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 3**

**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

**Добавление UI Theme**

Добавлена UI Theme для Avalonia 11.0.5. Установлен с помощью NuGet и добавлен в App.xaml вместо стандартной FluentTheme.

<Application xmlns="https://github.com/avaloniaui"

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

x:Class="InternetShopMobileApp.App"

RequestedThemeVariant="Light">

<!-- "Default" ThemeVariant follows system theme variant. "Dark" or "Light" are other available options. -->

<Application.Styles>

<StyleInclude Source="avares://SukiUI/Theme/Index.xaml" />

</Application.Styles>

</Application>

**Реализация общения с сервером**

При загрузке станицы будет вызываться следующая функция:

public async void LoadProducts()

{

HttpClient client = new HttpClient();

try

{

var response = await client.GetAsync(URLHelper.APIURL + "/api/Product");

if (response.IsSuccessStatusCode)

{

var jsonResponse = await response.Content.ReadAsStringAsync();

var products = JsonConvert.DeserializeObject<List<ProductData>>(jsonResponse);

products.Reverse();

products.Take(10);

foreach (var product in products)

{

Products.Add(product);

}

}

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

finally

{

client.Dispose();

}

}

Она будет отвечать за то, чтобы выгружать данные из сервера, парсить их в необходимую структуру данных, а также обновлять переменную для последующего отображения.

DTO хранятся в отдельной папке на клиенте и представляют собой классы с полями, которые необходимы для отображения. Например:

public class ProductData

{

public int ProductCode { get; set; }

public string? NameProduct { get; set; }

public int? MarketPriceProduct { get; set; }

public int? PurchasePriceProduct { get; set; }

public DateTime? DateOfManufactureProduct { get; set; }

public int? BestBeforeDateProduct { get; set; }

public int? CategoryId { get; set; }

public string? Description { get; set; }

public byte[]? Image { get; set; }

public int? NumberInStock { get; set; }

public virtual CategoryData? Category { get; set; }

public virtual ICollection<OrderItemData> OrderItemDatas { get; } = new List<OrderItemData>();

}

public partial class ClientData

{

public int ClientCode { get; set; }

public string? Name { get; set; }

public string? Surname { get; set; }

…

Вернёмся к функции: Запрос к серверу она делает с помощью команды

var response = await client.GetAsync(URLHelper.APIURL + "/api/Product");

Таким образом она обращается по нужному URL и получает необходимые данные.

Для того, чтобы получить данные, необходимые нам для размещения на странице используется:

var jsonResponse = await response.Content.ReadAsStringAsync();

var products = JsonConvert.DeserializeObject<List<ProductData>>(jsonResponse);

Таким образом мы получаем данные, которые преобразовались из JSON объекта в лист с структурой из DTO. После того, как данные получены обновляется свойство Products.

foreach (var product in products)

{

Products.Add(product);

}

Данный массив используется для отображения во View.

public ObservableCollection<ProductData> Products { get; set; }

После того, как данные сохранены, они биндятся во view.

<ListBox ItemsSource="{Binding Products}"

Background="#80D9FC"

Margin="16,20,16,10"

Padding="0"

SelectedItem="{Binding SelectedProduct}">

<ListBox.ItemTemplate>

<DataTemplate DataType="dtos:ProductData">

<Border x:Name="ClickableBorder" Classes="Card Hoverable" 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 Image, Converter={StaticResource ByteArrayToImageConverter}}"/>

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

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

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

</Border>

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

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

<TextBlock Foreground="Black" Margin="-8,-8,0,0" FontSize="12" Text="Купить" VerticalAlignment="Center" HorizontalAlignment="Center"/>

</Button>

</Border>

</Grid>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

Таким образом отображаются данные, которые были заданы.

Некоторые страницы требуют того, чтобы им передавали какие либо данные. В частности, странице ProductContentView необходимы данные о выбранном товаре.

