

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

«**Задание 7 - Индивидуальное задание**»

Выполнил: Миронов Илья Владимирович

Группа: ПР-31

Преподаватель: Мирошниченко Г.В.

2024

Содержание

[1. **Вариант 3. «Онлайн-бронирование гостиницы».** 3](#_Toc118960339)

[1.1 Описание задачи 3](#_Toc118960340)

[1.2 Описание разработанных функций 5](#_Toc118960342)

[1.3 Используемые библиотеки 16](#_Toc118960344)

[1.4 Используемые инструменты 17](#_Toc118960346)

# 1. **Вариант 3. «Онлайн-бронирование гостиницы».**

## Описание задачи

Создать проект с 3-я страницами:

А) На главной странице создать список онлайн-бронирование гостиницы (заказ гостиницы) со следующими полями с возможностью группировки по категории, наверху должен быть заголовок для страницы – Онлайн-бронирование гостиницы – выделен синим курсивом:

Название (с фото), Номер, Директор, Телефон, Категория, Число мест, Стоимость, Фамилия, Адрес, Возраст, Дата заезда, Срок проживания, Оплата (безналичный расчет или наличный расчет).

После выбора гостиницы снять выделение элемента списка.

Внизу страницы 2 кнопки – 1. «Выбор количества мест» (переход на 3-ую модальную страницу), 2. «Расчет стоимости проживания» (переход на 2-ю модальную страницу – стоимость рассчитывается на 1 человека). Прописать переходы с помощью методов навигации.

Б) На третьей странице – заголовок списка – «Выбор количества мест», название гостиницы (с фото), номер и цена номера по выбранному элементу списка (гостиниц) с главной страницы, ввод количества мест – сделать ограничение: количество мест не может быть больше 4-х и ввод количества номеров (количество номеров не должно быть больше количества выбранных мест для проживания), выбор категории номера из списка (значение выбранной категории записывается в список в поле категория на главной странице).

Внизу 2 кнопки – 1. «Расчет стоимости проживания» (переход на 2-ую модальную страницу) и 2-ая кнопка – «Возврат на главную страницу» – прописать переходы с помощью методов навигации.

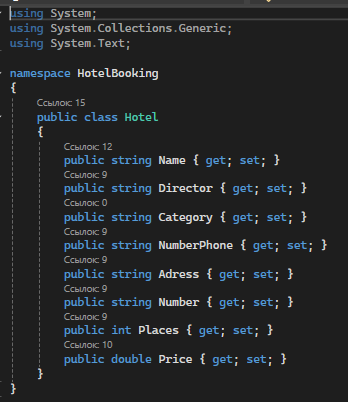
В) На 2-ой модальной странице калькулятор для расчета стоимости проживания в зависимости от количества мест или количества номеров (если количество мест равно 4-м, берется цена за место\*50%\*4 (берется со 2-ой страницы, если переход происходит с главной страницы, то количество мест равно 1), срок проживания (вводится на 2-ой странице и сохраняется в поле срок проживания на главной странице) и % скидки от возраста клиента гостиницы (процент скидки: 40% - дети до 7 лет, пенсионеры 20% - студенты и учащиеся, 10 % - многодетные семьи) . Рассчитанная стоимость отражается на главной странице в поле – Стоимость.

Внизу

1. Кнопка – «Выбор количества мест» - переход на третью страницу
2. Кнопка для возврата на корневую страницу
3. Прописать методы для перехода на третью страницу и для возврата на корневую страницу.

## Описание разработанных функций

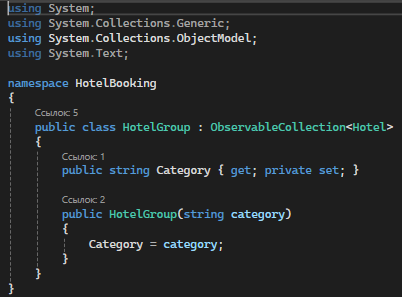
Класс Hotel (рис 1):



*Рис.1*

В данном классе прописаны свойства номера отеля, отображаемого на главной странице в списке.

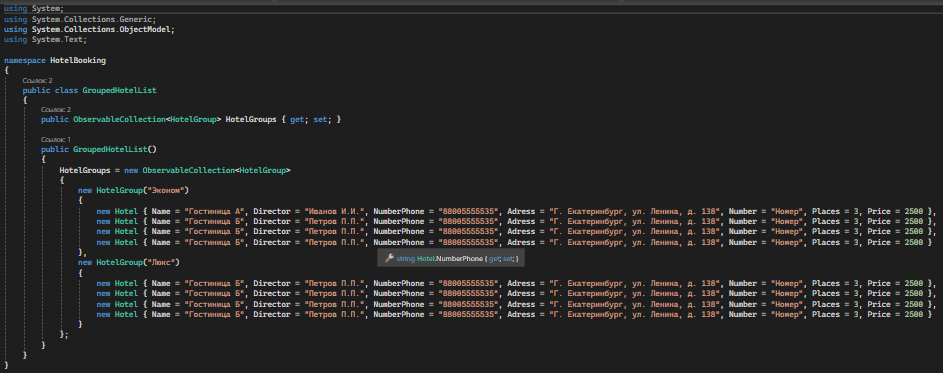
Класс HotelBooking (рис 2):



*Рис.2*

В данном классе прописан конструктор класса Hotel, который тут же присваивает свойству категория класса HotelGroup значение и создаёт объект Hotel.

