ПРИЛОЖЕНИЕ Б. ТЕКСТ ПРОГРАММЫ

АНОТАЦИЯ

В данном программном документе приведено текст программы для системы управления гостиницей «Star Maze Hotel».

В данном программном документе, в разделе «Текст программы» указана информация о приложении, области применения программы, модулей программы и код каждого программного модуля.

ОГЛАВЛЕНИЕ

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc125899161)

# ТЕКСТ ПРОГРАММЫ

* 1. Наименование программы

Наименование - информационная система управления гостиницей «Star Maze Hotel».

* 1. Область применения программы

Приложение предназначено к применению пользователям и сотрудникам для бронирования, учета финансов, контроля за персоналом.

* 1. Модули

Таблица 1 – Модули приложения API

| № | Модуль | Описание | Размер (в Кбайтах) | Количество строк кода |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 1 | BookingsController.cs | Данный класс отвечает за заимодействие с данными бронирования через сервер. | 4 | 146 |
| 2 | CheckInsController.cs | Данный класс отвечает за заимодействие с данными заселения через сервер. | 4 | 145 |
| 3 | CheckOutsController.cs | Данный класс отвечает за заимодействие с данными чек-аута через сервер. | 5 | 123 |
| 4 | HotelsController.cs | Данный класс отвечает за заимодействие с данными гостиниц через сервер. | 4 | 123 |
| 5 | RolesController.cs | Данный класс отвечает за заимодействие с данными роли через сервер. | 4 | 123 |
| 6 | RoomsController.cs | Данный класс отвечает за заимодействие с данными комнат через сервер. | 4 | 123 |
| 7 | ServicesController.cs | Данный класс отвечает за заимодействие с данными услуг через сервер. | 4 | 123 |
| 8 | UsersController.cs | Данный класс отвечает за заимодействие с данными пользователя через сервер. | 12 | 339 |
| 9 | Booking.cs |  | 1 | 19 |
| 10 | CheckIn.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 15 |
| 11 | CheckOut.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 16 |
| 12 | Hotel.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 15 |
| 13 | HotelContext.cs | Данный класс отвечает за модель используемую при создания запроса. | 10 | 286 |
| 14 | Role.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 13 |
| 15 | Room.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 17 |
| 16 | Service.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 14 |
| 17 | Status.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 13 |
| 18 | Token.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 12 |
| 19 | User.cs | Данный класс отвечает за модель используемую при создания запроса. | 1 | 23 |

Таблица 2 – Модули приложения WPF.

| № | Модуль | Описание | Размер (в Кбайтах) | Количество строк кода |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 1 | DataBaseClass.cs | Данный класс отвечает за подключение базы данных | 2 | 42 |
| 2 | Registration.xaml | Данный класс отвечает за отображение элементов и разметку при регитсрации | 9 | 163 |
| 3 | Registration.xaml.cs | Данный класс отвечает за логику функций регистрации | 3 | 78 |
| 4 | Manager.xaml | Данный класс отвечает за отображение элементов и разметку управляющего | 8 | 137 |
| 5 | Manager.xaml.cs | Данный класс отвечает за логику функций управляющий | 18 | 438 |
| 6 | MainWindow.xaml | Данный класс отвечает за отображение элементов и разметку при авторизации | 3 | 56 |
| 7 | MainWindow.xaml.cs | Данный класс отвечает за логику функций авторизации | 6 | 143 |
| 8 | Client.xaml | Данный класс отвечает за отображение элементов и разметку в клиенте | 10 | 178 |
| 9 | Client.xaml.cs | Данный класс отвечает за логику функций клиентов | 22 | 518 |
| 10 | AdminNeBD.xaml | Данный класс отвечает за отображение элементов и разметку в администраторе | 8 | 123 |
| 11 | AdminNeBD.xaml.cs | Данный класс отвечает за логику функций администратора | 14 | 351 |
| 12 | Admin.xaml.cs | Данный класс отвечает за отображение элементов и разметку в администраторе БД | 18 | 259 |
| 13 | Admin.xaml.cs | Данный класс отвечает за логику функций администратора БД | 29 | 704 |

* 1. Код программы

Код приложения API

BookingsController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class BookingsController : ControllerBase

{

private readonly HotelContext \_context;

public BookingsController(HotelContext context)

{

\_context = context;

}

// GET: api/Bookings

[HttpGet]

public async Task<ActionResult<IEnumerable<Booking>>> GetBookings()

{

if (\_context.Bookings == null)

{

return NotFound();

}

return await \_context.Bookings.ToListAsync();

}

// GET: api/Bookings/5

[HttpGet("{id}")]

public async Task<ActionResult<Booking>> GetBooking(int id)

{

if (\_context.Bookings == null)

{

return NotFound();

}

var booking = await \_context.Bookings.FindAsync(id);

if (booking == null)

{

return NotFound();

}

return booking;

}

// PUT: api/Bookings/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutBooking(int id, Booking booking)

{

if (id != booking.IdBooking)

{

return BadRequest();

}

\_context.Entry(booking).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!BookingExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Bookings

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Booking>> PostBooking(Booking booking)

{

if (\_context.Bookings == null)

{

return Problem("Entity set 'HotelContext.Bookings' is null.");

}

\_context.Bookings.Add(booking);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetBooking", new { id = booking.IdBooking }, booking);

}

// GET: api/Bookings/{userId}

[HttpGet("/get/{userId}")]

public async Task<ActionResult<IEnumerable<Booking>>> GetBookingsByUserId(int userId)

{

var bookings = await \_context.Bookings

.Where(b => b.UserId == userId)

.ToListAsync();

return bookings; // Вернуть пустой список, если нет значений

}

// DELETE: api/Bookings/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteBooking(int id)

{

if (\_context.Bookings == null)

{

return NotFound();

}

var booking = await \_context.Bookings.FindAsync(id);

if (booking == null)

{

return NotFound();

}

\_context.Bookings.Remove(booking);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool BookingExists(int id)

{

return (\_context.Bookings?.Any(e => e.IdBooking == id)).GetValueOrDefault();

}

}

}

CheckInsController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class CheckInsController : ControllerBase

{

private readonly HotelContext \_context;

public CheckInsController(HotelContext context)

{

\_context = context;

}

// GET: api/CheckIns

[HttpGet]

public async Task<ActionResult<IEnumerable<CheckIn>>> GetCheckIns()

{

if (\_context.CheckIns == null)

{

return NotFound();

}

return await \_context.CheckIns.ToListAsync();

}

// GET: api/CheckIns/5

[HttpGet("{id}")]

public async Task<ActionResult<CheckIn>> GetCheckIn(int id)

{

if (\_context.CheckIns == null)

{

return NotFound();

}

var checkIn = await \_context.CheckIns.FindAsync(id);

if (checkIn == null)

{

return NotFound();

}

return checkIn;

}

// PUT: api/CheckIns/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutCheckIn(int id, CheckIn checkIn)

{

if (id != checkIn.IdCheckIn)

{

return BadRequest();

}

\_context.Entry(checkIn).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CheckInExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/CheckIns

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<CheckIn>> PostCheckIn(CheckIn checkIn)

{

if (\_context.CheckIns == null)

{

return Problem("Entity set 'HotelContext.CheckIns' is null.");

}

\_context.CheckIns.Add(checkIn);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetCheckIn", new { id = checkIn.IdCheckIn }, checkIn);

}

// DELETE: api/CheckIns/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteCheckIn(int id)

{

if (\_context.CheckIns == null)

{

return NotFound();

}

var checkIn = await \_context.CheckIns.FindAsync(id);

if (checkIn == null)

{

return NotFound();

}

\_context.CheckIns.Remove(checkIn);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool CheckInExists(int id)

{

return (\_context.CheckIns?.Any(e => e.IdCheckIn == id)).GetValueOrDefault();

}

}

}

CheckOutsController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class CheckOutsController : ControllerBase

{

private readonly HotelContext \_context;

public CheckOutsController(HotelContext context)

{

\_context = context;

}

// GET: api/CheckOuts

[HttpGet]

public async Task<ActionResult<IEnumerable<CheckOut>>> GetCheckOuts()

{

if (\_context.CheckOuts == null)

{

return NotFound();

}

return await \_context.CheckOuts.ToListAsync();

}

// GET: api/CheckOuts/5

[HttpGet("{id}")]

public async Task<ActionResult<CheckOut>> GetCheckOut(int id)

{

if (\_context.CheckOuts == null)

{

return NotFound();

}

var checkOut = await \_context.CheckOuts.FindAsync(id);

if (checkOut == null)

{

return NotFound();

}

return checkOut;

}

// PUT: api/CheckOuts/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutCheckOut(int id, CheckOut checkOut)

{

if (id != checkOut.IdCheckOut)

{

return BadRequest();

}

\_context.Entry(checkOut).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CheckOutExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/CheckOuts

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<CheckOut>> PostCheckOut(CheckOut checkOut)

{

if (\_context.CheckOuts == null)

{

return Problem("Entity set 'HotelContext.CheckOuts' is null.");

}

\_context.CheckOuts.Add(checkOut);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetCheckOut", new { id = checkOut.IdCheckOut }, checkOut);

}

// DELETE: api/CheckOuts/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteCheckOut(int id)

{

if (\_context.CheckOuts == null)

{

return NotFound();

}

var checkOut = await \_context.CheckOuts.FindAsync(id);

if (checkOut == null)

{

return NotFound();

}

\_context.CheckOuts.Remove(checkOut);

await \_context.SaveChangesAsync();

return NoContent();

}

// GET: api/CheckOuts/ByUser/5

[HttpGet("ByUser/{userId}")]

public async Task<ActionResult<IEnumerable<CheckOut>>> GetCheckOutsByUser(int userId)

{

if (\_context.CheckOuts == null)

{

return NotFound();

}

var checkOuts = await \_context.CheckOuts

.Where(co => co.UserId == userId)

.ToListAsync();

if (checkOuts == null || checkOuts.Count == 0)

{

return NotFound();

}

return checkOuts;

}

private bool CheckOutExists(int id)

{

return (\_context.CheckOuts?.Any(e => e.IdCheckOut == id)).GetValueOrDefault();

}

}

}

HotelsController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class HotelsController : ControllerBase

{

private readonly HotelContext \_context;

public HotelsController(HotelContext context)

{

\_context = context;

}

// GET: api/Hotels

[HttpGet]

public async Task<ActionResult<IEnumerable<Hotel>>> GetHotels()

{

if (\_context.Hotels == null)

{

return NotFound();

}

return await \_context.Hotels.ToListAsync();

}

// GET: api/Hotels/5

[HttpGet("{id}")]

public async Task<ActionResult<Hotel>> GetHotel(int id)

{

if (\_context.Hotels == null)

{

return NotFound();

}

var hotel = await \_context.Hotels.FindAsync(id);

if (hotel == null)

{

return NotFound();

}

return hotel;

}

// PUT: api/Hotels/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutHotel(int id, Hotel hotel)

{

if (id != hotel.IdHotels)

{

return BadRequest();

}

\_context.Entry(hotel).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!HotelExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Hotels

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Hotel>> PostHotel(Hotel hotel)

{

if (\_context.Hotels == null)

{

return Problem("Entity set 'HotelContext.Hotels' is null.");

}

\_context.Hotels.Add(hotel);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetHotel", new { id = hotel.IdHotels }, hotel);

}

// DELETE: api/Hotels/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteHotel(int id)

{

if (\_context.Hotels == null)

{

return NotFound();

}

var hotel = await \_context.Hotels.FindAsync(id);

if (hotel == null)

{

return NotFound();

}

\_context.Hotels.Remove(hotel);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool HotelExists(int id)

{

return (\_context.Hotels?.Any(e => e.IdHotels == id)).GetValueOrDefault();

}

}

}

RolesController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class RolesController : ControllerBase

{

private readonly HotelContext \_context;

public RolesController(HotelContext context)

{

\_context = context;

}

// GET: api/Roles

[HttpGet]

public async Task<ActionResult<IEnumerable<Role>>> GetRoles()

{

if (\_context.Roles == null)

{

return NotFound();

}

return await \_context.Roles.ToListAsync();

}

// GET: api/Roles/5

[HttpGet("{id}")]

public async Task<ActionResult<Role>> GetRole(int id)

{

if (\_context.Roles == null)

{

return NotFound();

}

var role = await \_context.Roles.FindAsync(id);

if (role == null)

{

return NotFound();

}

return role;

}

// PUT: api/Roles/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutRole(int id, Role role)

{

if (id != role.IdRole)

{

return BadRequest();

}

\_context.Entry(role).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!RoleExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Roles

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Role>> PostRole(Role role)

{

if (\_context.Roles == null)

{

return Problem("Entity set 'HotelContext.Roles' is null.");