При выборе текущего элемента из товаров биндинтся свойство SelectedItems

SelectedItem="{Binding SelectedProduct}"

После того, как Item выбран, он меняется и во ViewModel:

private ProductData \_selectedProduct;

public ProductData SelectedProduct

{

get => \_selectedProduct;

set

{

this.RaiseAndSetIfChanged(ref \_selectedProduct, value);

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

}

}

Таким образом происходит переход на новую страницу и передачей текущего выбранного товара.

**Публикация сервер**

Для того, чтобы мобильное устройство могло обращаться к серверу, его необходимо было опубличить.

Для этого я использовал ngrok. Было установлено необходимое дополнение для Visual Studio 2022. Заданы необходимые настройки для расширения:

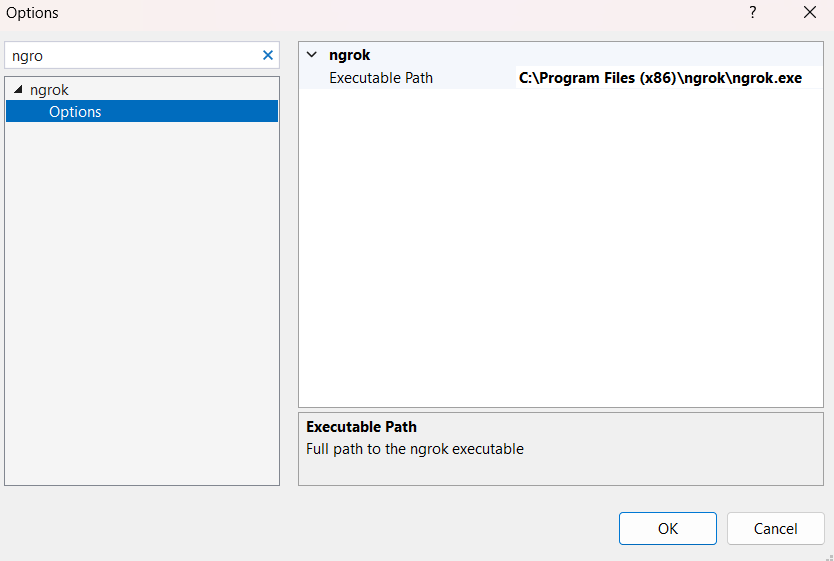


Рис. 1 – Настройки ngrok

После того, как сервер запущен, необходимо запустить работу расширения:

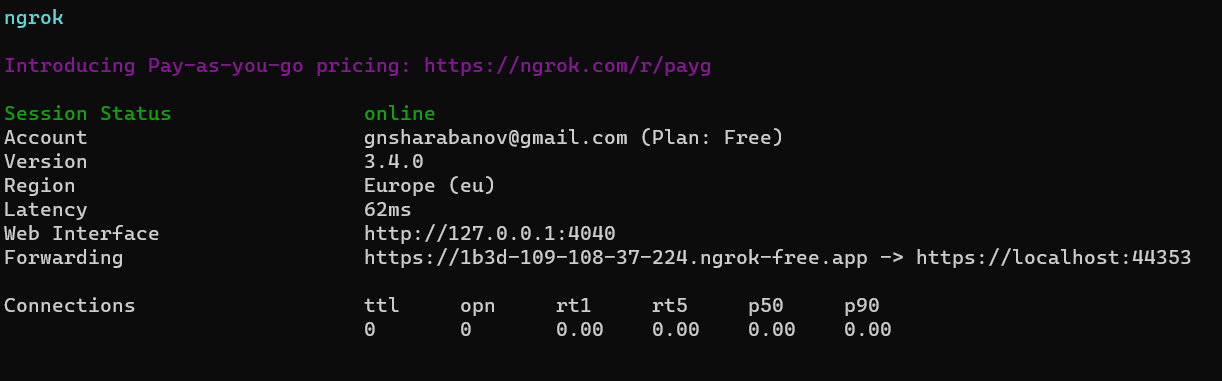


Рис. 2 – Запуск ngrok

Теперь к серверу можно обращаться с других устройств по URL, которое написано в поле Forwarding. Этот URL я и использую в программе:

public static class URLHelper

{

public static string APIURL = "https://3054-85-142-148-148.ngrok-free.app";