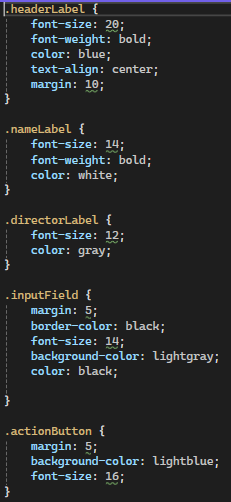
Класс GroupedHotelList (рис 3):



*Рис.3*

В данном классе прописан список групп номеров отелей.

Файл styles.css (рис 4):



*Рис.4*

В данном файле прописана стилизация для элементов экранов.

Класс MainPage:

namespace HotelBooking

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class MainPage : ContentPage

{

public MainPage()

{

InitializeComponent();

Title = "Онлайн-бронирование гостиницы";

Label header = new Label

{

Text = "Онлайн-бронирование гостиницы",

StyleClass = new[] { "headerLabel" }

};

var hotels = new GroupedHotelList();

hotelListView = new ListView

{

HasUnevenRows = true,

IsGroupingEnabled = true,

ItemsSource = hotels.HotelGroups,

GroupDisplayBinding = new Binding("Category"),

ItemTemplate = new DataTemplate(() =>

{

var stackLayout = new StackLayout

{

Padding = new Thickness(10),

Spacing = 5,

Orientation = StackOrientation.Vertical

};

var nameLabel = new Label

{

FontSize = 16,

FontAttributes = FontAttributes.Bold,

StyleClass = new[] { "nameLabel" }

};

nameLabel.SetBinding(Label.TextProperty, "Name");

var directorLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

directorLabel.SetBinding(Label.TextProperty, "Director");

var numberPhoneLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

numberPhoneLabel.SetBinding(Label.TextProperty, "NumberPhone", stringFormat: "Номер телефона: {0}");

var adressLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

adressLabel.SetBinding(Label.TextProperty, "Adress");

var numberLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

numberLabel.SetBinding(Label.TextProperty, "Number");

var placesLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

placesLabel.SetBinding(Label.TextProperty, "Places", stringFormat: "Количество мест: {0}");

var priceLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

priceLabel.SetBinding(Label.TextProperty, "Price", stringFormat: "Цена: {0} руб.");

stackLayout.Children.Add(nameLabel);

stackLayout.Children.Add(directorLabel);

stackLayout.Children.Add(numberPhoneLabel);

stackLayout.Children.Add(adressLabel);

stackLayout.Children.Add(numberLabel);

stackLayout.Children.Add(placesLabel);

stackLayout.Children.Add(priceLabel);

var cell = new ViewCell { View = stackLayout };

return cell;

}

)

};

Button selectRoomsButton = new Button

{

Text = "Выбор количества мест",

StyleClass = new[] { "actionButton" },

};

selectRoomsButton.Clicked += goToSelection;

Button calculateCostButton = new Button

{

Text = "Расчет стоимости проживания",

StyleClass = new[] { "actionButton" },

};

calculateCostButton.Clicked += goToCalculation;

ageEntry.TextChanged += (s, e) =>

{

if (int.TryParse(ageEntry.Text, out int result) && result > 120)

{

ageEntry.Text = "120";

} else if (int.TryParse(ageEntry.Text, out int result1) && result1 < 0 )

{

ageEntry.Text = "0";

}

};

checkInDatePicker.MinimumDate = DateTime.Today;

Content = new StackLayout

{

Padding = new Thickness(10, 20),

Children = { header, hotelListView, surnameEntry, ageEntry, stackWithPicker, stackPicker, selectRoomsButton, calculateCostButton }

};

}

async public void goToSelection(object sender, EventArgs e)

{

try

{

if (!string.IsNullOrEmpty(surnameEntry.Text) && !string.IsNullOrEmpty(ageEntry.Text))

{

if (int.Parse(ageEntry.Text) >= 18)

{

if (hotelListView.SelectedItem != null)

if (CheckForSurname(surnameEntry.Text))

if (picker.SelectedIndex != -1)

await Navigation.PushModalAsync(new RoomSelectionPage(hotelListView.SelectedItem));

else

DisplayAlert("Ошибка", "Выберите вид оплаты", "OK");

else

DisplayAlert("Ошибка", "В фамилии можно вводить только буквы", "OK");

else

DisplayAlert("Ошибка", "Выберите номер, нажав на него", "OK");