}

\_context.Roles.Add(role);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetRole", new { id = role.IdRole }, role);

}

// DELETE: api/Roles/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteRole(int id)

{

if (\_context.Roles == null)

{

return NotFound();

}

var role = await \_context.Roles.FindAsync(id);

if (role == null)

{

return NotFound();

}

\_context.Roles.Remove(role);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool RoleExists(int id)

{

return (\_context.Roles?.Any(e => e.IdRole == id)).GetValueOrDefault();

}

}

}

RoomsController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class RoomsController : ControllerBase

{

private readonly HotelContext \_context;

public RoomsController(HotelContext context)

{

\_context = context;

}

// GET: api/Rooms

[HttpGet]

public async Task<ActionResult<IEnumerable<Room>>> GetRooms()

{

if (\_context.Rooms == null)

{

return NotFound();

}

return await \_context.Rooms.ToListAsync();

}

// GET: api/Rooms/5

[HttpGet("{id}")]

public async Task<ActionResult<Room>> GetRoom(int id)

{

if (\_context.Rooms == null)

{

return NotFound();

}

var room = await \_context.Rooms.FindAsync(id);

if (room == null)

{

return NotFound();

}

return room;

}

// PUT: api/Rooms/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutRoom(int id, Room room)

{

if (id != room.IdRoom)

{

return BadRequest();

}

\_context.Entry(room).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!RoomExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Rooms

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Room>> PostRoom(Room room)

{

if (\_context.Rooms == null)

{

return Problem("Entity set 'HotelContext.Rooms' is null.");

}

\_context.Rooms.Add(room);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetRoom", new { id = room.IdRoom }, room);

}

// DELETE: api/Rooms/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteRoom(int id)

{

if (\_context.Rooms == null)

{

return NotFound();

}

var room = await \_context.Rooms.FindAsync(id);

if (room == null)

{

return NotFound();

}

\_context.Rooms.Remove(room);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool RoomExists(int id)

{

return (\_context.Rooms?.Any(e => e.IdRoom == id)).GetValueOrDefault();

}

}

}

ServicesController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ServicesController : ControllerBase

{

private readonly HotelContext \_context;

public ServicesController(HotelContext context)

{

\_context = context;

}

// GET: api/Services

[HttpGet]

public async Task<ActionResult<IEnumerable<Service>>> GetServices()

{

if (\_context.Services == null)

{

return NotFound();

}

return await \_context.Services.ToListAsync();

}

// GET: api/Services/5

[HttpGet("{id}")]

public async Task<ActionResult<Service>> GetService(int id)

{

if (\_context.Services == null)

{

return NotFound();

}

var service = await \_context.Services.FindAsync(id);

if (service == null)

{

return NotFound();

}

return service;

}

// PUT: api/Services/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutService(int id, Service service)

{

if (id != service.IdService)

{

return BadRequest();

}

\_context.Entry(service).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ServiceExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Services

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Service>> PostService(Service service)

{

if (\_context.Services == null)

{

return Problem("Entity set 'HotelContext.Services' is null.");

}

\_context.Services.Add(service);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetService", new { id = service.IdService }, service);

}

// DELETE: api/Services/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteService(int id)

{

if (\_context.Services == null)

{

return NotFound();

}

var service = await \_context.Services.FindAsync(id);

if (service == null)

{

return NotFound();

}

\_context.Services.Remove(service);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool ServiceExists(int id)

{

return (\_context.Services?.Any(e => e.IdService == id)).GetValueOrDefault();

}

}

}

StatusController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class StatusController : ControllerBase

{

private readonly HotelContext \_context;

public StatusController(HotelContext context)

{

\_context = context;

}

// GET: api/Status

[HttpGet]

public async Task<ActionResult<IEnumerable<Status>>> GetStatuses()

{

if (\_context.Statuses == null)

{

return NotFound();

}

return await \_context.Statuses.ToListAsync();

}

// GET: api/Status/5

[HttpGet("{id}")]

public async Task<ActionResult<Status>> GetStatus(int id)

{

if (\_context.Statuses == null)

{

return NotFound();

}

var status = await \_context.Statuses.FindAsync(id);

if (status == null)

{

return NotFound();

}

return status;

}

// PUT: api/Status/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutStatus(int id, Status status)

{

if (id != status.IdStatus)

{

return BadRequest();

}

\_context.Entry(status).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!StatusExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Status

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Status>> PostStatus(Status status)

{

if (\_context.Statuses == null)

{

return Problem("Entity set 'HotelContext.Statuses' is null.");

}

\_context.Statuses.Add(status);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetStatus", new { id = status.IdStatus }, status);

}

// DELETE: api/Status/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteStatus(int id)

{

if (\_context.Statuses == null)

{

return NotFound();

}

var status = await \_context.Statuses.FindAsync(id);

if (status == null)

{

return NotFound();

}

\_context.Statuses.Remove(status);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool StatusExists(int id)

{

return (\_context.Statuses?.Any(e => e.IdStatus == id)).GetValueOrDefault();

}

}

}

UsersController.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using ApiHotels.Models;

using System.Security.Cryptography;

using System.Text;

namespace ApiHotels.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UsersController : ControllerBase

{

private readonly HotelContext \_context;

public UsersController(HotelContext context)

{

\_context = context;

}

public static byte[] GenerateSalt(int length)

{

byte[] salt = new byte[length];

using (RandomNumberGenerator rng = RandomNumberGenerator.Create())

{

rng.GetBytes(salt);

}

return salt;

}

// GET: api/Users

[HttpGet]

public async Task<ActionResult<IEnumerable<User>>> GetUsers()

{

if (\_context.Users == null)

{

return NotFound();

}

return await \_context.Users.ToListAsync();

}

// GET: api/Users/5

[HttpGet("{id}")]

public async Task<ActionResult<User>> GetUser(int id)

{

if (\_context.Users == null)

{

return NotFound();

}

var user = await \_context.Users.FindAsync(id);

if (user == null)

{

return NotFound();

}

return user;

}

// PUT: api/Users/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutUser(int id, User user)

{

if (id != user.IdUser)

{

return BadRequest();

}

\_context.Entry(user).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UserExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Users

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<User>> PostUser(User user)

{

byte[] Salt = GenerateSalt(20);

user.Salt = Convert.ToBase64String(Salt);

byte[] passwordBytes = Encoding.UTF8.GetBytes(user.PasswordUser);

byte[] hashedBytes = new Rfc2898DeriveBytes(passwordBytes, Salt, 10000).GetBytes(32);

user.PasswordUser = Convert.ToBase64String(hashedBytes);

if (\_context.Users == null)

{

return Problem("Entity set 'HotelContext.Users' is null.");

}

\_context.Users.Add(user);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetUser", new { id = user.IdUser }, user);

}

// DELETE: api/Users/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteUser(int id)

{

if (\_context.Users == null)

{

return NotFound();

}

var user = await \_context.Users.FindAsync(id);

if (user == null)

{

return NotFound();

}

\_context.Users.Remove(user);

await \_context.SaveChangesAsync();

return NoContent();

}

// AUTH: api/Users/5

[HttpGet("{LoginUser}/{Password}")]

public async Task<ActionResult<string>> Authorization(string LoginUser, string Password)

{

var Users = await \_context.Users.Where(u => u.LoginUser == LoginUser).ToListAsync();

if (Users.Count == 0)

{

// пользователь не найден

return NotFound();

}

else if (Users.Count > 1)

{

// обнаружено несколько пользователей с таким именем

return BadRequest("Multiple usernames detected");

}

var user = Users[0];

// преобразовываем строку Salt в массив байтов

byte[] saltBytes = Convert.FromBase64String(user.Salt);

// преобразовываем строку Password в массив байтов

byte[] passwordBytes = Encoding.UTF8.GetBytes(Password);

// вычисляем хеш пароля с помощью соли и 10000 итераций

byte[] hashBytes = new Rfc2898DeriveBytes(passwordBytes, saltBytes, 10000).GetBytes(32);

string hashedPassword = Convert.ToBase64String(hashBytes);

if (hashedPassword == user.PasswordUser)

{

// пароль совпадает, генерируем случайный токен и добавляем его в базу данных

string token;

Token existingToken;

do

{

token = Guid.NewGuid().ToString();

existingToken = await \_context.Tokens.FirstOrDefaultAsync(t => t.Token1 == token);

}

while (existingToken != null);

// создаем новую запись Token и сохраняем ее в базу данных

// создаем новую запись Token и сохраняем ее в базу данных

Token tok = new Token();

tok.Token1 = token;

tok.TokenDatetime = DateTime.Now;

\_context.Tokens.Add(tok);

await \_context.SaveChangesAsync();

return token;

}

else

{

// пароль не совпадает

return BadRequest("Неправильный пароль");

}

}

[HttpPut("Password\_Change")]

public async Task<IActionResult> PutUser(int id, string New\_password)

{

var user = await \_context.Users.FindAsync(id);

if (user == null)

{

return NotFound();

}

// хешируем новый пароль

byte[] Salt = GenerateSalt(20);

user.Salt = Convert.ToBase64String(Salt);

byte[] passwordBytes = Encoding.UTF8.GetBytes(New\_password);

byte[] hashBytes = new Rfc2898DeriveBytes(passwordBytes, Salt, 10000).GetBytes(32);

user.PasswordUser = Convert.ToBase64String(hashBytes);

\_context.Entry(user).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UserExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// GET: api/Users

[HttpGet("auth\_key")]

public async Task<ActionResult<string>> GetAuthKey(string LoginUser)

{

var user = await \_context.Users.SingleOrDefaultAsync(u => u.LoginUser == LoginUser);

if (user == null)

{

return NotFound($"Admin with login '{LoginUser}' was not found.");

}

else if (\_context.Users.Count(u => u.LoginUser == LoginUser) > 1)

{

return BadRequest($"Multiple admins with login '{LoginUser}' were found.");

}

string salt = user.Salt;

if (string.IsNullOrEmpty(salt))

{

return BadRequest($"Salt for user with login '{LoginUser}' is missing or empty.");

}

byte[] saltBytes = Encoding.UTF8.GetBytes(salt.Substring(0, Math.Min(salt.Length, 5)));

byte[] reverseSalt = saltBytes.Reverse().ToArray();

string hashedReverse = Convert.ToBase64String(reverseSalt);

return hashedReverse;

}

// GET: api/Users

[HttpGet("authentication")]

public async Task<ActionResult<string>> GetAuthentication(string LoginUser, string AuthKey)

{

// Retrieve the user's password salt from the database

var user = await \_context.Users.FirstOrDefaultAsync(u => u.LoginUser == LoginUser);

if (user == null)

{

return BadRequest("Invalid LoginUser");

}

var salt = user.Salt;

// Compute the AuthKey from the password salt

byte[] saltBytes = Encoding.UTF8.GetBytes(salt.Substring(0, Math.Min(salt.Length, 5)));

byte[] reverseSalt = saltBytes.Reverse().ToArray();

string hashedReverse = Convert.ToBase64String(reverseSalt);

// Check if the computed AuthKey matches the provided AuthKey

if (hashedReverse != AuthKey)

{

return BadRequest("Invalid AuthKey");

}

// Generate a random token and add it to the database

string token;

Token existingToken;

do

{

token = Guid.NewGuid().ToString();

existingToken = await \_context.Tokens.FirstOrDefaultAsync(t => t.Token1 == token);

}

while (existingToken != null);

Token tok = new Token();

tok.Token1 = token;

tok.TokenDatetime = DateTime.Now;

\_context.Tokens.Add(tok);

await \_context.SaveChangesAsync();

return token;

}

[HttpGet("GetRoleIdByLogin")]

public async Task<ActionResult<int>> GetRoleIdByLogin(string login)

{

var employee = await \_context.Users

.Where(e => e.LoginUser == login)

.FirstOrDefaultAsync();

if (employee == null)

{

return NotFound(); // Или другой статус код, который соответствует ситуации

}

return employee.RoleId;

}

[HttpGet("GetIdByLogin")]

public async Task<ActionResult<int>> GetIdByLogin(string login)

{

var employee = await \_context.Users

.Where(e => e.LoginUser == login)

.FirstOrDefaultAsync();

if (employee == null)

{

return NotFound(); // Или другой статус код, который соответствует ситуации

}

return employee.IdUser;

}

private bool UserExists(int id)

{

return (\_context.Users?.Any(e => e.IdUser == id)).GetValueOrDefault();

