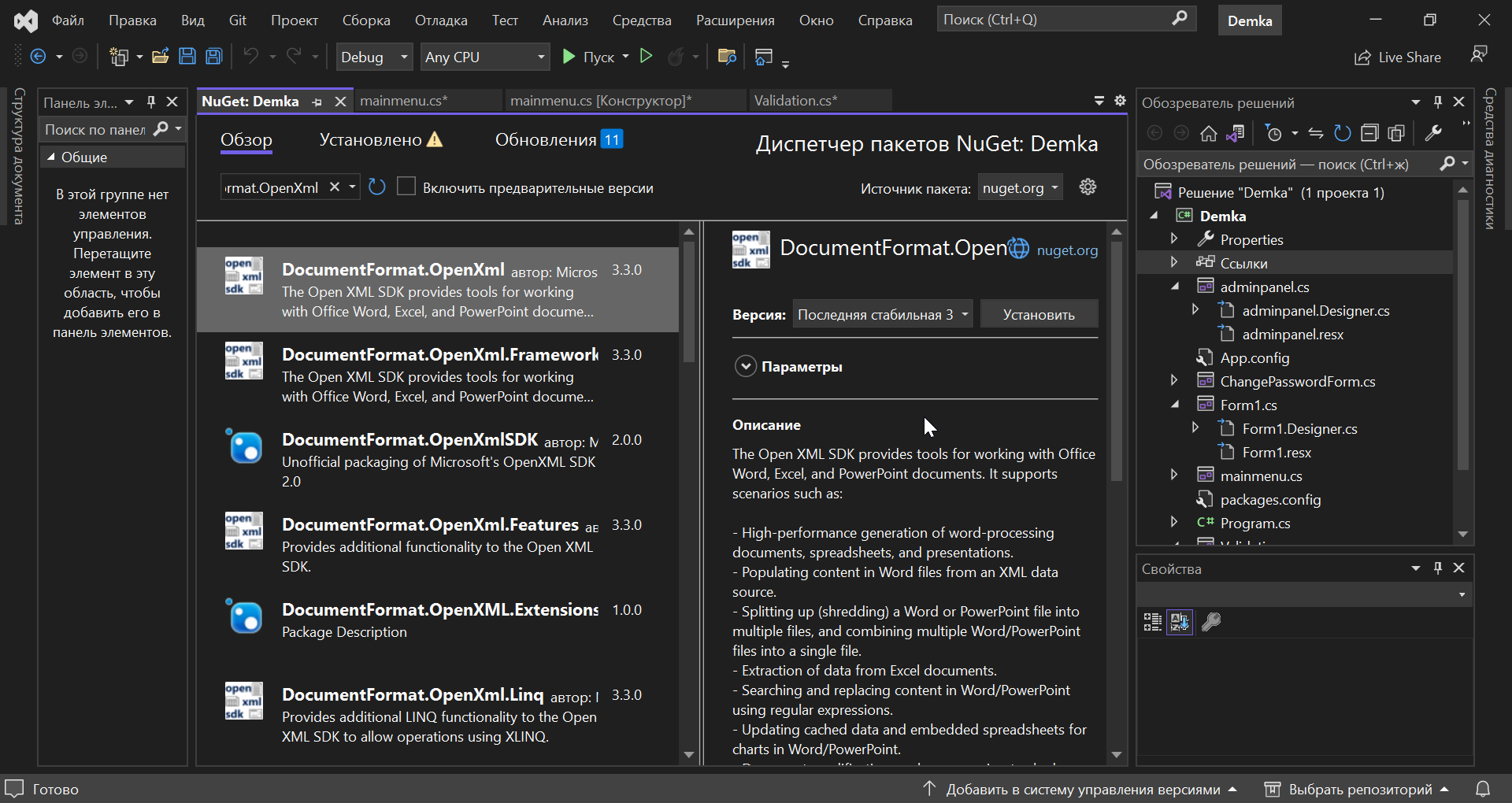


Рисунок 24 – Скачивание библиотеки ворд

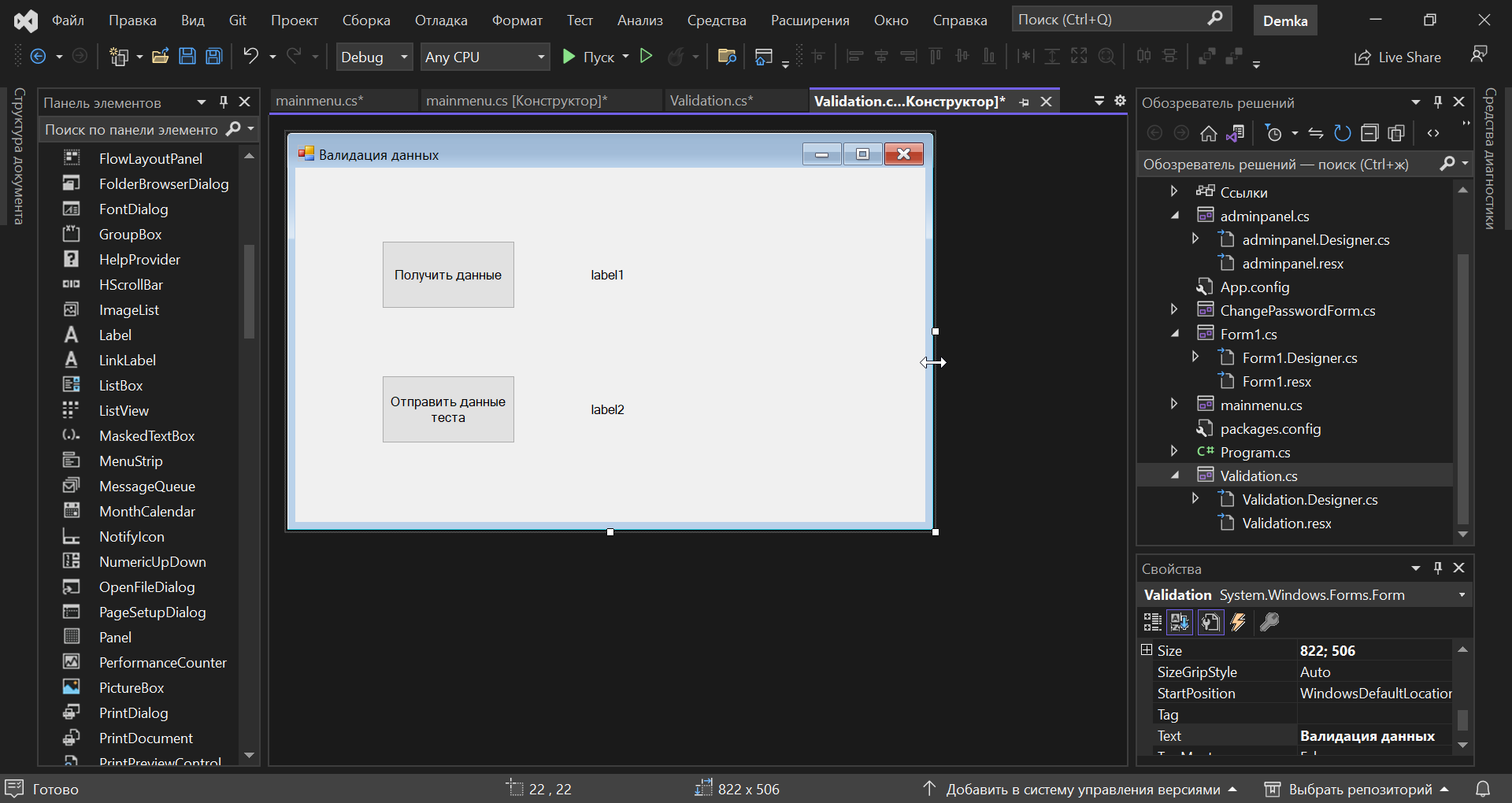


Рисунок 25 – Форма валидации данных

Листинг 8. Код формы валидации данных

| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209 | using MySqlX.XDevAPI.Relational;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.IO;  using System.Linq;  using System.Net;  using System.Net.Http;  using System.Text;  using System.Text.RegularExpressions;  using System.Threading.Tasks;  using System.Windows.Forms;  using DocumentFormat.OpenXml;  using DocumentFormat.OpenXml.Packaging;  using DocumentFormat.OpenXml.Wordprocessing;  using WordTable = DocumentFormat.OpenXml.Wordprocessing.Table; // Замена переменной Table для устранения неоднозначности(Table в Mysql и Table в Word)  namespace Demka  {  public partial class Validation : Form  {  private HttpClient httpClient;  private const string TestCasePath = @"C:\Users\honor\Desktop\КОД 09.02.07-5-2025 Приложения к образцу задания Том 1\ТестКейс.docx"; // Путь к файлу Word  public Validation()  {  InitializeComponent();  httpClient = new HttpClient();  }  private void mainmenu\_Load(object sender, EventArgs e)  {  }  private string ExtractValueFromJson(string json)  {  try  {  // Находим начало значения  int valueIndex = json.IndexOf("\"value\"");  if (valueIndex == -1) return null;  // Находим двоеточие после "value"  int colonIndex = json.IndexOf(':', valueIndex);  if (colonIndex == -1) return null;  // Находим первую кавычку после двоеточия  int firstQuote = json.IndexOf('"', colonIndex);  if (firstQuote == -1) return null;  // Находим закрывающую кавычку  int secondQuote = json.IndexOf('"', firstQuote + 1);  if (secondQuote == -1) return null;  // Извлекаем значение между кавычками  return json.Substring(firstQuote + 1, secondQuote - firstQuote - 1);  }  catch  {  return null;  }  }  private async void button1\_Click(object sender, EventArgs e)  {  try  {  HttpResponseMessage response = await httpClient.GetAsync("http://prb.sylas.ru/TransferSimulator/fullName");  if (!response.IsSuccessStatusCode)  {  MessageBox.Show($"Ошибка сервера: {response.StatusCode}");  return;  }  string responseBody = await response.Content.ReadAsStringAsync();  string fullName = ExtractValueFromJson(responseBody);  if (string.IsNullOrEmpty(fullName))  {  MessageBox.Show("Не удалось извлечь ФИО из ответа сервера");  return;  }  labelFullName.Text = fullName;  MessageBox.Show($"Получено ФИО: {fullName}", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при получении данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  private void button2\_Click(object sender, EventArgs e)  {  if (string.IsNullOrWhiteSpace(labelFullName.Text))  {  MessageBox.Show("Сначала получите данные ФИО!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);  return;  }  bool isValid = ValidateFullName(labelFullName.Text);  labelResult.Text = isValid ? "Тест пройден" : "Тест не пройден";  SaveTestResultToWord(labelFullName.Text, isValid);  if (!isValid)  {  // Находим все недопустимые символы  var invalidChars = labelFullName.Text  .Where(c => !char.IsLetter(c) && c != ' ' && c != '-' && c != '\'')  .Distinct()  .ToArray();  MessageBox.Show(  $"ФИО содержит недопустимые символы: {string.Join(", ", invalidChars)}\n" +  "Допустимы только буквы, пробелы, дефисы и апострофы.",  "Ошибка валидации",  MessageBoxButtons.OK,  MessageBoxIcon.Error  );  }  else  {  MessageBox.Show("ФИО прошло валидацию", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  }    private bool ValidateFullName(string fullName)  {  // Разрешаем: русские/английские