}

Таким образом программа обращается к серверу и успешно получает данные от него:

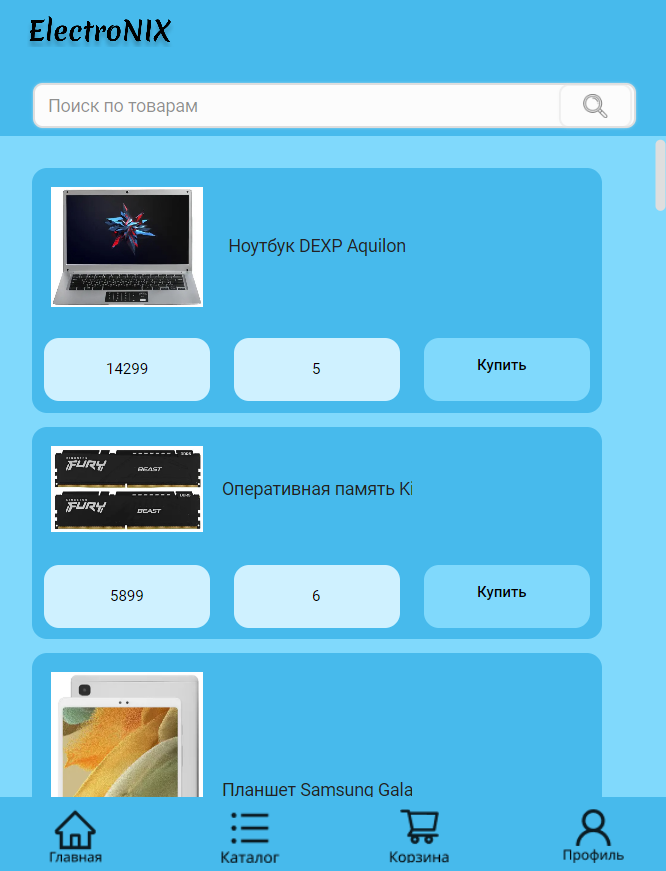


Рис. 3 – Окно каталога

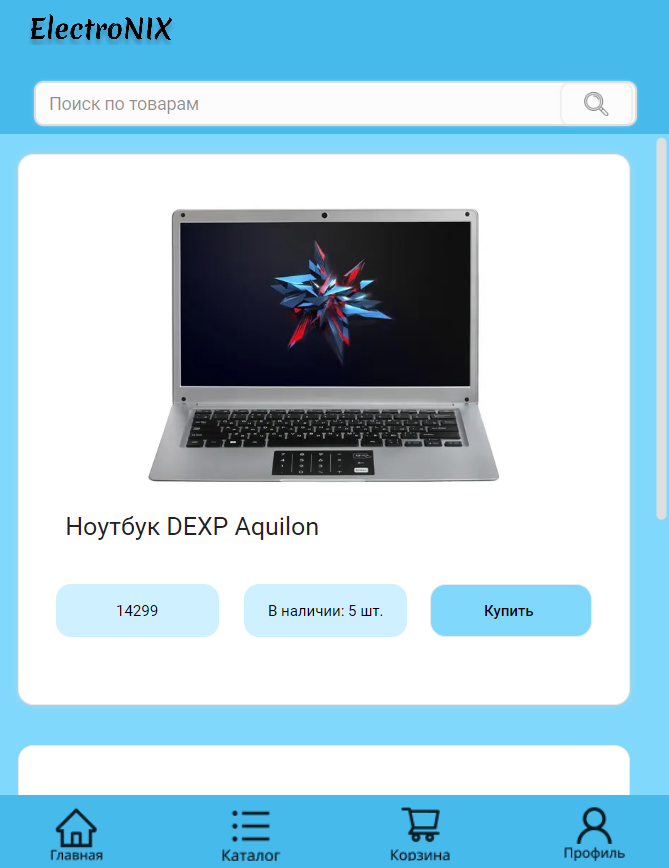


Рис. 4 – Окно товара