}

}

}

Booking.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Booking

{

public int IdBooking { get; set; }

public DateTime ArrivalDate { get; set; }

public DateTime DepartureDate { get; set; }

public int UserId { get; set; }

public int ServiceId { get; set; }

public int RoomId { get; set; }

public bool IsBooking { get; set; }

}

}

CheckIn.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class CheckIn

{

public int IdCheckIn { get; set; }

public string StatusCheckIn { get; set; }

public int UserId { get; set; }

public int BookingId { get; set; }

}

}

CheckOut.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class CheckOut

{

public int IdCheckOut { get; set; }

public DateTime PaymentDate { get; set; }

public int TotalCost { get; set; }

public int CheckInId { get; set; }

public int UserId { get; set; }

}

}

Hotel.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Hotel

{

public int IdHotels { get; set; }

public string AdressHotel { get; set; }

public int RateHotel { get; set; }

public string NumberPhoneHotel { get; set; }

public string EmailHotel { get; set; }

}

}

HotelContext.cs:

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

namespace ApiHotels.Models

{

public partial class HotelContext : DbContext

{

public HotelContext()

{

}

public HotelContext(DbContextOptions<HotelContext> options)

: base(options)

{

}

public virtual DbSet<Booking> Bookings { get; set; }

public virtual DbSet<CheckIn> CheckIns { get; set; }

public virtual DbSet<CheckOut> CheckOuts { get; set; }

public virtual DbSet<Hotel> Hotels { get; set; }

public virtual DbSet<Role> Roles { get; set; }

public virtual DbSet<Room> Rooms { get; set; }

public virtual DbSet<Service> Services { get; set; }

public virtual DbSet<Status> Statuses { get; set; }

public virtual DbSet<Token> Tokens { get; set; }

public virtual DbSet<User> Users { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.

optionsBuilder.UseSqlServer("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;Persist Security Info=True;User ID=sa;Password=123");

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Booking>(entity =>

{

entity.HasKey(e => e.IdBooking);

entity.ToTable("Booking");

entity.Property(e => e.IdBooking).HasColumnName("ID\_Booking");

entity.Property(e => e.ArrivalDate)

.HasColumnType("date")

.HasColumnName("Arrival\_date");

entity.Property(e => e.DepartureDate)

.HasColumnType("date")

.HasColumnName("Departure\_date");

entity.Property(e => e.IsBooking).HasColumnName("Is\_Booking");

entity.Property(e => e.RoomId).HasColumnName("Room\_ID");

entity.Property(e => e.ServiceId).HasColumnName("Service\_ID");

entity.Property(e => e.UserId).HasColumnName("User\_ID");

});

modelBuilder.Entity<CheckIn>(entity =>

{

entity.HasKey(e => e.IdCheckIn);

entity.ToTable("Check\_in");

entity.Property(e => e.IdCheckIn).HasColumnName("ID\_Check\_in");

entity.Property(e => e.BookingId).HasColumnName("Booking\_ID");

entity.Property(e => e.StatusCheckIn)

.HasMaxLength(20)

.IsUnicode(false)

.HasColumnName("Status\_Check\_in");

entity.Property(e => e.UserId).HasColumnName("User\_ID");

});

modelBuilder.Entity<CheckOut>(entity =>

{

entity.HasKey(e => e.IdCheckOut);

entity.ToTable("Check\_out");

entity.Property(e => e.IdCheckOut).HasColumnName("ID\_Check\_out");

entity.Property(e => e.CheckInId).HasColumnName("Check\_in\_ID");

entity.Property(e => e.PaymentDate)

.HasColumnType("date")

.HasColumnName("Payment\_date");

entity.Property(e => e.TotalCost).HasColumnName("Total\_cost");

entity.Property(e => e.UserId).HasColumnName("User\_ID");

});

modelBuilder.Entity<Hotel>(entity =>

{

entity.HasKey(e => e.IdHotels);

entity.Property(e => e.IdHotels).HasColumnName("ID\_Hotels");

entity.Property(e => e.AdressHotel)

.IsUnicode(false)

.HasColumnName("Adress\_hotel");

entity.Property(e => e.EmailHotel)

.HasMaxLength(60)

.IsUnicode(false)

.HasColumnName("Email\_hotel");

entity.Property(e => e.NumberPhoneHotel)

.HasMaxLength(20)

.IsUnicode(false)

.HasColumnName("Number\_Phone\_hotel");

entity.Property(e => e.RateHotel).HasColumnName("Rate\_hotel");

});

modelBuilder.Entity<Role>(entity =>

{

entity.HasKey(e => e.IdRole);

entity.ToTable("Role");

entity.HasIndex(e => e.NameRole, "UQ\_Name\_Role")

.IsUnique();

entity.Property(e => e.IdRole).HasColumnName("ID\_Role");

entity.Property(e => e.NameRole)

.HasMaxLength(50)

.IsUnicode(false)

.HasColumnName("Name\_Role");

});

modelBuilder.Entity<Room>(entity =>

{

entity.HasKey(e => e.IdRoom);

entity.ToTable("Room");

entity.Property(e => e.IdRoom).HasColumnName("ID\_Room");

entity.Property(e => e.CountRoom).HasColumnName("Count\_room");

entity.Property(e => e.HotelsId).HasColumnName("Hotels\_ID");

entity.Property(e => e.NumberRoom).HasColumnName("Number\_room");

entity.Property(e => e.StatusId).HasColumnName("Status\_ID");

entity.Property(e => e.TypeRoom)

.HasMaxLength(60)

.IsUnicode(false)

.HasColumnName("Type\_room");

});

modelBuilder.Entity<Service>(entity =>

{

entity.HasKey(e => e.IdService);

entity.ToTable("Service");

entity.Property(e => e.IdService).HasColumnName("ID\_Service");

entity.Property(e => e.DescriptionServices)

.IsUnicode(false)

.HasColumnName("Description\_Services");

entity.Property(e => e.NameServices)

.HasMaxLength(50)

.IsUnicode(false)

.HasColumnName("Name\_Services");

entity.Property(e => e.PriceServices).HasColumnName("Price\_Services");

});

modelBuilder.Entity<Status>(entity =>

{

entity.HasKey(e => e.IdStatus);

entity.ToTable("Status");

entity.HasIndex(e => e.Availability, "UQ\_Name\_Status")

.IsUnique();

entity.Property(e => e.IdStatus).HasColumnName("ID\_Status");

entity.Property(e => e.Availability)

.HasMaxLength(30)

.IsUnicode(false);

});

modelBuilder.Entity<Token>(entity =>

{

entity.Property(e => e.TokenId).HasColumnName("token\_id");

entity.Property(e => e.Token1)

.HasMaxLength(200)

.IsUnicode(false)

.HasColumnName("token");

entity.Property(e => e.TokenDatetime)

.HasColumnName("token\_datetime")

.HasDefaultValueSql("(sysdatetime())");

});

modelBuilder.Entity<User>(entity =>

{

entity.HasKey(e => e.IdUser);

entity.ToTable("User");

entity.Property(e => e.IdUser).HasColumnName("ID\_User");

entity.Property(e => e.EmailUser)

.HasMaxLength(60)

.IsUnicode(false)

.HasColumnName("Email\_User");

entity.Property(e => e.FirstNameUser)

.HasMaxLength(30)

.IsUnicode(false)

.HasColumnName("First\_Name\_User");

entity.Property(e => e.LastNameUser)

.HasMaxLength(30)

.IsUnicode(false)

.HasColumnName("Last\_Name\_User");

entity.Property(e => e.LoginUser)

.HasMaxLength(50)

.IsUnicode(false)

.HasColumnName("Login\_User");

entity.Property(e => e.MiddleNameUser)

.HasMaxLength(30)

.IsUnicode(false)

.HasColumnName("Middle\_name\_User");

entity.Property(e => e.NumberPhone)

.HasMaxLength(30)

.IsUnicode(false)

.HasColumnName("Number\_Phone");

entity.Property(e => e.PasportNumber)

.HasMaxLength(6)

.IsUnicode(false)

.HasColumnName("Pasport\_number");

entity.Property(e => e.PasportSeries)

.HasMaxLength(4)

.IsUnicode(false)

.HasColumnName("Pasport\_series");

entity.Property(e => e.PasswordUser)

.HasMaxLength(100)

.IsUnicode(false)

.HasColumnName("Password\_User");

entity.Property(e => e.RoleId).HasColumnName("Role\_ID");

entity.Property(e => e.Salt)

.HasMaxLength(256)

.IsUnicode(false);

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

Role.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Role

{

public int IdRole { get; set; }

public string NameRole { get; set; }

}

}

Room.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Room

{

public int IdRoom { get; set; }

public int CountRoom { get; set; }

public int Price { get; set; }

public int NumberRoom { get; set; }

public string TypeRoom { get; set; }

public int StatusId { get; set; }

public int HotelsId { get; set; }

}

}

Service.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Service

{

public int IdService { get; set; }

public string NameServices { get; set; }

public string DescriptionServices { get; set; }

public int PriceServices { get; set; }

}

}

Status.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Status

{

public int IdStatus { get; set; }

public string Availability { get; set; }

}

}

Token.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class Token

{

public int TokenId { get; set; }

public string Token1 { get; set; }

public DateTime TokenDatetime { get; set; }

}

}

User.cs:

using System;

using System.Collections.Generic;

namespace ApiHotels.Models

{

public partial class User

{

public int IdUser { get; set; }

public string FirstNameUser { get; set; }

public string LastNameUser { get; set; }

public string MiddleNameUser { get; set; }

public string? EmailUser { get; set; }

public string? NumberPhone { get; set; }

public string? PasportSeries { get; set; }

public string? PasportNumber { get; set; }

public string LoginUser { get; set; }

public string PasswordUser { get; set; }

public string? Salt { get; set; }

public int RoleId { get; set; }

}

}

Код приложения WPF

DataBaseClass.cs:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace WpfHotel

{

public class DataBaseClass

{

public static string Users\_ID = "null", Password = "null", App\_Name = "Администратор";

public static string ConnectionStrig = "Data Source = ALEXANDER-LAPTO\\SQLEXPRESS; Initial Catalog = Hotel; Persist Security Info = true; User ID = sa; Password = 123;";

public SqlConnection connection = new SqlConnection(ConnectionStrig);

private SqlCommand command = new SqlCommand();

public DataTable resultTable = new DataTable();

public SqlDependency dependency = new SqlDependency();

public enum act { select, manipulation };

public void sqlExecute(string quety, act act)

{

command.Connection = connection;

command.CommandText = quety;

command.Notification = null;

switch (act)

{

case act.select:

dependency.AddCommandDependency(command);

SqlDependency.Start(connection.ConnectionString);

connection.Open();

resultTable.Load(command.ExecuteReader());

connection.Close();

break;

case act.manipulation:

connection.Open();

command.ExecuteNonQuery();

connection.Close();

break;

}

}

}

}

Registration.xaml:

<Window x:Class="WpfHotel.Registration"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfHotel"

mc:Ignorable="d"

Title="Registration" Height="450" Width="350">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<TextBox Name="FirstName" Grid.Column="0" Grid.ColumnSpan="2" Text="" Grid.Row="0" HorizontalAlignment="Center" VerticalAlignment="Center" Width="130" Margin="10,20,0,5">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Введите фамилию" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

<TextBox Name="LastName" Grid.Column="0" Grid.ColumnSpan="2" Text="" Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Center" Width="130" Margin="10,10,0,0">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Введите имя" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

<TextBox Name="MiddleName" Grid.Column="0" Grid.ColumnSpan="2" Text="" Grid.Row="2" HorizontalAlignment="Center" VerticalAlignment="Center" Width="130" Margin="10,10,0,0">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Введите отчество" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

<TextBox Name="Login" Grid.Column="0" Grid.ColumnSpan="2" Text="" Grid.Row="3" HorizontalAlignment="Center" VerticalAlignment="Center" Width="130" Margin="10,20,0,5">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Введите логин" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

<TextBox Name="Password" Grid.Column="0" Grid.ColumnSpan="2" Text="" Grid.Row="4" HorizontalAlignment="Center" VerticalAlignment="Center" Width="130" Margin="10,10,0,0">

<TextBox.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</TextBox.Resources>

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Введите пароль" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

<Button Grid.Row="5" Grid.ColumnSpan="2" Content="Зарегистрироваться" HorizontalAlignment="Center" VerticalAlignment="Center" Width="130" Height="20" Foreground="White" Click="Button\_Click\_1" >

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

<Button Grid.Row="6" Grid.ColumnSpan="2" Content="Авторизоваться" HorizontalAlignment="Center" VerticalAlignment="Center" Width="110" Height="20" Foreground="White" Click="Button\_Click" >

<Button.Resources>

<Style TargetType="{x:Type Border}">

<Setter Property="CornerRadius" Value="10"/>

</Style>

</Button.Resources>

</Button>

</Grid>

</Window>

Registration.xaml.cs:

using Newtonsoft.Json;

using System;

using System.Net.Http;

using System.Net;

using System.Text;

using System.Windows;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для Registration.xaml

/// </summary>

public partial class Registration : Window

{

public Registration()

{

InitializeComponent();

}

/// <summary>

/// Обработчик события нажатия кнопки "Назад".

