Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего  
образования

«Ивановский государственный энергетический университет им. В. И. Ленина»

Кафедра программного обеспечения компьютерных систем

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ 2**

**на тему:**

**«Интернет-магазин»**

**по дисциплине:**

**Разработка мобильных приложений**

Обучающийся:

студент гр. 3-42 \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Шарабанов

*(подпись)*

Руководитель от университета:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.Д. Малафеев

*(подпись)*

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(оценка промежуточной аттестации)*

Иваново 2022

**Разбиение на View и ViewModel**

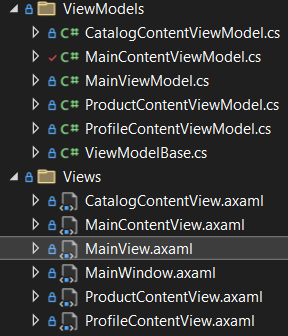
****

Рис.1 – Разработка окон

Были созданы следующие окна: Главное окно, окно главного контента, окно контента товара, окно контента профиля, окно контента всех каталога.

Для каждого окна были созданы соответствующие ViewModels.

**Реализация окон**

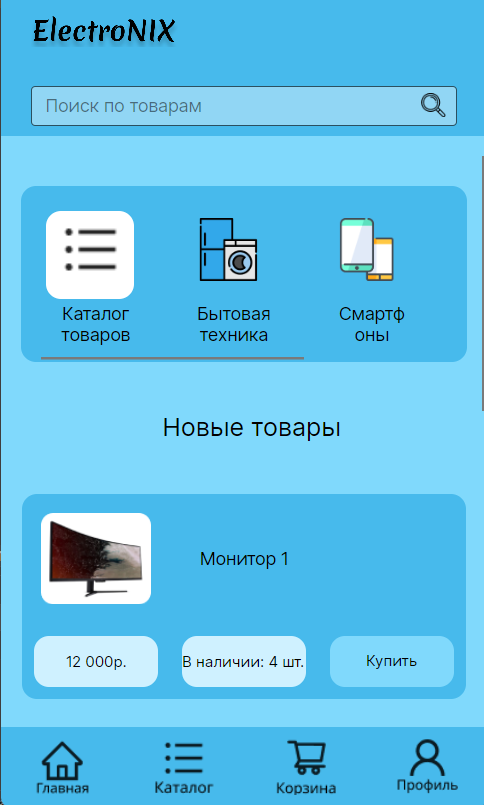


Рис. 2 – Реализация главного окна

В главном окне размечены меню с логотипом и поиском, и меню переключение разделов. Сам контент изменяемый и изменяется с помощью Router в соответствии с документацией (https://docs.avaloniaui.net/docs/next/concepts/reactiveui/routing#routing-example):

<ScrollViewer Grid.Row="1" VerticalScrollBarVisibility="Auto">

<!-- Место для динамического содержимого -->

<rxui:RoutedViewHost Grid.Row="0" Router="{Binding Router}">

<rxui:RoutedViewHost.ViewLocator>

<!-- See AppViewLocator.cs section below -->

<app:AppViewLocator />

</rxui:RoutedViewHost.ViewLocator>

</rxui:RoutedViewHost>

</ScrollViewer>

Сам Router находится в соответствующем View model:

public class MainViewModel : ReactiveObject, IScreen

{

// The Router associated with this Screen.

// Required by the IScreen interface.

public RoutingState Router { get; } = new RoutingState();

// The command that navigates a user to first view model.

public ReactiveCommand<Unit, IRoutableViewModel> NavigateToMain { get; }

public ReactiveCommand<Unit, IRoutableViewModel> NavigateToProfile { get; }

public ReactiveCommand<Unit, IRoutableViewModel> NavigateToCatalog { get; }

public MainViewModel()

{

// Manage the routing state. Use the Router.Navigate.Execute

// command to navigate to different view models.