буквы, пробелы, дефисы и апострофы  // Запрещаем все остальные символы  var regex = new Regex(@"^[a-zA-Zа-яА-ЯёЁ\s\-']+$");  return regex.IsMatch(fullName);  }  private void SaveTestResultToWord(string fullName, bool isValid)  {  try  {  using (WordprocessingDocument doc = File.Exists(TestCasePath)  ? WordprocessingDocument.Open(TestCasePath, true)  : WordprocessingDocument.Create(TestCasePath, WordprocessingDocumentType.Document))  {  MainDocumentPart mainPart = doc.MainDocumentPart;  if (mainPart.Document == null)  {  mainPart.Document = new Document(new Body());  }  Body body = mainPart.Document.Body;  WordTable table = body.Elements<WordTable>().FirstOrDefault();  if (table == null)  {  // Если таблицы нет, создаем новую  table = new WordTable();  TableRow headerRow = new TableRow();  headerRow.Append(  CreateCell("Действие", true),  CreateCell("Ожидаемый результат", true),  CreateCell("Результат", true)  );  table.Append(headerRow);  body.AppendChild(table);  }  // Добавляем новую строку с результатами  TableRow dataRow = new TableRow();  dataRow.Append(  CreateCell("Проверка ФИО на запрещенные символы"),  CreateCell(fullName),  CreateCell(isValid ? "Успешно" : "Не успешно")  );  table.Append(dataRow);  mainPart.Document.Save();  }  MessageBox.Show("Результат сохранён в Word!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);  }  catch (Exception ex)  {  MessageBox.Show($"Ошибка при сохранении в Word: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  }  private TableCell CreateCell(string text, bool isHeader = false)  {  TableCell cell = new TableCell(new Paragraph(new Run(new Text(text))));  if (isHeader)  {  RunProperties runProps = new RunProperties(new Bold());  cell.Descendants<Run>().First().RunProperties = runProps;  }  return cell;  }  }  } |
| --- | --- |