/// </summary>

/// <param name="sender">Объект, вызвавший событие (кнопка).</param>

/// <param name="e">Информация о событии (RoutedEventArgs).</param>

private void Button\_Click(object sender, RoutedEventArgs e)

{

// Создаем экземпляр главного окна и отображаем его

MainWindow main = new MainWindow();

main.Show();

// Закрываем текущее окно регистрации

Close();

}

/// <summary>

/// Обработчик события нажатия кнопки "Зарегистрироваться".

/// </summary>

/// <param name="sender">Объект, вызвавший событие (кнопка).</param>

/// <param name="e">Информация о событии (RoutedEventArgs).</param>

private async void Button\_Click\_1(object sender, RoutedEventArgs e)

{

// Создаем экземпляр HttpClient

HttpClient client = new HttpClient();

// Задаем адрес вашего API

string apiUrl = "https://192.168.43.59:7071/api/Users";

// Создаем объект, который будет отправлен в теле запроса

var userData = new

{

firstNameUser = FirstName.Text,

lastNameUser = LastName.Text,

middleNameUser = MiddleName.Text,

loginUser = Login.Text,

passwordUser = Password.Text,

roleId = 2, // Замените на реальный идентификатор роли

salt = "string"

};

// Сериализуем объект в формат JSON

var userJson = JsonConvert.SerializeObject(userData);

// Настройки безопасности для обеспечения поддержки TLS

ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls;

ServicePointManager.ServerCertificateValidationCallback += (senders, cert, chain, sslPolicyErrors) => true;

// Отправляем POST-запрос к API

var response = await client.PostAsync(apiUrl, new StringContent(userJson, Encoding.UTF8, "application/json"));

// Проверяем успешность запроса

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Пользователь зарегистрирован успешно!");

}

else

{

MessageBox.Show($"Ошибка при регистрации пользователя. Код ошибки: {response.StatusCode}");

}

}

}

}

Manager.xaml:

<Window x:Class="WpfHotel.Manager"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfHotel"

mc:Ignorable="d"

Title="Manager" Height="450" Width="800">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="10">

<!-- Задайте здесь радиус закругления -->

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Button Height="30" Width="60" VerticalAlignment="Top" Content="Выход" HorizontalAlignment="Right" Click="Button\_Click\_3" />

<TabControl Margin="0,20,0,0">

<TabItem Loaded="TabItem\_Loaded" Header="Бронирование">

<Grid >

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgBooking" Grid.ColumnSpan="7" Margin="20"/>

<Button Grid.Column="2" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click" />

<Button Grid.Column="3" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_9" />

<Button Grid.Column="4" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_18" />

<Button Height="40" Width="80" Grid.Column="2" Grid.Row="2" Content="Экспорт в CSV" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ExportToCsv\_Click" Background="White"/>

<Button Height="40" Width="80" Grid.Column="4" Grid.Row="2" Content="Импорт из CSV" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ImportFromCsv\_Click" Background="White"/>

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_1" Header="Заселение">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgCheckIn" Grid.ColumnSpan="3" Margin="20" />

<DataGrid Name="dgCheckOut" Grid.ColumnSpan="4" Grid.Column="3" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_1" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_10" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_19" />

<Button Grid.Column="3" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_2" />

<Button Grid.Column="4" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_11" />

<Button Grid.Column="5" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_20" />

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_4" Header="Пользователи">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgUser" Grid.ColumnSpan="3" Margin="20"/>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Window>

Manager.xaml.cs:

using Microsoft.Win32;

using Newtonsoft.Json;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Net.Http;

using System.Net.NetworkInformation;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WpfHotel.Models;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для Manager.xaml

/// </summary>

public partial class Manager : Window

{

public Manager()

{

InitializeComponent();

}

public int count;

List<Service> data6;

public async Task LoadData(int count)

{

HttpClient client = new HttpClient();

switch (count)

{

case 1:

try

{

dgBooking.ItemsSource = null;

HttpResponseMessage response = await client.GetAsync($"https://192.168.74.212:7071/api/Bookings");

string json1 = await response.Content.ReadAsStringAsync();

List<Booking> data = JsonConvert.DeserializeObject<List<Booking>>(json1);

dgBooking.ItemsSource = data;

var columns = dgBooking.Columns;

columns[1].Header = "Дата прибытия";

columns[2].Header = "Дата отбытия";

columns[3].Header = "Идентификатор пользователя";

columns[4].Header = "Идентификатор услуги";

columns[5].Header = "Идентификатор номера";

columns[6].Header = "Бронирование";

columns[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(1); }

break;

case 2:

try

{

dgCheckIn.ItemsSource = null;

dgCheckOut.ItemsSource = null;

HttpResponseMessage response2 = await client.GetAsync($"https://192.168.74.212:7071/api/CheckIns");

string json2 = await response2.Content.ReadAsStringAsync();

List<CheckIn> data2 = JsonConvert.DeserializeObject<List<CheckIn>>(json2);

dgCheckIn.ItemsSource = data2;

var columns2 = dgCheckIn.Columns;

columns2[1].Header = "Статус заселения";

columns2[2].Header = "Идентификатор пользователя";

columns2[3].Header = "Идентификатор бронирования";

columns2[0].Visibility = Visibility.Collapsed;

HttpResponseMessage response3 = await client.GetAsync($"https://192.168.74.212:7071/api/CheckOuts");

string json3 = await response3.Content.ReadAsStringAsync();

List<CheckOut> data3 = JsonConvert.DeserializeObject<List<CheckOut>>(json3);

dgCheckOut.ItemsSource = data3;

var columns3 = dgCheckOut.Columns;

columns3[0].Visibility = Visibility.Collapsed;

columns3[1].Header = "Дата оплаты";

columns3[2].Header = "Общая стоимость";

columns3[3].Header = "Идентификатор заселения";

columns3[4].Header = "Идентификатор пользователя";

}

catch { await LoadData(2); }

break;

case 5:

try

{

dgUser.ItemsSource = null;

HttpResponseMessage response8 = await client.GetAsync($"https://192.168.74.212:7071/api/Users");

string json8 = await response8.Content.ReadAsStringAsync();

List<User> data8 = JsonConvert.DeserializeObject<List<User>>(json8);

dgUser.ItemsSource = data8;

HttpResponseMessage response9 = await client.GetAsync($"https://192.168.74.212:7071/api/Roles");

string json9 = await response9.Content.ReadAsStringAsync();

List<Role> data9 = JsonConvert.DeserializeObject<List<Role>>(json9);

var columns8 = dgUser.Columns;

columns8[1].Header = "Имя";

columns8[2].Header = "Фамилия";

columns8[3].Header = "Отчество";

columns8[4].Header = "Email";

columns8[5].Header = "Номер телефона";

columns8[6].Header = "Серия паспорта";

columns8[7].Header = "Номер паспорта";

columns8[8].Header = "Логин";

columns8[9].Header = "Пароль";

columns8[10].Header = "Роль";

columns8[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(5); }

break;

}

}

private async void TabItem\_Loaded(object sender, RoutedEventArgs e)

{

count = 1;

await LoadData(count);

}

private async void TabItem\_Loaded\_1(object sender, RoutedEventArgs e)

{

count = 2;

await LoadData(count);

}

private async void TabItem\_Loaded\_2(object sender, RoutedEventArgs e)

{

count = 3;

await LoadData(count);

}

private async void TabItem\_Loaded\_3(object sender, RoutedEventArgs e)

{

count = 4;

await LoadData(count);

}

private async void TabItem\_Loaded\_4(object sender, RoutedEventArgs e)

{

count = 5;

await LoadData(count);

}

public async void CreateTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

properties[0].SetValue(data, null);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PostAsync($"https://192.168.74.212:7071/api/{ClassName}s", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click(object sender, RoutedEventArgs e)

{

CreateTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_1(object sender, RoutedEventArgs e)

{

CreateTable<CheckIn>(dgCheckIn);

await LoadData(2);

}

private async void Button\_Click\_2(object sender, RoutedEventArgs e)

{

CreateTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_7(object sender, RoutedEventArgs e)

{

CreateTable<User>(dgUser);

await LoadData(5);

}

public async void PutTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int ID = (int)properties[0].GetValue(data);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PutAsync($"https://192.168.74.212:7071/api/{ClassName}s/{ID}", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click\_9(object sender, RoutedEventArgs e)

{

PutTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_10(object sender, RoutedEventArgs e)

{

PutTable<CheckIn>(dgCheckIn);

await LoadData(2);

}

private async void Button\_Click\_11(object sender, RoutedEventArgs e)

{

PutTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_16(object sender, RoutedEventArgs e)

{

PutTable<User>(dgUser);

await LoadData(5);

}

public async Task DeleteTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int id = (int)properties[0].GetValue(data);

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.DeleteAsync($"https://192.168.74.212:7071/api/{ClassName}s/{id}");

await LoadData(count);

}

else

{

MessageBox.Show("Выберете одну строку для удаления!");

}

}

private async void Button\_Click\_18(object sender, RoutedEventArgs e)

{

await DeleteTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_19(object sender, RoutedEventArgs e)

{

await DeleteTable<CheckIn>(dgCheckIn);

await LoadData(2);

}

private async void Button\_Click\_20(object sender, RoutedEventArgs e)

{

await DeleteTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_25(object sender, RoutedEventArgs e)

{

await DeleteTable<User>(dgUser);

await LoadData(5);

}

private void ExportToSql\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;User ID=sa;Password=123"))

{

connection.Open();

List<Booking> data = dgBooking.ItemsSource.Cast<Booking>().ToList();

foreach (var item in data)

{

string insertQuery = $"INSERT INTO Booking (Arrival\_date, Departure\_date, User\_ID, Service\_ID, Room\_ID) " +

$"VALUES ('{item.ArrivalDate}', '{item.DepartureDate}', {item.UserId}, {item.ServiceId}, {item.RoomId})";

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

{

command.ExecuteNonQuery();

}

}

}

MessageBox.Show("Export to SQL completed.");

}

private void ExportToCsv\_Click(object sender, RoutedEventArgs e)

{

List<Booking> data = dgBooking.ItemsSource.Cast<Booking>().ToList();

StringBuilder csv = new StringBuilder();

csv.AppendLine("IdBooking,ArrivalDate,DepartureDate,UserId,ServiceId,RoomId");

foreach (var item in data)

{

csv.AppendLine($"{item.IdBooking},{item.ArrivalDate},{item.DepartureDate},{item.UserId},{item.ServiceId},{item.RoomId}");

}

File.WriteAllText("Booking.csv", csv.ToString());

MessageBox.Show("Export to CSV completed.");

}

private void ImportFromSql\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;User ID=sa;Password=123"))

{

connection.Open();

string selectQuery = "SELECT \* FROM Booking";

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

{

List<Booking> importedData = new List<Booking>();

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Booking booking = new Booking

{

IdBooking = Convert.ToInt32(reader["ID\_Booking"]),

ArrivalDate = Convert.ToDateTime(reader["Arrival\_date"]),

DepartureDate = Convert.ToDateTime(reader["Departure\_date"]),

UserId = Convert.ToInt32(reader["User\_ID"]),

ServiceId = Convert.ToInt32(reader["Service\_ID"]),

RoomId = Convert.ToInt32(reader["Room\_ID"])

};

importedData.Add(booking);

}

}

dgBooking.ItemsSource = importedData;

}

}

MessageBox.Show("Import from SQL completed.");

}

private void ImportFromCsv\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "CSV files (\*.csv)|\*.csv|All files (\*.\*)|\*.\*";

if (openFileDialog.ShowDialog() == true)

{

string[] lines = File.ReadAllLines(openFileDialog.FileName);

List<Booking> importedData = new List<Booking>();

foreach (var line in lines.Skip(1)) // Skip header

{

var values = line.Split(',');

if (values.Length == 6)

{

Booking booking = new Booking

{

IdBooking = int.Parse(values[0]),

ArrivalDate = DateTime.Parse(values[1]),

DepartureDate = DateTime.Parse(values[2]),

UserId = int.Parse(values[3]),

ServiceId = int.Parse(values[4]),

RoomId = int.Parse(values[5])

};

importedData.Add(booking);

}

}

dgBooking.ItemsSource = importedData;

MessageBox.Show("Import from CSV completed.");

}

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

}

}

MainWindow.xaml:

<Window x:Class="WpfHotel.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfHotel"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="350">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="10">

<!-- Задайте здесь радиус закругления -->

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Grid.Column="0" Content="Почта" Grid.Row="1" HorizontalAlignment="Center" VerticalAlignment="Bottom" Margin="120,20,0,0"/>

<Label Grid.Column="0" Content="Пароль" Grid.Row="2" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="120,5,0,0"/>

<TextBox Name="Login" Grid.Column="1" Text="client" Grid.Row="1" HorizontalAlignment="Left" VerticalAlignment="Bottom" Width="130" Margin="10,20,0,5"/>

<TextBox Name="Password" Grid.Column="1" Text="string" Grid.Row="2" HorizontalAlignment="Left" VerticalAlignment="Top" Width="130" Margin="10,10,0,0"/>

<Button Grid.Row="3" Grid.ColumnSpan="2" Content="Войти" HorizontalAlignment="Center" VerticalAlignment="Center" Width="100" Height="20" Click="Button\_Click" />

<Button Grid.Row="4" Grid.ColumnSpan="2" Content="Зарегистрироваться" HorizontalAlignment="Center" VerticalAlignment="Top" Width="150" Height="20" Click="Button\_Click\_1" />

</Grid>

</Window>

MainWindow.xaml.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net.Http;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using DocumentFormat.OpenXml.Math;

using System.Numerics;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

public static string roleIdString;

public static string IDUser;

protected override void OnMouseLeftButtonDown(MouseButtonEventArgs e)

{

base.OnMouseLeftButtonDown(e);

this.DragMove();

}

private readonly string \_apiUrl = "https://192.168.43.59:7071/api";

/// <summary>

/// Обработчик события нажатия кнопки "Войти".