//

// Note, that the Navigate.Execute method accepts an instance

// of a view model, this allows you to pass parameters to

// your view models, or to reuse existing view models.

//

NavigateToMain = ReactiveCommand.CreateFromObservable(

() => Router.Navigate.Execute(new MainContentViewModel(this))

);

NavigateToProfile = ReactiveCommand.CreateFromObservable(

() => Router.Navigate.Execute(new ProfileContentViewModel(this))

);

NavigateToCatalog = ReactiveCommand.CreateFromObservable(

() => Router.Navigate.Execute(new CatalogContentViewModel(this))

);

Router.Navigate.Execute(new MainContentViewModel(this));

}

}

При смене View контекст окон будет меняться с помощью AppViewLocator:

public IViewFor ResolveView<T>(T viewModel, string contract = null) => viewModel switch

{

MainContentViewModel mainPageViewModel => new MainContentView { DataContext = mainPageViewModel },

ProductContentViewModel productPageViewModel => new ProductContentView { DataContext = productPageViewModel },

ProfileContentViewModel profilePageViewModel => new ProfileContentView { DataContext = profilePageViewModel },

CatalogContentViewModel catalogPageViewModel => new CatalogContentView { DataContext = catalogPageViewModel },

// ... добавьте другие модели представлений и представления здесь ...

\_ => throw new ArgumentOutOfRangeException(nameof(viewModel), $"Could not find view for view model type {viewModel.GetType()}."),

};

**Создание дочерних окон**

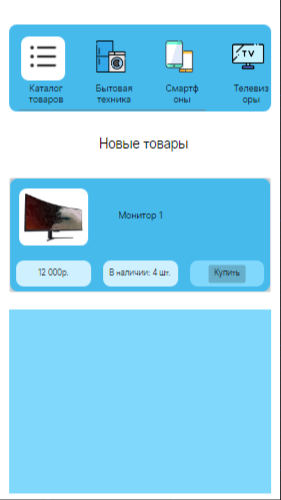


Рис. 3 – Окно контента главного окна

При создании окон, которых будут отображаться с помощью Router, в .axaml.cs окно было унаследовано от ReactiveUserControl:

public partial class MainContentView : ReactiveUserControl<MainContentViewModel>

{

public MainContentView()

{

this.WhenActivated(disposables => { });

AvaloniaXamlLoader.Load(this);

}

}

Во ViewModel экземпляр окна экрана передаётся в конструктор:

public class MainContentViewModel : ReactiveObject, IRoutableViewModel

{

public ObservableCollection<ProductDTO> Products { get; }

// Reference to IScreen that owns the routable view model.

public IScreen HostScreen { get; }

// Unique identifier for the routable view model.

public string UrlPathSegment { get; } = Guid.NewGuid().ToString().Substring(0, 5);

public MainContentViewModel(IScreen screen)

{

HostScreen = screen;

GoToProductPage = ReactiveCommand.Create(RunTheThing);

// Initialize the Products collection with example data

Products = new ObservableCollection<ProductDTO>

{

// Example data

new ProductDTO { ImageSource = ImagePathConverter.LoadFromResource(new Uri("avares://InternetShopMobileApp/Resources/monitor.png")), Title = "Монитор 1", Price = "12 000р.", Availability = "В наличии: 4 шт." },

new ProductDTO { ImageSource = ImagePathConverter.LoadFromResource(new Uri("avares://InternetShopMobileApp/Resources/monitor.png")), Title = "Монитор 1", Price = "12 000р.", Availability = "В наличии: 4 шт." },

new ProductDTO { ImageSource = ImagePathConverter.LoadFromResource(new Uri("avares://InternetShopMobileApp/Resources/monitor.png")), Title = "Монитор 1", Price = "12 000р.", Availability = "В наличии: 4 шт." },

// Add other products here

};

}

private IRoutableViewModel RunTheThing()