В методе `ValidateFullName` используется регулярное выражение для проверки строки `fullName` на соответствие определённым критериям. Давайте разберём это регулярное выражение:

- `^` и `$`: Эти символы обозначают начало и конец строки соответственно. Они используются, чтобы убедиться, что вся строка соответствует шаблону, а не только её часть.

- `[a-zA-Zа-яА-ЯёЁ\s\-']`:

- `a-zA-Z`: разрешает любые латинские буквы в нижнем и верхнем регистре.

- `а-яА-ЯёЁ`: разрешает любые русские буквы в нижнем и верхнем регистре, включая буквы "ё" и "Ё".

- `\s`: разрешает пробелы.

- `\-`: разрешает дефисы. Дефис экранирован обратным слешем, чтобы показать, что это символ дефиса, а не часть синтаксиса регулярного выражения.

- `'`: разрешает апострофы.

- `+`: указывает, что предыдущий элемент (в данном случае символы в квадратных скобках) может повторяться один или более раз.

Если вы хотите убрать разрешение на использование апострофа, просто уберите его из регулярного выражения. Вот как будет выглядеть изменённый метод:

| 1  2  3  4  5  6  7  8  9  10 | private bool ValidateFullName(string fullName)  {  // Разрешаем: русские/английские буквы, пробелы и дефисы  // Запрещаем все остальные символы, включая апострофы  var regex = new Regex(@"^[a-zA-Zа-яА-ЯёЁ\s\-]+$");  return regex.IsMatch(fullName);  } |
| --- | --- |

Теперь регулярное выражение не разрешает использование апострофов в строке `fullName`.

Да, если вы изменили метод ValidateFullName, чтобы запретиь символ ‘, вам также нужно обновить часть кода, которая определяет недопустимые символы. В текущей версии кода недопустимые символы определяются следующим образом:

| 1 | .Where(c => !char.IsLetter(c) && c != ' ' && c != '-' && c != '\'') |
| --- | --- |

Чтобы запретить символ ‘, нужно убрать его из списка допустимых символов. Вот обновлённый код:

| 1  2  3  4  5  6  7  8  9  10  11  12  13 | bool isValid = ValidateFullName(label1.Text);  label2.Text = isValid ? "Тест пройден" : "Тест не пройден";  SaveTestResultToWord(label1.Text, isValid);  if (!isValid)  {  // Находим все недопустимые символы  var invalidChars = label1.Text  .Where(c => !char.IsLetter(c) && c != ' ' && c != '-') |
| --- | --- |

УТВЕРЖДЕН

ИС-01 02 03-ГО

СПЕЦИАЛЬНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Информационная система “Гостиница”

Описание программы

…-01 13 01

Страниц 13

Литера

| Инв. № подл. | Подп. и дата | Взам. Инв. № | Инв. № дубл. | Подп. и дата |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

ИС-01 02 03

Разработал: ФИО

ИС-01 02 03

**АННОТАЦИЯ**

Документ содержит принципы построения специального программного обеспечения для информационной системы “Гостиница” В документе приведены следующие сведения:

- функциональное назначение программы;

- описание логической структуры программы;

- вызов и загрузка программы;

ПО разработано на языке C# .

В качестве среды разработки использована среда MS VS.

ПО обеспечивает пользовательский интерфейс для управления техническими средствами для управления гостиницей, отображения состояния номеров, обмен данными с сервером баз данных.

ИС-01 02 03

**1. ОБЩИЕ СВЕДЕНИЯ**

1.1. Обозначение и наименование программы

Наименование программы – Информационная система “Гостиница”

Обозначение программы – ИС-01 02 03

**2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ**

ПО выполняет следующие функции:

1. Управление бронированием номеров;
2. Регистрация и выписка гостей;
3. Отображение состояния номеров на визуальных формах;
4. Ввод и редактирование данных о гостях;
5. Ввод и редактирование данных о номерах;
6. Идентификация и аутентификация пользователей путем авторизации;
7. Формирование отчетов о загрузке номерного фонда.