/// </summary>

/// <param name="sender">Объект-инициатор события.</param>

/// <param name="e">Аргументы события.</param>

private async void Button\_Click(object sender, RoutedEventArgs e)

{

string login = Login.Text;

string password = Password.Text;

string token = await Authorization(login, password);

if (token != null)

{

HttpClient client = new HttpClient();

string apiUrl = "https://192.168.43.59:7071/api/Users"; // Замените на фактический URL вашего API

string requestUrl = $"{apiUrl}/GetRoleIdByLogin?login={login}";

HttpClient clientID = new HttpClient(); // Замените на фактический URL вашего API

string requestUrlID = $"{apiUrl}/GetIdByLogin?login={login}";

try

{

HttpResponseMessage response = await client.GetAsync(requestUrl);

HttpResponseMessage responseID = await client.GetAsync(requestUrlID);

if (response.IsSuccessStatusCode)

{

roleIdString = await response.Content.ReadAsStringAsync();

IDUser = await responseID.Content.ReadAsStringAsync();

if (int.TryParse(roleIdString, out int roleId))

{

switch (roleId)

{

case 1:

Admin admin = new Admin();

admin.Show();

Close();

break;

case 2:

Client doctor = new Client();

doctor.Show();

Close();

break;

case 3:

AdminNeBD pacient = new AdminNeBD();

pacient.Show();

Close();

break;

case 4:

Manager registrator = new Manager();

registrator.Show();

Close();

break;

}

}

else

{

// Обработка ошибки преобразования строки в число

MessageBox.Show("Невозможно преобразовать RoleId в число.");

}

}

else

{

// Обработка ошибки, например, вывод сообщения пользователю

MessageBox.Show($"Ошибка: {response.StatusCode}");

}

}

catch (Exception ex)

{

// Обработка исключения, например, вывод сообщения об ошибке

MessageBox.Show($"Произошла ошибка: {ex.Message}");

}

}

}

/// <summary>

/// Метод для выполнения авторизации на сервере.

/// </summary>

/// <param name="login">Логин пользователя.</param>

/// <param name="password">Пароль пользователя.</param>

/// <returns>Токен авторизации.</returns>

private async Task<string> Authorization(string login, string password)

{

using (var httpClient = new HttpClient())

{

ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls;

ServicePointManager.ServerCertificateValidationCallback += (sender, cert, chain, sslPolicyErrors) => true;

HttpResponseMessage response = await httpClient.GetAsync($"{\_apiUrl}/Users/{login}/{password}");

if (response.IsSuccessStatusCode)

{

string result = await response.Content.ReadAsStringAsync();

return result;

}

else

{

MessageBox.Show("Неверный логин или пароль.");

return null;

}

}

}

/// <summary>

/// Обработчик события нажатия кнопки "Зарегистрироваться".

/// </summary>

/// <param name="sender">Объект-инициатор события.</param>

/// <param name="e">Аргументы события.</param>

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

Registration registration = new Registration();

registration.Show();

Close();

}

}

}

Client.xaml:

<Window x:Class="WpfHotel.Client"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfHotel"

mc:Ignorable="d"

Title="Client" Height="450" Width="800">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="10">

<!-- Задайте здесь радиус закругления -->

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Button Height="30" Width="60" VerticalAlignment="Top" Content="Выход" HorizontalAlignment="Right" Click="Button\_Click\_1" />

<TabControl Margin="0,20,0,0">

<TabItem Loaded="TabItem\_Loaded" Header="Бронирование">

<Grid >

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="113\*"/>

<ColumnDefinition Width="113\*"/>

<ColumnDefinition Width="113\*"/>

<ColumnDefinition Width="36\*"/>

<ColumnDefinition Width="77\*"/>

<ColumnDefinition Width="113\*"/>

<ColumnDefinition Width="113\*"/>

<ColumnDefinition Width="113\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgBooking" Grid.ColumnSpan="8" Margin="20,20,20,20"/>

<DatePicker Name="datePickerArrival" Grid.Column="2" Grid.Row="1" VerticalAlignment="Center"/>

<DatePicker Name="datePickerDeparture" Grid.Column="2" Grid.Row="1" VerticalAlignment="Center" Grid.ColumnSpan="4" Margin="113,0,113,0"/>

<ComboBox Name="cbRoom" Grid.Row="1" Grid.Column="5" VerticalAlignment="Center" Width="90" Margin="11,0,12,0"/>

<ComboBox Name="cbService" Grid.Row="1" Grid.Column="6" VerticalAlignment="Center" Width="90" Margin="12,0,12,0"/>

<Button Grid.Column="2" Grid.Row="2" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click" />

<Button Grid.Column="5" Grid.Row="2" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_18" />

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_1" Header="Заселение">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgCheckOut" Grid.ColumnSpan="4" Grid.Column="0" Margin="20"/>

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_2" Header="Отель">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgHotel" Grid.ColumnSpan="3" Margin="20"/>

<DataGrid Name="dgRoom" Grid.ColumnSpan="3" Grid.Column="3" Margin="20"/>

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_3" Header="Услуги">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgService" Grid.ColumnSpan="3" Margin="20,28,20,20"/>

<DataGrid Name="dgStatus" Grid.ColumnSpan="3" Grid.Column="3" Margin="20"/>

<ComboBox DisplayMemberPath="NameServices" Name="cmbTypeServices" Grid.Column="0" Grid.Row="0" VerticalAlignment="Top" SelectionChanged="cmbTypeServices\_SelectionChanged" />

<TextBox x:Name="txtSearch" Grid.Column="1" Grid.Row="0" VerticalAlignment="Top" Margin="5" TextChanged="txtSearch\_TextChanged">

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Поиск" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Window>

Client.xaml.cs:

using DocumentFormat.OpenXml.Spreadsheet;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using WpfHotel.Models;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для Client.xaml

/// </summary>

public partial class Client : Window

{

public Client()

{

InitializeComponent();

MilFill();

MilFills();

// SetDataGridColumns();

}

private void MilFill()

{

DataBaseClass dataBaseClass = new DataBaseClass();

dataBaseClass.sqlExecute(string.Format("select [ID\_Room], [Type\_room] from [dbo].[Room] "), DataBaseClass.act.select);

dataBaseClass.dependency.OnChange += Dependency\_OnChange\_Mil;

Action action = () =>

{

cbRoom.ItemsSource = dataBaseClass.resultTable.DefaultView;

cbRoom.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;

cbRoom.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;

};

Dispatcher.Invoke(action);

}

private void Dependency\_OnChange\_Mil(object sender, SqlNotificationEventArgs e)

{

if (e.Info != SqlNotificationInfo.Invalid)

MilFill();

}

private void MilFills()

{

DataBaseClass dataBaseClass = new DataBaseClass();

dataBaseClass.sqlExecute(string.Format("select [ID\_Service], [Name\_Services] from [dbo].[Service] "), DataBaseClass.act.select);

dataBaseClass.dependency.OnChange += Dependency\_OnChange\_Mils;

Action action = () =>

{

cbService.ItemsSource = dataBaseClass.resultTable.DefaultView;

cbService.SelectedValuePath = dataBaseClass.resultTable.Columns[0].ColumnName;

cbService.DisplayMemberPath = dataBaseClass.resultTable.Columns[1].ColumnName;

};

Dispatcher.Invoke(action);

}

private void Dependency\_OnChange\_Mils(object sender, SqlNotificationEventArgs e)

{

if (e.Info != SqlNotificationInfo.Invalid)

MilFills();

}

public int count;

List<Service> data6;

public async Task LoadData(int count)

{

HttpClient client = new HttpClient();

switch (count)

{

case 1:

dgBooking.ItemsSource = null;

string id = MainWindow.IDUser;

HttpResponseMessage response = await client.GetAsync($"https://192.168.74.212:7071/get/{id}");

if (response.IsSuccessStatusCode)

{

string json1 = await response.Content.ReadAsStringAsync();

List<Booking> data = JsonConvert.DeserializeObject<List<Booking>>(json1);

dgBooking.ItemsSource = data;

var columns = dgBooking.Columns;

columns[1].Header = "Дата прибытия";

columns[2].Header = "Дата отбытия";

columns[3].Header = "Идентификатор пользователя";

columns[4].Header = "Идентификатор услуги";

columns[5].Header = "Идентификатор номера";

columns[6].Header = "Бронирование";

columns[0].Visibility = Visibility.Collapsed;

}

else if (response.StatusCode == HttpStatusCode.NotFound)

{

// Обработка случая, когда нет бронирований для указанного User\_ID

dgBooking.ItemsSource = new List<Booking>(); // Установить пустой список

}

break;

case 2:

try

{

dgCheckOut.ItemsSource = null;

// Получение данных по CheckOut для конкретного пользователя

HttpResponseMessage response3 = await client.GetAsync($"https://192.168.74.212:7071/api/CheckOuts/ByUser/{MainWindow.IDUser}");

if (response3.IsSuccessStatusCode)

{

string json3 = await response3.Content.ReadAsStringAsync();

List<CheckOut> data3 = JsonConvert.DeserializeObject<List<CheckOut>>(json3);

dgCheckOut.ItemsSource = data3;

var columns3 = dgCheckOut.Columns;

columns3[0].Visibility = Visibility.Collapsed;

columns3[1].Header = "Дата оплаты";

columns3[2].Header = "Общая стоимость";

columns3[3].Header = "Идентификатор заселения";

columns3[4].Header = "Идентификатор пользователя";

}

else if (response3.StatusCode == HttpStatusCode.NotFound)

{

// Обработка случая, когда нет данных для указанного User\_ID

dgCheckOut.ItemsSource = new List<CheckOut>(); // Установить пустой список

}

}

catch { await LoadData(2); }

break;

case 3:

try

{

dgHotel.ItemsSource = null;

dgRoom.ItemsSource = null;

HttpResponseMessage response4 = await client.GetAsync($"https://192.168.74.212:7071/api/Hotels");

string json4 = await response4.Content.ReadAsStringAsync();

List<Hotel> data4 = JsonConvert.DeserializeObject<List<Hotel>>(json4);

dgHotel.ItemsSource = data4;

HttpResponseMessage response5 = await client.GetAsync($"https://192.168.74.212:7071/api/Rooms");

string json5 = await response5.Content.ReadAsStringAsync();

List<Room> data5 = JsonConvert.DeserializeObject<List<Room>>(json5);

dgRoom.ItemsSource = data5;

var columns4 = dgHotel.Columns;

columns4[1].Header = "Адрес отеля";

columns4[2].Header = "Рейтинг отеля";

columns4[3].Header = "Номер телефона отеля";

columns4[4].Header = "Email отеля";

columns4[0].Visibility = Visibility.Collapsed;

var columns5 = dgRoom.Columns;

columns5[1].Header = "Количество комнат";

columns5[2].Header = "Цена";

columns5[3].Header = "Номер комнаты";

columns5[4].Header = "Тип комнаты";

columns5[5].Header = "Статус комнаты";

columns5[6].Header = "Идентификатор отеля";

columns5[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(3); }

break;

case 4:

try

{

dgService.ItemsSource = null;

dgStatus.ItemsSource = null;