{

HostScreen.Router.Navigate.Execute(new ProductContentViewModel(HostScreen));

return null;

}

public ReactiveCommand<Unit, IRoutableViewModel> GoToProductPage { get; }

}

Таким образом окно отображается с помощью Router как необходимо:

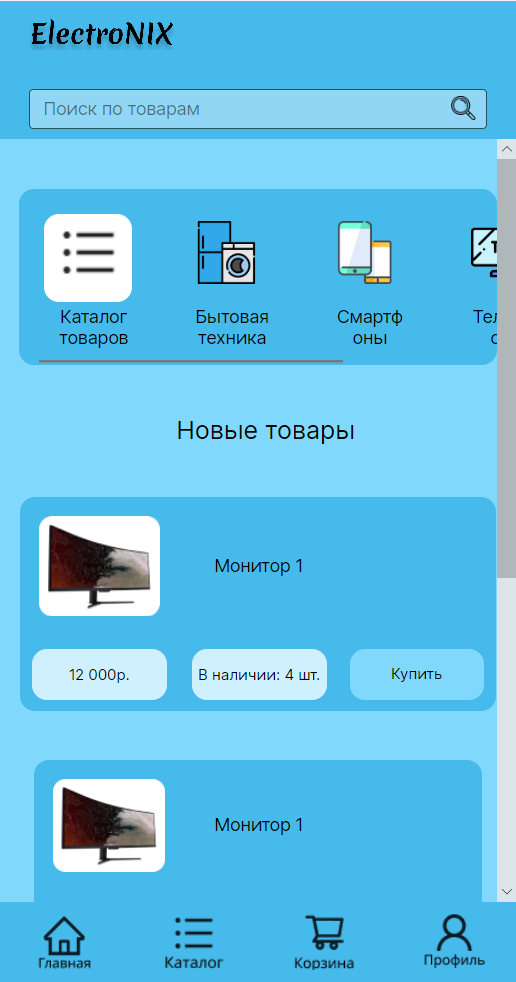


Рис.4 – Отображение главного контента

С помощью этого экземпляра можно переключать окна:

private IRoutableViewModel RunTheThing()

{

HostScreen.Router.Navigate.Execute(new ProductContentViewModel(HostScreen));

return null;

}

С помощью такого принципа работают все дочерние окна:

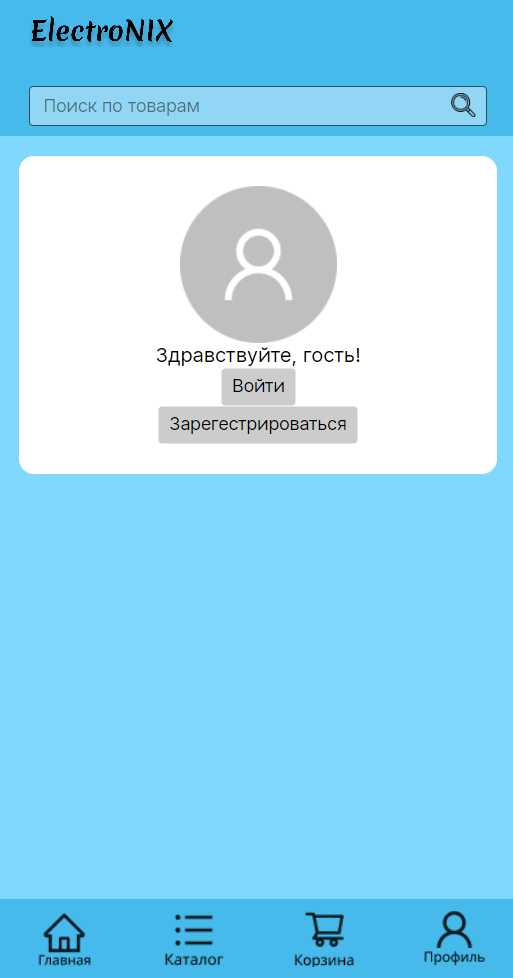


Рис.5 – Отображение контента профиля

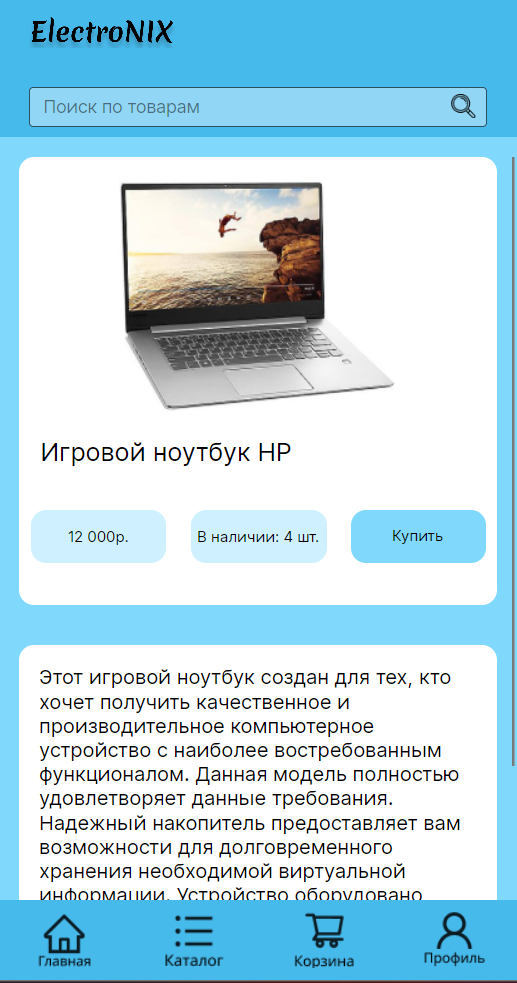


Рис.6 – Отображение контента товара



Рис.7 – Отображение контента каталога

**Привязка данных**

Была создана предварительная DTO:

public class ProductDTO

{

public Bitmap? ImageSource { get; set; }

public string Title { get; set; }

public string Price { get; set; }

public string Availability { get; set; }

}

В ViewModel создаю массив элементов:

public ObservableCollection<ProductDTO> Products { get; }

В конструкторе я задам примерные значения:

// Initialize the Products collection with example data

Products = new ObservableCollection<ProductDTO>

{

// Example data

new ProductDTO { ImageSource = ImagePathConverter.LoadFromResource(new Uri("avares://InternetShopMobileApp/Resources/monitor.png")), Title = "Монитор 1", Price = "12 000р.", Availability = "В наличии: 4 шт." },

new ProductDTO { ImageSource = ImagePathConverter.LoadFromResource(new Uri("avares://InternetShopMobileApp/Resources/monitor.png")), Title = "Монитор 1", Price = "12 000р.", Availability = "В наличии: 4 шт." },

new ProductDTO { ImageSource = ImagePathConverter.LoadFromResource(new Uri("avares://InternetShopMobileApp/Resources/monitor.png")), Title = "Монитор 1", Price = "12 000р.", Availability = "В наличии: 4 шт." },

// Add other products here

};

Теперь я могу забиндить массив во View:

<ListBox ItemsSource="{Binding Products}" Margin="16,20,16,10" Padding="0" Background="#80D9FC">

<ListBox.ItemTemplate>

<DataTemplate DataType="dtos:ProductDTO">

<Border CornerRadius="12" Background="#47BAEC">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="60"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Image Grid.Row="0" Grid.Column="0" Margin="15" Source="{Binding ImageSource}"/>

<TextBlock Grid.Row="0" Grid.Column="1" Text="{Binding Title}" TextAlignment="Center" VerticalAlignment="Center"/>

<Border Grid.Row="1" Grid.Column="0" CornerRadius="12" Background="#CFF1FF" Margin="10">