**3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ**

3.2.1. Основные классы

Необходимые для программы классы и их исходные файлы представлены в таблице 1.

ИС-01 02 03

Таблица 1

| Наименование класса | Файл объявления(\*.h) | Файл описания(\*.cpp) |
| --- | --- | --- |
| QAbout | about | about |
| ArhivSeans | arh\_seans | arh\_seans |
| BaseClassDlg | baseclass.h | baseclass.cpp |
| cBase\_interf | Base\_interf.h | base\_interf.cpp |
| BaseEditDlg | baseeditdlg.cpp | baseeditdlg.cpp |

3.2.2. Описание классов

Для запуска приложения используется функция main, которая принимает аргументы приложения. В качестве первого аргумента передается IP-адрес сервера, на котором находится база данных ПРЦ (далее БД). Все остальные настройки программы берутся из базы данных.

Функция main прежде всего выполняет соединение с базой данных MySQL. Если соединение с базой данных по каким-то причинам не произошло (не тот IP-адрес, либо база не установлена, либо не работает СУБД MySQL), функция main выдает сообщение об этом и завершает свою работу.

Иначе, функция далее выполняет следующие действия:

* создает объекты данных для дальнейшего управления;
* запускает главное окно приложения ArmNdrNa.

Далее приводится краткое описание основных классов программы.

Класс SqlDatabase обеспечивает соединение с базой данных PSQL и предоставляет функции для работы с БД.

Класс ManageApp на основе содержимого БД создает список объектов данных, которые являются основными элементами программы и модели данных,

ИС-01 02 03

включающие в себя разные наборы объектов данных.

Класс MainWindow обеспечивает работу главного окна приложения, которое включает в себя обработку команд меню, визуальное отображение состояния технических средств комплекса, вызов диалоговых форм управления техническими средствами.

Класс ObjectData имеет несколько конструкторов и может создавать объекты с типом, номером и данными, структурированными в XML-формат.

Класс ObjectManage является потомком ObjectData и является основным классом программы, так как выполняет важнейшие функции:

* содержит статический указатель на класс SqlDatabase, который используется во всех файлах программы;
* инициализирует объект, загружая его данные из БД (функция loadObjFromDB);
* при необходимости создает диалоговую форму для объекта данных, используя указатель на класс BaseClassDlg;
* обновляет данные объекта при их изменении в процессе работы, в том числе и на диалоговой форме.

Класс BaseClassDlg создает базовую диалоговую форму для управления ТС, в которой определяется основной стиль внешнего вида и поведения окна при его вызове и в процессе выполнения команд.

Классы QAK, QBFM, QDUKK, QPK, QMRPU, QUFKS, QUFSOCH, QSEV, QMRPU являются потомками BaseClassDlg. Они создают диалоговые формы управления техническими средствами ПРЦ.

Класс DlgMainWin – базовый класс для создания диалоговых форм с наполнением в зависимости от заданного делегата. В данном случае реализовано отображение графики и отображение моделей данных в виде элементов – потомков класса QWidget.

Класс BaseClassKmd обеспечивает процесс отправки и выполнения

ИС-01 02 03

команд, отображения результатов выполнения на соответствующих диалогах управления средствами.

Класс BaseThread – базовый класс создания потоков в программе, а класс DemonThreads осуществляет выполнение потоков из очереди потоков.

Класс Magazine – шаблонный класс очереди, который обеспечивает добавление и выборку элементов очереди.

Класс Socket – класс на основе QUdpSocket осуществляет прием и передачу данных через заданный порт.

По командам меню Данные вызываются классы: QEditAdrLIS, QEditLM, QEditOper, QEditPRA, QEditRDM, QEditRPDU, QEditWP, которые отвечают за редактирование определенных таблиц базы данных.

**6. ВЫЗОВ И ЗАГРУЗКА**

Запуск программы осуществляется с помощью MS VS “Запуск без отладки”.

После запуска приложения появится диалоговое окно аутентификации пользователя. После ввода в данном окне логина и пароля выполняется процедура аутентификации пользователя и, в случае успешного ее выполнения, загружается главное окно программы в соответствии с ролью пользователя.