HttpResponseMessage response6 = await client.GetAsync($"https://192.168.74.212:7071/api/Services");

string json6 = await response6.Content.ReadAsStringAsync();

data6 = JsonConvert.DeserializeObject<List<Service>>(json6);

dgService.ItemsSource = data6;

cmbTypeServices.ItemsSource = data6;

HttpResponseMessage response7 = await client.GetAsync($"https://192.168.74.212:7071/api/Status");

string json7 = await response7.Content.ReadAsStringAsync();

List<Status> data7 = JsonConvert.DeserializeObject<List<Status>>(json7);

dgStatus.ItemsSource = data7;

var columns6 = dgService.Columns;

columns6[1].Header = "Название услуги";

columns6[2].Header = "Описание услуги";

columns6[3].Header = "Цена услуги";

columns6[0].Visibility = Visibility.Collapsed;

var columns7 = dgStatus.Columns;

columns7[1].Header = "Доступность";

columns7[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(4); }

break;

}

}

private async void TabItem\_Loaded(object sender, RoutedEventArgs e)

{

count = 1;

await LoadData(count);

}

private async void TabItem\_Loaded\_1(object sender, RoutedEventArgs e)

{

count = 2;

await LoadData(count);

}

private async void TabItem\_Loaded\_2(object sender, RoutedEventArgs e)

{

count = 3;

await LoadData(count);

}

private async void TabItem\_Loaded\_3(object sender, RoutedEventArgs e)

{

count = 4;

await LoadData(count);

}

private async void TabItem\_Loaded\_4(object sender, RoutedEventArgs e)

{

count = 5;

await LoadData(count);

}

public async void CreateTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

properties[0].SetValue(data, null);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PostAsync($"https://192.168.74.212:7071/api/{ClassName}s", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private HttpClient client = new HttpClient();

private async void Button\_Click(object sender, RoutedEventArgs e)

{

/\*

if (dgBooking.SelectedItems.Count > 0)

{\*/

// Получаем выбранный объект Booking из DataGrid

// Booking selectedBooking = dgBooking.SelectedItem as Booking;

/\* if (selectedBooking != null)

{

// Чтение значения свойства IsBooking

bool isBookingValue = selectedBooking.IsBooking;

\*/

// Используйте значение isBookingValue по вашему усмотрению

// MessageBox.Show($"Значение галочки: {isBookingValue}");

// Создаем объект Booking на основе введенных данных в элементы управления

Booking newBooking = new Booking

{

ArrivalDate = datePickerArrival.SelectedDate.GetValueOrDefault(),

DepartureDate = datePickerDeparture.SelectedDate.GetValueOrDefault(),

UserId = Convert.ToInt32(MainWindow.IDUser),

RoomId = (int)cbRoom.SelectedValue,

ServiceId = (int)cbService.SelectedValue,

IsBooking = false // По умолчанию можно установить false или какой вам нужно

};

string json = JsonConvert.SerializeObject(newBooking);

HttpContent content = new StringContent(json, Encoding.UTF8, "application/json");

HttpResponseMessage response = await client.PostAsync("https://192.168.74.212:7071/api/Bookings", content);

if (response.IsSuccessStatusCode)

{

// Обработка успешного добавления бронирования

MessageBox.Show("Бронирование успешно добавлено!");

await LoadData(1);

}

else

{

// Обработка ошибки добавления бронирования

MessageBox.Show("Ошибка при добавлении бронирования. Пожалуйста, проверьте введенные данные.");

}

/\* }\*/

/\* }

else

{

MessageBox.Show("Выберите строку в DataGrid.");

}

\*/

}

private async void Button\_Click\_2(object sender, RoutedEventArgs e)

{

CreateTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_3(object sender, RoutedEventArgs e)

{

CreateTable<Hotel>(dgHotel);

await LoadData(3);

}

private async void Button\_Click\_4(object sender, RoutedEventArgs e)

{

CreateTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_5(object sender, RoutedEventArgs e)

{

CreateTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_6(object sender, RoutedEventArgs e)

{

CreateTable<Status>(dgStatus);

await LoadData(4);

}

public async void PutTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int ID = (int)properties[0].GetValue(data);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PutAsync($"https://192.168.74.212:7071/api/{ClassName}s/{ID}", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click\_9(object sender, RoutedEventArgs e)

{

PutTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_11(object sender, RoutedEventArgs e)

{

PutTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_12(object sender, RoutedEventArgs e)

{

PutTable<Hotel>(dgHotel);

await LoadData(3);

}

private async void Button\_Click\_13(object sender, RoutedEventArgs e)

{

PutTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_14(object sender, RoutedEventArgs e)

{

PutTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_15(object sender, RoutedEventArgs e)

{

PutTable<Status>(dgStatus);

await LoadData(4);

}

public async Task DeleteTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int id = (int)properties[0].GetValue(data);

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.DeleteAsync($"https://192.168.74.212:7071/api/{ClassName}s/{id}");

await LoadData(count);

}

else

{

MessageBox.Show("Выберете одну строку для удаления!");

}

}

private async void Button\_Click\_18(object sender, RoutedEventArgs e)

{

await DeleteTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_20(object sender, RoutedEventArgs e)

{

await DeleteTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_21(object sender, RoutedEventArgs e)

{

await DeleteTable<Hotel>(dgHotel);

await LoadData(3);

}

private async void Button\_Click\_22(object sender, RoutedEventArgs e)

{

await DeleteTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_23(object sender, RoutedEventArgs e)

{

await DeleteTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_24(object sender, RoutedEventArgs e)

{

await DeleteTable<Status>(dgStatus);

await LoadData(4);

}

private void cmbTypeServices\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

// Получаем выбранное значение из комбобокса

string selectedType = (string)(cmbTypeServices.SelectedItem as Service)?.NameServices;

// Фильтруем данные в data7 в зависимости от выбранного значения

List<Service> filteredData = data6.Where(s => s.NameServices == selectedType).ToList();

// Обновляем источник данных для DataGrid

dgService.ItemsSource = filteredData;

}

private void txtSearch\_TextChanged(object sender, TextChangedEventArgs e)

{

// Получаем текст из TextBox

string searchText = txtSearch.Text;

// Фильтруем данные в data7 в зависимости от введенного текста

List<Service> filteredData = data6.Where(s => s.NameServices.Contains(searchText)).ToList();

// Обновляем источник данных для DataGrid

dgService.ItemsSource = filteredData;

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

}

}

App.xaml:

<Application x:Class="WpfHotel.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:WpfHotel"

StartupUri="MainWindow.xaml">

<Application.Resources>

</Application.Resources>

</Application>

App.xaml.cs:

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

}

}

AdminNeBD.xaml:

<Window x:Class="WpfHotel.AdminNeBD"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfHotel"

mc:Ignorable="d"

Title="AdminNeBD" Height="450" Width="800">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="10">

<!-- Задайте здесь радиус закругления -->

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Button Height="30" Width="60" VerticalAlignment="Top" Content="Выход" HorizontalAlignment="Right" Click="Button\_Click" />

<TabControl Margin="0,20,0,0">

<TabItem Loaded="TabItem\_Loaded\_2" Header="Отель">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgRoom" Grid.ColumnSpan="3" Grid.Column="0" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_4" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_13" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_22" />

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_3" Header="Услуги">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgService" Grid.ColumnSpan="3" Margin="20,28,20,20"/>

<DataGrid Name="dgStatus" Grid.ColumnSpan="3" Grid.Column="3" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_5" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_14" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_23" />

<Button Grid.Column="3" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_6" />

<Button Grid.Column="4" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_15" />

<Button Grid.Column="5" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_24" />

<ComboBox DisplayMemberPath="NameServices" Name="cmbTypeServices" Grid.Column="0" Grid.Row="0" VerticalAlignment="Top" SelectionChanged="cmbTypeServices\_SelectionChanged" />

<TextBox x:Name="txtSearch" Grid.Column="1" Grid.Row="0" VerticalAlignment="Top" Margin="5" TextChanged="txtSearch\_TextChanged">

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Поиск" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Window>

AdminNeBD.xaml.cs:

using Microsoft.Win32;

using Newtonsoft.Json;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WpfHotel.Models;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для AdminNeBD.xaml

/// </summary>

public partial class AdminNeBD : Window

{

public AdminNeBD()

{

InitializeComponent();

}

public int count;

List<Service> data6;

public async Task LoadData(int count)

{

HttpClient client = new HttpClient();

switch (count)

{

case 3:

try

{

dgRoom.ItemsSource = null;

HttpResponseMessage response4 = await client.GetAsync($"https://192.168.74.212:7071/api/Hotels");

string json4 = await response4.Content.ReadAsStringAsync();

List<Hotel> data4 = JsonConvert.DeserializeObject<List<Hotel>>(json4);

HttpResponseMessage response5 = await client.GetAsync($"https://192.168.74.212:7071/api/Rooms");

string json5 = await response5.Content.ReadAsStringAsync();

List<Room> data5 = JsonConvert.DeserializeObject<List<Room>>(json5);

dgRoom.ItemsSource = data5;

var columns5 = dgRoom.Columns;

columns5[1].Header = "Количество комнат";

columns5[2].Header = "Цена";

columns5[3].Header = "Номер комнаты";

columns5[4].Header = "Тип комнаты";

columns5[5].Header = "Статус комнаты";

columns5[6].Header = "Идентификатор отеля";

columns5[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(3); }

break;

case 4:

try

{

dgService.ItemsSource = null;

dgStatus.ItemsSource = null;

HttpResponseMessage response6 = await client.GetAsync($"https://192.168.74.212:7071/api/Services");

string json6 = await response6.Content.ReadAsStringAsync();

data6 = JsonConvert.DeserializeObject<List<Service>>(json6);

dgService.ItemsSource = data6;

cmbTypeServices.ItemsSource = data6;

HttpResponseMessage response7 = await client.GetAsync($"https://192.168.74.212:7071/api/Status");

string json7 = await response7.Content.ReadAsStringAsync();

List<Status> data7 = JsonConvert.DeserializeObject<List<Status>>(json7);

dgStatus.ItemsSource = data7;

var columns6 = dgService.Columns;

columns6[1].Header = "Название услуги";

columns6[2].Header = "Описание услуги";

columns6[3].Header = "Цена услуги";

columns6[0].Visibility = Visibility.Collapsed;

var columns7 = dgStatus.Columns;

columns7[1].Header = "Доступность";

columns7[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(4); }

break;

}

}

private async void TabItem\_Loaded(object sender, RoutedEventArgs e)

{

count = 1;

await LoadData(count);

}

private async void TabItem\_Loaded\_1(object sender, RoutedEventArgs e)

{

count = 2;

await LoadData(count);

}

private async void TabItem\_Loaded\_2(object sender, RoutedEventArgs e)

{

count = 3;

await LoadData(count);

}

private async void TabItem\_Loaded\_3(object sender, RoutedEventArgs e)

{

count = 4;

await LoadData(count);

}

private async void TabItem\_Loaded\_4(object sender, RoutedEventArgs e)

{

count = 5;

await LoadData(count);

}

public async void CreateTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

properties[0].SetValue(data, null);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PostAsync($"https://192.168.74.212:7071/api/{ClassName}s", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click\_4(object sender, RoutedEventArgs e)

{

CreateTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_5(object sender, RoutedEventArgs e)

{

CreateTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_6(object sender, RoutedEventArgs e)

{

CreateTable<Status>(dgStatus);

await LoadData(4);

}

public async void PutTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int ID = (int)properties[0].GetValue(data);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PutAsync($"https://192.168.74.212:7071/api/{ClassName}s/{ID}", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click\_13(object sender, RoutedEventArgs e)

{

PutTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_14(object sender, RoutedEventArgs e)

{

PutTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_15(object sender, RoutedEventArgs e)

{

PutTable<Status>(dgStatus);

await LoadData(4);

}

public async Task DeleteTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int id = (int)properties[0].GetValue(data);

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.DeleteAsync($"https://192.168.74.212:7071/api/{ClassName}s/{id}");

await LoadData(count);

}

else

{

MessageBox.Show("Выберете одну строку для удаления!");

}

}

private async void Button\_Click\_22(object sender, RoutedEventArgs e)

{

await DeleteTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_23(object sender, RoutedEventArgs e)

{

await DeleteTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_24(object sender, RoutedEventArgs e)

{

await DeleteTable<Status>(dgStatus);

await LoadData(4);

}

private void Button\_Click\_27(object sender, RoutedEventArgs e)

{

string databaseName = "Hotel"; // Замените на имя вашей базы данных

try

{

// Диалоговое окно выбора файла для сохранения резервной копии

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Backup files (\*.bak)|\*.bak";

saveFileDialog.FileName = $"{databaseName}\_Backup.bak";

if (saveFileDialog.ShowDialog() == true)