<TextBlock Foreground="Black" FontSize="12" Text="{Binding Price}" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Border>

<Border Grid.Row="1" Grid.Column="1" CornerRadius="12" Background="#CFF1FF" Margin="10">

<TextBlock Foreground="Black" FontSize="12" Text="{Binding Availability}" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Border>

<Border Grid.Row="1" Grid.Column="2" CornerRadius="12" Background="#80D9FC" Margin="10">

<Button Background="Transparent" VerticalAlignment="Center" HorizontalAlignment="Center">

<TextBlock Foreground="Black" FontSize="12" Text="Купить" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Button>

</Border>

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

Все данные отображаются правильно, но изображения отсутствуют, поэтому написан конвертер:

public static class ImagePathConverter

{

public static Bitmap LoadFromResource(Uri resourceUri)

{

return new Bitmap(AssetLoader.Open(resourceUri));

}

public static async Task<Bitmap?> LoadFromWeb(Uri url)

{

using var httpClient = new HttpClient();

try

{

var response = await httpClient.GetAsync(url);

response.EnsureSuccessStatusCode();

var data = await response.Content.ReadAsByteArrayAsync();

return new Bitmap(new MemoryStream(data));

}

catch (HttpRequestException ex)

{

Console.WriteLine($"An error occurred while downloading image '{url}' : {ex.Message}");

return null;

}

}

}

Он находит ресурс и предоставляет его для отображения.

**База данных**

База данных, которая используется в моём проекте – MsSQL.

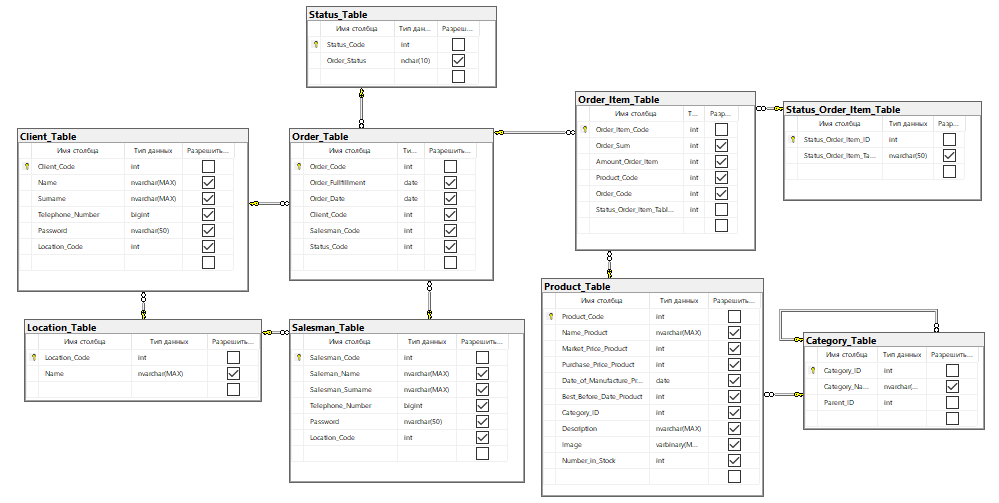


Рис.8 – База данных

**Написание сервера**

Для бд были написаны следующие модели:

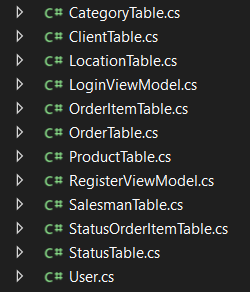


Рис.9 – Модели

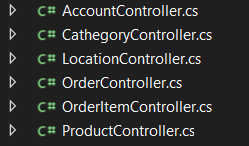


Рис.10 – Контроллеры

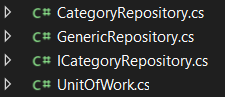


Рис.11 – Реализация репозитория

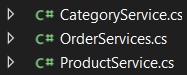


Рис.12 – Реализация сервисов