}

else

DisplayAlert("Ошибка", "Несовершеннолетние не могут бронировать номера", "OK");

}

else

DisplayAlert("Ошибка", "Заполните все поля", "OK");

}

catch

{

DisplayAlert("Ошибка", "Введите свой возрастах в полных годах", "OK");

}

}

async public void goToCalculation(object sender, EventArgs e)

{

try

{

if (!string.IsNullOrEmpty(surnameEntry.Text) && !string.IsNullOrEmpty(ageEntry.Text))

{

if (int.Parse(ageEntry.Text) >= 18)

{

if (hotelListView.SelectedItem != null)

if (CheckForSurname(surnameEntry.Text))

if (picker.SelectedIndex != -1)

await Navigation.PushModalAsync(new CostCalculationPage(hotelListView.SelectedItem, 1));

else

DisplayAlert("Ошибка", "Выберите вид оплаты", "OK");

else

DisplayAlert("Ошибка", "В фамилии можно вводить только буквы", "OK");

else

DisplayAlert("Ошибка", "Выберите номер, нажав на него", "OK");

}

else

DisplayAlert("Ошибка", "Несовершеннолетние не могут бронировать номера", "OK");

}

else

DisplayAlert("Ошибка", "Заполните все поля", "OK");

}

catch

{

DisplayAlert("Ошибка", "Введите свой возрастах в полных годах", "OK");

}

}

public bool CheckForSurname(string surname)

{

foreach (var ch in surname)

{

if (!char.IsLetter(ch))

return false;

}

return true;

}

}

}

В данном классе прописана основная логика приложения и частичная отрисовка экрана приложения.

Класс RoomSelectionPage:

namespace HotelBooking

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class RoomSelectionPage : ContentPage

{

Hotel hotel1;

public RoomSelectionPage(object hotel)

{

hotel1 = hotel as Hotel;

var stackLayout = new StackLayout

{

Padding = new Thickness(10),

Spacing = 5,

Orientation = StackOrientation.Vertical

};

if (hotel1.Name != null)

{

var nameLabel = new Label

{

FontSize = 16,

FontAttributes = FontAttributes.Bold,

StyleClass = new[] { "nameLabel" }

};

nameLabel.Text = hotel1.Name;

var directorLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

directorLabel.Text = hotel1.Director;

var numberPhoneLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

numberPhoneLabel.Text = $"Номер телефона: {hotel1.NumberPhone}";

var adressLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

adressLabel.Text = hotel1.Adress;

var numberLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

numberLabel.Text = hotel1.Number;

var placesLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

placesLabel.Text = $"Количество мест: {hotel1.Places}";

var priceLabel = new Label

{

FontSize = 14,

TextColor = Color.Gray,

StyleClass = new[] { "directorLabel" }

};

priceLabel.Text = $"Цена: {hotel1.Price} руб.";

stackLayout.Children.Add(nameLabel);

stackLayout.Children.Add(directorLabel);

stackLayout.Children.Add(numberPhoneLabel);

stackLayout.Children.Add(adressLabel);

stackLayout.Children.Add(numberLabel);

stackLayout.Children.Add(placesLabel);

stackLayout.Children.Add(priceLabel);

}

InitializeComponent();

Title = "Выбор количества мест";

Label header = new Label

{

Text = "Выбор количества мест",

StyleClass = new[] { "headerLabel" }

};

numberOfPlaces.TextChanged += (s, e) =>

{

try

{

if (int.TryParse(numberOfPlaces.Text, out int result) && result > 4)

{

numberOfPlaces.Text = "4";

}

else if (int.TryParse(numberOfPlaces.Text, out int result1) && result1 < 1)

{

numberOfPlaces.Text = "1";

}

}

catch

{

DisplayAlert("Ошибка", "Нельзя вводить отрицательные и/или нецелые числа", "OK");

}

};

numberOfRooms.TextChanged += (s, e) =>

{

try

{

if (!string.IsNullOrEmpty(numberOfPlaces.Text))

{

if (int.TryParse(numberOfRooms.Text, out int result) && result > int.Parse(numberOfPlaces.Text))

{

numberOfRooms.Text = numberOfPlaces.Text;

}

}

else if (int.TryParse(numberOfPlaces.Text, out int result1) && result1 < 1)

{

numberOfPlaces.Text = "1";

}

else

{

numberOfRooms.Text = "";

DisplayAlert("Ошибка", "Введите сначала количество мест", "OK");

}

}

catch

{

DisplayAlert("Ошибка", "Нельзя вводить отрицательные и/или нецелые значения", "OK");

}

};