{

string backupPath = saveFileDialog.FileName;

using (SqlConnection connection = new SqlConnection("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;User ID=sa;Password=123"))

{

connection.Open();

// Создание резервной копии базы данных

string backupQuery = $"BACKUP DATABASE [{databaseName}] TO DISK = '{backupPath}'";

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

{

command.ExecuteNonQuery();

}

}

MessageBox.Show($"Backup created successfully. Path: {backupPath}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Error creating backup: {ex.Message}");

}

}

private void cmbTypeServices\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

// Получаем выбранное значение из комбобокса

string selectedType = (string)(cmbTypeServices.SelectedItem as Service)?.NameServices;

// Фильтруем данные в data7 в зависимости от выбранного значения

List<Service> filteredData = data6.Where(s => s.NameServices == selectedType).ToList();

// Обновляем источник данных для DataGrid

dgService.ItemsSource = filteredData;

}

private void txtSearch\_TextChanged(object sender, TextChangedEventArgs e)

{

// Получаем текст из TextBox

string searchText = txtSearch.Text;

// Фильтруем данные в data7 в зависимости от введенного текста

List<Service> filteredData = data6.Where(s => s.NameServices.Contains(searchText)).ToList();

// Обновляем источник данных для DataGrid

dgService.ItemsSource = filteredData;

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

}

}

Admin.xaml.cs:

<Window x:Class="WpfHotel.Admin"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:ScottPlot="clr-namespace:ScottPlot;assembly=ScottPlot.WPF"

xmlns:local="clr-namespace:WpfHotel"

mc:Ignorable="d"

Title="Admin" Height="450" Width="800">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}"

BorderBrush="{TemplateBinding BorderBrush}"

BorderThickness="{TemplateBinding BorderThickness}"

CornerRadius="10">

<!-- Задайте здесь радиус закругления -->

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

</Style>

</Window.Resources>

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Button Height="30" Width="60" VerticalAlignment="Top" Content="Выход" HorizontalAlignment="Right" Click="Button\_Click\_29" />

<TabControl Margin="0,20,0,0">

<TabItem Loaded="TabItem\_Loaded" Header="Бронирование">

<Grid >

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgBooking" Grid.ColumnSpan="7" Margin="20" SelectionChanged="dgBooking\_SelectionChanged"/>

<Button Grid.Column="2" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click" />

<Button Grid.Column="3" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_9" />

<Button Grid.Column="4" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_18" />

<Button Height="40" Width="80" Grid.Column="5" Grid.Row="2" VerticalAlignment="Center" Content="Back Up" HorizontalAlignment="Center" Click="Button\_Click\_27" Background="White" />

<Button Height="40" Width="80" Grid.Column="1" Grid.Row="2" Content="Экспорт в SQL" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ExportToSql\_Click" Background="White"/>

<Button Height="40" Width="80" Grid.Column="2" Grid.Row="2" Content="Экспорт в CSV" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ExportToCsv\_Click" Background="White"/>

<Button Height="40" Width="80" Grid.Column="3" Grid.Row="2" Content="Импорт из SQL" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ImportFromSql\_Click" Background="White"/>

<Button Height="40" Width="80" Grid.Column="4" Grid.Row="2" Content="Импорт из CSV" HorizontalAlignment="Center" VerticalAlignment="Center" Click="ImportFromCsv\_Click" Background="White"/>

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_1" Header="Заселение" Height="20" VerticalAlignment="Top">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgCheckIn" Grid.ColumnSpan="3" Margin="20"/>

<DataGrid Loaded="dgCheckOut\_Loaded" Name="dgCheckOut" Grid.ColumnSpan="4" Grid.Column="3" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_1" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_10" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_19" />

<Button Grid.Column="3" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_2" />

<Button Grid.Column="4" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_11" />

<Button Grid.Column="5" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_20" />

<Button Height="40" Width="80" Grid.Column="6" Grid.Row="2" Content="График" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_28" Background="White" />

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_2" Header="Отель">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgHotel" Grid.ColumnSpan="3" Margin="20"/>

<DataGrid Name="dgRoom" Grid.ColumnSpan="3" Grid.Column="3" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_3" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_12" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_21" />

<Button Grid.Column="3" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_4" />

<Button Grid.Column="4" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_13" />

<Button Grid.Column="5" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_22" />

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_3" Header="Услуги">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgService" Grid.ColumnSpan="3" Margin="20,28,20,20"/>

<DataGrid Name="dgStatus" Grid.ColumnSpan="3" Grid.Column="3" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_5" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_14" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_23" />

<Button Grid.Column="3" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_6" />

<Button Grid.Column="4" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_15" />

<Button Grid.Column="5" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_24" />

<ComboBox DisplayMemberPath="NameServices" Name="cmbTypeServices" Grid.Column="0" Grid.Row="0" VerticalAlignment="Top" SelectionChanged="cmbTypeServices\_SelectionChanged" />

<TextBox x:Name="txtSearch" Grid.Column="1" Grid.Row="0" VerticalAlignment="Top" Margin="5" TextChanged="txtSearch\_TextChanged">

<TextBox.Style>

<Style TargetType="TextBox" xmlns:sys="clr-namespace:System;assembly=mscorlib">

<Style.Resources>

<VisualBrush x:Key="CueBannerBrush" AlignmentX="Left" AlignmentY="Center" Stretch="None">

<VisualBrush.Visual>

<Label Content="Поиск" Foreground="White" />

</VisualBrush.Visual>

</VisualBrush>

</Style.Resources>

<Style.Triggers>

<Trigger Property="Text" Value="{x:Static sys:String.Empty}">

<Setter Property="Background" Value="{StaticResource CueBannerBrush}" />

</Trigger>

</Style.Triggers>

</Style>

</TextBox.Style>

</TextBox>

</Grid>

</TabItem>

<TabItem Loaded="TabItem\_Loaded\_4" Header="Пользователи">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="166\*"/>

<RowDefinition Height="37\*"/>

</Grid.RowDefinitions>

<DataGrid Name="dgUser" Grid.ColumnSpan="3" Margin="20"/>

<DataGrid Name="dgRole" Grid.ColumnSpan="3" Grid.Column="3" Margin="20"/>

<Button Grid.Column="0" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_7" />

<Button Grid.Column="1" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_16" />

<Button Grid.Column="2" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_25" />

<Button Grid.Column="3" Grid.Row="1" Content="Добавить" Height="40" Width="80" Background="#FFCAEDBE" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_8" />

<Button Grid.Column="4" Grid.Row="1" Content="Изменить" Height="40" Width="80" Background="#FFD1D0AB" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_17" />

<Button Grid.Column="5" Grid.Row="1" Content="Удалить" Height="40" Width="80" Background="#FFED8D8D" HorizontalAlignment="Center" VerticalAlignment="Center" Click="Button\_Click\_26" />

</Grid>

</TabItem>

<TabItem Header="График">

<Grid>

<Grid.Background>

<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">

<GradientStop Color="#FF9D3DE9" Offset="0.278"/>

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FFCD2BEB" Offset="0.603"/>

</LinearGradientBrush>

</Grid.Background>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ScottPlot:WpfPlot Grid.Column="0" x:Name="PlotGraph" VerticalAlignment="Center" HorizontalAlignment="Center" Background="Black" Foreground="DarkGreen"/>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Window>

Admin.xaml.cs:

using Microsoft.Win32;

using Newtonsoft.Json;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Net.Http;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using WpfHotel.Models;

namespace WpfHotel

{

/// <summary>

/// Логика взаимодействия для Admin.xaml

/// </summary>

public partial class Admin : Window

{

public Admin()

{

InitializeComponent();

}

public int count;

List<Service> data6;

public async Task LoadData(int count)

{

HttpClient client = new HttpClient();

switch (count)

{

case 1:

try {

dgBooking.ItemsSource = null;

HttpResponseMessage response = await client.GetAsync($"https://192.168.74.212:7071/api/Bookings");

string json1 = await response.Content.ReadAsStringAsync();

List<Booking> data = JsonConvert.DeserializeObject<List<Booking>>(json1);

dgBooking.ItemsSource = data;

var columns = dgBooking.Columns;

columns[1].Header = "Дата прибытия";

columns[2].Header = "Дата отбытия";

columns[3].Header = "Идентификатор пользователя";

columns[4].Header = "Идентификатор услуги";

columns[5].Header = "Идентификатор номера";

columns[6].Header = "Бронирование";

columns[0].Visibility = Visibility.Collapsed;

}

catch { await LoadData(1); }

break;

case 2:

try

{

dgCheckIn.ItemsSource = null;

dgCheckOut.ItemsSource = null;

HttpResponseMessage response2 = await client.GetAsync($"https://192.168.74.212:7071/api/CheckIns");

string json2 = await response2.Content.ReadAsStringAsync();

List<CheckIn> data2 = JsonConvert.DeserializeObject<List<CheckIn>>(json2);

dgCheckIn.ItemsSource = data2;

var columns2 = dgCheckIn.Columns;

columns2[1].Header = "Статус заселения";

columns2[2].Header = "Идентификатор пользователя";

columns2[3].Header = "Идентификатор бронирования";

columns2[0].Visibility = Visibility.Collapsed;

HttpResponseMessage response3 = await client.GetAsync($"https://192.168.74.212:7071/api/CheckOuts");

string json3 = await response3.Content.ReadAsStringAsync();

List<CheckOut> data3 = JsonConvert.DeserializeObject<List<CheckOut>>(json3);

dgCheckOut.ItemsSource = data3;

var columns3 = dgCheckOut.Columns;

columns3[0].Visibility = Visibility.Collapsed;

columns3[1].Header = "Дата оплаты";

columns3[2].Header = "Общая стоимость";

columns3[3].Header = "Идентификатор заселения";

columns3[4].Header = "Идентификатор пользователя";

} catch { await LoadData(2); }

break;

case 3:

try {

dgHotel.ItemsSource = null;

dgRoom.ItemsSource = null;

HttpResponseMessage response4 = await client.GetAsync($"https://192.168.74.212:7071/api/Hotels");

string json4 = await response4.Content.ReadAsStringAsync();

List<Hotel> data4 = JsonConvert.DeserializeObject<List<Hotel>>(json4);

dgHotel.ItemsSource = data4;

HttpResponseMessage response5 = await client.GetAsync($"https://192.168.74.212:7071/api/Rooms");

string json5 = await response5.Content.ReadAsStringAsync();

List<Room> data5 = JsonConvert.DeserializeObject<List<Room>>(json5);

dgRoom.ItemsSource = data5;

var columns4 = dgHotel.Columns;

columns4[1].Header = "Адрес отеля";

columns4[2].Header = "Рейтинг отеля";

columns4[3].Header = "Номер телефона отеля";

columns4[4].Header = "Email отеля";

columns4[0].Visibility = Visibility.Collapsed;

var columns5 = dgRoom.Columns;

columns5[1].Header = "Количество комнат";

columns5[2].Header = "Цена";

columns5[3].Header = "Номер комнаты";

columns5[4].Header = "Тип комнаты";

columns5[5].Header = "Статус комнаты";

columns5[6].Header = "Идентификатор отеля";

columns5[0].Visibility = Visibility.Collapsed;

} catch { await LoadData(3); }

break;

case 4:

try

{

dgService.ItemsSource = null;

dgStatus.ItemsSource = null;

HttpResponseMessage response6 = await client.GetAsync($"https://192.168.74.212:7071/api/Services");

string json6 = await response6.Content.ReadAsStringAsync();

data6 = JsonConvert.DeserializeObject<List<Service>>(json6);

dgService.ItemsSource = data6;

cmbTypeServices.ItemsSource = data6;

HttpResponseMessage response7 = await client.GetAsync($"https://192.168.74.212:7071/api/Status");

string json7 = await response7.Content.ReadAsStringAsync();

List<Status> data7 = JsonConvert.DeserializeObject<List<Status>>(json7);

dgStatus.ItemsSource = data7;

var columns6 = dgService.Columns;

columns6[1].Header = "Название услуги";

columns6[2].Header = "Описание услуги";

columns6[3].Header = "Цена услуги";

columns6[0].Visibility = Visibility.Collapsed;

var columns7 = dgStatus.Columns;

columns7[1].Header = "Доступность";

columns7[0].Visibility = Visibility.Collapsed;

} catch { await LoadData(4); }

break;

case 5:

try

{

dgUser.ItemsSource = null;

dgRole.ItemsSource = null;

HttpResponseMessage response8 = await client.GetAsync($"https://192.168.74.212:7071/api/Users");

string json8 = await response8.Content.ReadAsStringAsync();

List<User> data8 = JsonConvert.DeserializeObject<List<User>>(json8);

dgUser.ItemsSource = data8;

HttpResponseMessage response9 = await client.GetAsync($"https://192.168.74.212:7071/api/Roles");

string json9 = await response9.Content.ReadAsStringAsync();

List<Role> data9 = JsonConvert.DeserializeObject<List<Role>>(json9);

dgRole.ItemsSource = data9;

var columns8 = dgUser.Columns;

columns8[1].Header = "Имя";

columns8[2].Header = "Фамилия";

columns8[3].Header = "Отчество";

columns8[4].Header = "Email";

columns8[5].Header = "Номер телефона";

columns8[6].Header = "Серия паспорта";

columns8[7].Header = "Номер паспорта";

columns8[8].Header = "Логин";

columns8[9].Header = "Пароль";

columns8[10].Header = "Роль";

columns8[0].Visibility = Visibility.Collapsed;

var columns9 = dgRole.Columns;

columns9[1].Header = "Название роли";

columns9[0].Visibility = Visibility.Collapsed;

} catch { await LoadData(5); }

break;

}

}

private async void TabItem\_Loaded(object sender, RoutedEventArgs e)

{

count = 1;

await LoadData(count);

}

private async void TabItem\_Loaded\_1(object sender, RoutedEventArgs e)

{

count = 2;

await LoadData(count);

}

private async void TabItem\_Loaded\_2(object sender, RoutedEventArgs e)

{

count = 3;

await LoadData(count);

}

private async void TabItem\_Loaded\_3(object sender, RoutedEventArgs e)

{

count = 4;

await LoadData(count);

}

private async void TabItem\_Loaded\_4(object sender, RoutedEventArgs e)

{

count = 5;

await LoadData(count);

}

public async void CreateTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

properties[0].SetValue(data, null);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PostAsync($"https://192.168.74.212:7071/api/{ClassName}s", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click(object sender, RoutedEventArgs e)

{

CreateTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_1(object sender, RoutedEventArgs e)

{

CreateTable<CheckIn>(dgCheckIn);

await LoadData(2);

}

private async void Button\_Click\_2(object sender, RoutedEventArgs e)

{

CreateTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_3(object sender, RoutedEventArgs e)

{

CreateTable<Hotel>(dgHotel);

await LoadData(3);

}

private async void Button\_Click\_4(object sender, RoutedEventArgs e)

{

CreateTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_5(object sender, RoutedEventArgs e)

{

CreateTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_6(object sender, RoutedEventArgs e)

{

CreateTable<Status>(dgStatus);

await LoadData(4);

}

private async void Button\_Click\_7(object sender, RoutedEventArgs e)

{

CreateTable<User>(dgUser);

await LoadData(5);

}

private async void Button\_Click\_8(object sender, RoutedEventArgs e)

{

CreateTable<Role>(dgRole);

await LoadData(5);

}

public async void PutTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int ID = (int)properties[0].GetValue(data);

string jsonString = JsonConvert.SerializeObject(data);

HttpContent content = new StringContent(jsonString, Encoding.UTF8, "application/json");

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.PutAsync($"https://192.168.74.212:7071/api/{ClassName}s/{ID}", content);

await LoadData(count);

}

else

{

MessageBox.Show("Выберете строку!");

}

}

private async void Button\_Click\_9(object sender, RoutedEventArgs e)

{

PutTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_10(object sender, RoutedEventArgs e)

{

PutTable<CheckIn>(dgCheckIn);

await LoadData(2);

}

private async void Button\_Click\_11(object sender, RoutedEventArgs e)

{

PutTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_12(object sender, RoutedEventArgs e)

{

PutTable<Hotel>(dgHotel);

await LoadData(3);

}

private async void Button\_Click\_13(object sender, RoutedEventArgs e)

{

PutTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_14(object sender, RoutedEventArgs e)

{

PutTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_15(object sender, RoutedEventArgs e)

{

PutTable<Status>(dgStatus);

await LoadData(4);

}

private async void Button\_Click\_16(object sender, RoutedEventArgs e)

{

PutTable<User>(dgUser);

await LoadData(5);

}

private async void Button\_Click\_17(object sender, RoutedEventArgs e)

{

PutTable<Role>(dgRole);

await LoadData(5);

}

public async Task DeleteTable<T>(DataGrid name)

{

HttpClient client = new HttpClient();

var selectedItems = name.SelectedItems.Cast<T>().ToList();

if (selectedItems.Count == 1)

{

var data = selectedItems.FirstOrDefault();

var properties = typeof(T).GetProperties();

int id = (int)properties[0].GetValue(data);

string ClassName = typeof(T).Name;

switch (ClassName)

{

case ("Status"):

ClassName = "Statu";

break;

}

HttpResponseMessage response = await client.DeleteAsync($"https://192.168.74.212:7071/api/{ClassName}s/{id}");

await LoadData(count);

}

else

{

MessageBox.Show("Выберете одну строку для удаления!");

}

}

private async void Button\_Click\_18(object sender, RoutedEventArgs e)

{

await DeleteTable<Booking>(dgBooking);

await LoadData(1);

}

private async void Button\_Click\_19(object sender, RoutedEventArgs e)

{

await DeleteTable<CheckIn>(dgCheckIn);

await LoadData(2);

}

private async void Button\_Click\_20(object sender, RoutedEventArgs e)

{

await DeleteTable<CheckOut>(dgCheckOut);

await LoadData(2);

}

private async void Button\_Click\_21(object sender, RoutedEventArgs e)

{

await DeleteTable<Hotel>(dgHotel);

await LoadData(3);

}

private async void Button\_Click\_22(object sender, RoutedEventArgs e)

{

await DeleteTable<Room>(dgRoom);

await LoadData(3);

}

private async void Button\_Click\_23(object sender, RoutedEventArgs e)

{

await DeleteTable<Service>(dgService);

await LoadData(4);

}

private async void Button\_Click\_24(object sender, RoutedEventArgs e)

{

await DeleteTable<Status>(dgStatus);

await LoadData(4);

}

private async void Button\_Click\_25(object sender, RoutedEventArgs e)

{

await DeleteTable<User>(dgUser);

await LoadData(5);

}

private async void Button\_Click\_26(object sender, RoutedEventArgs e)

{

await DeleteTable<Role>(dgRole);

await LoadData(5);

}

private void ExportToSql\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;User ID=sa;Password=123"))

{

connection.Open();

List<Booking> data = dgBooking.ItemsSource.Cast<Booking>().ToList();

foreach (var item in data)

{

string insertQuery = $"INSERT INTO Booking (Arrival\_date, Departure\_date, User\_ID, Service\_ID, Room\_ID) " +

$"VALUES ('{item.ArrivalDate}', '{item.DepartureDate}', {item.UserId}, {item.ServiceId}, {item.RoomId})";

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

{

command.ExecuteNonQuery();

}

}

}

MessageBox.Show("Export to SQL completed.");

}

private void ExportToCsv\_Click(object sender, RoutedEventArgs e)

{

List<Booking> data = dgBooking.ItemsSource.Cast<Booking>().ToList();

StringBuilder csv = new StringBuilder();

csv.AppendLine("IdBooking,ArrivalDate,DepartureDate,UserId,ServiceId,RoomId");

foreach (var item in data)

{

csv.AppendLine($"{item.IdBooking},{item.ArrivalDate},{item.DepartureDate},{item.UserId},{item.ServiceId},{item.RoomId}");

}

File.WriteAllText("Booking.csv", csv.ToString());

MessageBox.Show("Export to CSV completed.");

}

private void ImportFromSql\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;User ID=sa;Password=123"))

{

connection.Open();

string selectQuery = "SELECT \* FROM Booking";

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

{

List<Booking> importedData = new List<Booking>();

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Booking booking = new Booking

{

IdBooking = Convert.ToInt32(reader["ID\_Booking"]),

ArrivalDate = Convert.ToDateTime(reader["Arrival\_date"]),

DepartureDate = Convert.ToDateTime(reader["Departure\_date"]),

UserId = Convert.ToInt32(reader["User\_ID"]),

ServiceId = Convert.ToInt32(reader["Service\_ID"]),

RoomId = Convert.ToInt32(reader["Room\_ID"])

};

importedData.Add(booking);

}

}

dgBooking.ItemsSource = importedData;

}

}

MessageBox.Show("Import from SQL completed.");

}

private void ImportFromCsv\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "CSV files (\*.csv)|\*.csv|All files (\*.\*)|\*.\*";

if (openFileDialog.ShowDialog() == true)

{

string[] lines = File.ReadAllLines(openFileDialog.FileName);

List<Booking> importedData = new List<Booking>();

foreach (var line in lines.Skip(1)) // Skip header

{

var values = line.Split(',');

if (values.Length == 6)

{

Booking booking = new Booking

{

IdBooking = int.Parse(values[0]),

ArrivalDate = DateTime.Parse(values[1]),

DepartureDate = DateTime.Parse(values[2]),

UserId = int.Parse(values[3]),

ServiceId = int.Parse(values[4]),

RoomId = int.Parse(values[5])

};

importedData.Add(booking);

}

}

dgBooking.ItemsSource = importedData;

MessageBox.Show("Import from CSV completed.");

}

}

private void Button\_Click\_27(object sender, RoutedEventArgs e)

{

string databaseName = "Hotel"; // Замените на имя вашей базы данных

try

{

// Диалоговое окно выбора файла для сохранения резервной копии

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Backup files (\*.bak)|\*.bak";

saveFileDialog.FileName = $"{databaseName}\_Backup.bak";

if (saveFileDialog.ShowDialog() == true)

{

string backupPath = saveFileDialog.FileName;

using (SqlConnection connection = new SqlConnection("Data Source=ALEXANDER-LAPTO\\SQLEXPRESS;Initial Catalog=Hotel;User ID=sa;Password=123"))

{

connection.Open();

// Создание резервной копии базы данных

string backupQuery = $"BACKUP DATABASE [{databaseName}] TO DISK = '{backupPath}'";

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

{

command.ExecuteNonQuery();

}

}

MessageBox.Show($"Backup created successfully. Path: {backupPath}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Error creating backup: {ex.Message}");

}

}

private void cmbTypeServices\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

// Получаем выбранное значение из комбобокса

string selectedType = (string)(cmbTypeServices.SelectedItem as Service)?.NameServices;

// Фильтруем данные в data7 в зависимости от выбранного значения

List<Service> filteredData = data6.Where(s => s.NameServices == selectedType).ToList();

// Обновляем источник данных для DataGrid

dgService.ItemsSource = filteredData;

}

private void txtSearch\_TextChanged(object sender, TextChangedEventArgs e)

{

// Получаем текст из TextBox

string searchText = txtSearch.Text;

// Фильтруем данные в data7 в зависимости от введенного текста

List<Service> filteredData = data6.Where(s => s.NameServices.Contains(searchText)).ToList();

// Обновляем источник данных для DataGrid

dgService.ItemsSource = filteredData;

}

private void Button\_Click\_28(object sender, RoutedEventArgs e)

{

ClearPlot(PlotGraph);

List<CheckOut> data = GetDataFromGrid<CheckOut>(dgCheckOut); // получение данных из базы данных

PlotData<CheckOut>(data, PlotGraph); // генерация графика

}

public List<T> GetDataFromGrid<T>(DataGrid name)

{

List<T> itemsSourceList = new List<T>();

if (name.ItemsSource is IEnumerable<T> itemsSource)

{

itemsSourceList = itemsSource.ToList();

}

else if (name.ItemsSource is IEnumerable itemsSourceNonGeneric)

{

foreach (var item in itemsSourceNonGeneric)

{

if (item is T tItem)

{

itemsSourceList.Add(tItem);

}

}

}

return itemsSourceList;

}

public void PlotData<T>(List<T> data, ScottPlot.WpfPlot plotControl)

{

double[] xValues;

double[] yValues;

// выбираем данные для графика из List<T> для таблицы 1

// например, первый столбец - это x, второй столбец - это y

xValues = data.Select(item => Convert.ToDouble(item.GetType().GetProperty("CheckInId").GetValue(item))).ToArray();

yValues = data.Select(item => Convert.ToDouble(item.GetType().GetProperty("TotalCost").GetValue(item))).ToArray();

plotControl.Plot.XLabel("ID");

plotControl.Plot.YLabel("Цена");

plotControl.Plot.Title("График данных");

// добавляем точки на существующий объект ScottPlot.WpfPlot

plotControl.Plot.AddScatter(xValues, yValues);

// обновляем отображение графика

plotControl.Render();

}

public void ClearPlot(ScottPlot.WpfPlot plotControl)

{

plotControl.Reset();

}

private async void dgCheckOut\_Loaded(object sender, RoutedEventArgs e)

{

await LoadData(3);

}

private void dgBooking\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

}

private void Button\_Click\_29(object sender, RoutedEventArgs e)

{

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

Close();

}

}

}