Button calculateCostButton = new Button

{

Text = "Расчет стоимости проживания",

StyleClass = new[] { "actionButton" },

};

calculateCostButton.Clicked += goToCalculate;

Button returnToMainButton = new Button

{

Text = "Возврат на главную страницу",

StyleClass = new[] { "actionButton" },

Command = new Command(async() =>

{

try

{

await Navigation.PopModalAsync();

}

catch

{

}

})

};

Content = new StackLayout

{

Padding = new Thickness(10, 20),

Children = { header, stackLayout, numberOfPlaces, numberOfRooms, calculateCostButton, returnToMainButton }

};

}

async public void goToCalculate(object sender, EventArgs e)

{

try

{

if (!string.IsNullOrEmpty(numberOfPlaces.Text) && !string.IsNullOrEmpty(numberOfRooms.Text))

{

if (hotel1.Name != null && int.Parse(numberOfPlaces.Text) > 0 && int.Parse(numberOfRooms.Text) > 0)

await Navigation.PushModalAsync(new CostCalculationPage(hotel1, int.Parse(numberOfPlaces.Text)));

else if (int.Parse(numberOfPlaces.Text) < 1 || int.Parse(numberOfRooms.Text) < 1)

DisplayAlert("Ошибка", "Нельзя вводить значения меньше единицы", "OK");

else

DisplayAlert("Ошибка", "Отель для заселения не был выбран", "OK");

}

else

DisplayAlert("Ошибка", "Заполните все поля", "OK");

}

catch

{

DisplayAlert("Ошибка", "Вводите только целые адекватные значения", "OK");

}

}

}

}

В данном классе прописана основная логика приложения и частичная отрисовка экрана приложения.

Класс CostCalculationPage:

namespace HotelBooking

{

[XamlCompilation(XamlCompilationOptions.Compile)]

public partial class CostCalculationPage : ContentPage

{

public CostCalculationPage(object hotel, int numberOfPlaces)

{

Hotel hotel1 = hotel as Hotel;

int places = numberOfPlaces;

InitializeComponent();

Title = "Калькулятор стоимости";

Label header = new Label

{

Text = "Калькулятор стоимости",

StyleClass = new[] { "headerLabel" }

};

Entry duration = new Entry { Placeholder = "Срок проживания в днях", Keyboard = Keyboard.Numeric, StyleClass = new[] { "inputField" }, PlaceholderColor = Color.Black };

Label label = new Label { StyleClass = new[] { "nameLabel" }, HorizontalTextAlignment = TextAlignment.Center };

duration.TextChanged += (s, e) =>

{

try

{

if (int.TryParse(duration.Text, out int result) && result > 365)

{

duration.Text = "365";

}

else if (int.TryParse(duration.Text, out int result1) && result1 < 1)

{

duration.Text = "1";

}

}

catch

{

DisplayAlert("Ошибка", "Нельзя вводить отрицательные и/или нецелые значения", "OK");

}

};

Button calculateButton = new Button

{

Text = "Рассчитать стоимость",

StyleClass = new[] { "actionButton" },

Command = new Command(() =>

{

try

{

if (!string.IsNullOrEmpty(duration.Text))

{

if (hotel1.Name != null)

{

if (int.Parse(duration.Text) >= 1)

{

double sum = hotel1.Price \* int.Parse(duration.Text) \* numberOfPlaces;

if (numberOfPlaces == 4)

sum /= 2;

label.Text = $"Итого: {sum}";

}

else

DisplayAlert("Ошибка", "Нельзя вводить значения меньше единицы", "OK");

}

else

DisplayAlert("Ошибка", "Отель не был выбран, вернитесь на главную страницу, заполните все данные, выберите отель и нажмите кнопку 'Рассчет стоимости проживания'", "OK");

}

else

DisplayAlert("Ошибка", "Заполните поле", "OK");

}

catch

{

DisplayAlert("Ошибка", "Вводите только целые адекватные значения", "OK");

}

})

};

Content = new StackLayout

{

Padding = new Thickness(10, 20),

Children = { header, duration, label, calculateButton }

};

}

}

}

В данном классе прописана основная логика приложения и частичная отрисовка экрана приложения.

Класс App:

namespace HotelBooking

{

public partial class App : Application

{

public App()

{

InitializeComponent();

DependencyService.Register<MockDataStore>();

MainPage = new CarouselPage

{

Children = { new MainPage(), new CostCalculationPage(new Hotel(), 0), new RoomSelectionPage(new Hotel()) }

};

}

protected override void OnStart()

{

}

protected override void OnSleep()

{

}

protected override void OnResume()

{

}

}

}

В данном классе прописан CarouselPage для переключения между основными экранами свайпом влево-вправо.

1.3 Используемые библиотеки

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

1.5 Используемые инструменты

## Visual Studio Community 2022.

# Язык: C#.