.NET MAUI로 아이스크림 SHOP 앱 만들기

I. 개요.

1. 시작하기.

1) blank app

2) .NET MAUI App

프로젝트 이름: IcecreamShop.APP

3) ASP.NET Core Web API

프로젝트 이름: IcecreamShop.API

4) Class Library

프로젝트 이름: IcecreamShop.Shared

IcecreamShop.APP

다음과 같이 폴더를 생성한다.

Components

Data

Models

ViewModels

Pages

2). Pages 폴더에

OnboardingPage.xaml

SigninPage.xaml

SignupPage.xaml

HomePage.xaml

CartPage.xaml

ProfilePage.xaml

MyOrdersPage.xaml

OrderDetailPage.xaml

DetailsPage.xaml

3) 아이콘들 구하기.

<https://boxicons.com/>

<https://www.flaticon.com/kr/>

이미지 리소스

|  |  |
| --- | --- |
| Cart |  |
| cross |  |
| Home |  |
| logout |  |
| minus |  |
| plus |  |
| right\_arrow |  |
| trash |  |
| user |  |
| lock |  |

**II. 기본 화면 구성**

1. Resources/Styles/Color.xaml에 색상 추가(이것은 styles.xml에서만 사용되는걸까?

|  |
| --- |
| <Color x:Key="Red900">#45080b</Color>  <Color x:Key="Red500">#a80e16</Color>  <Color x:Key="Pink900">#f1378f</Color>  <Color x:Key="Frame1">#d55584</Color>  <Color x:Key="DeepPink">#ee0fa8</Color> |

2. Resources/Styles/Styles.xaml

|  |
| --- |
| <Style TargetType="Shell" ApplyToDerivedTypes="True">  <Setter Property="Shell.BackgroundColor" Value="{AppThemeBinding *Light*={StaticResource DeepPink}, *Dark*={StaticResource OffBlack}}" />  <Setter Property="Shell.ForegroundColor" Value="{AppThemeBinding *Light*={StaticResource Black}, *Dark*={StaticResource SecondaryDarkText}}" />  <Setter Property="Shell.TitleColor" Value="{AppThemeBinding *Light*={StaticResource White}, *Dark*={StaticResource SecondaryDarkText}}" />  <Setter Property="Shell.DisabledColor" Value="{AppThemeBinding *Light*={StaticResource Gray200}, *Dark*={StaticResource Gray950}}" />  <Setter Property="Shell.UnselectedColor" Value="{AppThemeBinding *Light*={StaticResource Gray200}, *Dark*={StaticResource Gray200}}" />  <Setter Property="Shell.NavBarHasShadow" Value="False" />  <Setter Property="Shell.TabBarBackgroundColor" Value="{AppThemeBinding *Light*={StaticResource White}, *Dark*={StaticResource Black}}" />  <Setter Property="Shell.TabBarForegroundColor" Value="{AppThemeBinding *Light*={StaticResource Magenta}, *Dark*={StaticResource White}}" />  <Setter Property="Shell.TabBarTitleColor" Value="{AppThemeBinding *Light*={StaticResource Magenta}, *Dark*={StaticResource White}}" />  <Setter Property="Shell.TabBarUnselectedColor" Value="{AppThemeBinding *Light*={StaticResource Gray900}, *Dark*={StaticResource Gray200}}" />  </Style> |

3. AppShell.xaml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <Shell  x:Class="IcecreamShop.APP.AppShell"  xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:IcecreamShop.APP"  xmlns:pages="clr-namespace:IcecreamShop.APP.Pages"  Shell.FlyoutBehavior="Flyout"  Title="IcecreamShop.APP">  <ShellContent  Title="Home"  ContentTemplate="{DataTemplate pages:OnboardingPage}"  Shell.FlyoutBehavior="Disabled"  Shell.FlyoutItemIsVisible="False"/>  <TabBar FlyoutDisplayOptions="AsMultipleItems">  <ShellContent  Title="Home"  ContentTemplate="{DataTemplate pages:HomePage}"  Icon="home.png"  Route="HomePage"/>  <ShellContent  Title="Cart"  ContentTemplate="{DataTemplate pages:CartPage}"  Icon="cart.png"  Route="CartPage"/>  <ShellContent  Title="Profile"  ContentTemplate="{DataTemplate pages:ProfilePage}"  Icon="user.png"  Route="Profile"/>  </TabBar>  <MenuItem IconImageSource="logout.png" Text="Sign out" Clicked="SignoutMenuItem\_Clicked"/>    <Shell.ItemTemplate>  <DataTemplate>  <Grid ColumnDefinitions="Auto, \*,Auto"  RowDefinitions="Auto, Auto"  Padding="5" ColumnSpacing="15">  <Image Grid.Column="0" Source="{Binding *FlyoutIcon*}"  HeightRequest="24" WidthRequest="24"/>  <Label Grid.Column="1"  Text="{Binding *Title*}" TextColor="DeepPink"  FontSize="16"/>  <Image Grid.Column="2"  Source="right\_arrow.png"  WidthRequest="24">  </Image>  <BoxView Grid.Row="1"  Grid.ColumnSpan="3"  HeightRequest="1"/>  </Grid>  </DataTemplate>  </Shell.ItemTemplate>  <Shell.MenuItemTemplate>  <DataTemplate>  <Grid ColumnDefinitions="Auto, \*,Auto"  Padding="5" ColumnSpacing="15">  <Image Grid.Column="0" Source="{Binding *Icon*}"  HeightRequest="24" WidthRequest="24"/>  <Label Grid.Column="1"  Text="{Binding *Text*}" TextColor="DeepPink"  FontSize="16"/>  </Grid>  </DataTemplate>  </Shell.MenuItemTemplate>  <Shell.FlyoutHeader>  <Border BackgroundColor="DeepPink" Padding="10">  <VerticalStackLayout HorizontalOptions="Center" Spacing="10">  <Image Source="ice\_cream.png" HeightRequest="150"  WidthRequest="150" HorizontalOptions="Center"/>  <Label Text="MAUI Ice Cream"  HorizontalOptions="Center"  FontSize="18"  TextColor="White"  TextTransform="Uppercase"  FontAttributes="Bold"/>  </VerticalStackLayout>  </Border>  </Shell.FlyoutHeader>  <Shell.FlyoutFooter>  <Border BackgroundColor="DeepPink"  StrokeThickness="0"  Padding="10">  <Label Text="Developed by "  HorizontalOptions="Center"  FontSize="15"  TextColor="Wheat"/>  <Border.GestureRecognizers>  <TapGestureRecognizer Tapped="TapGestureRecognizer\_Tapped"/>  </Border.GestureRecognizers>  </Border>  </Shell.FlyoutFooter>    </Shell> |

4. AppShell.xaml.cs

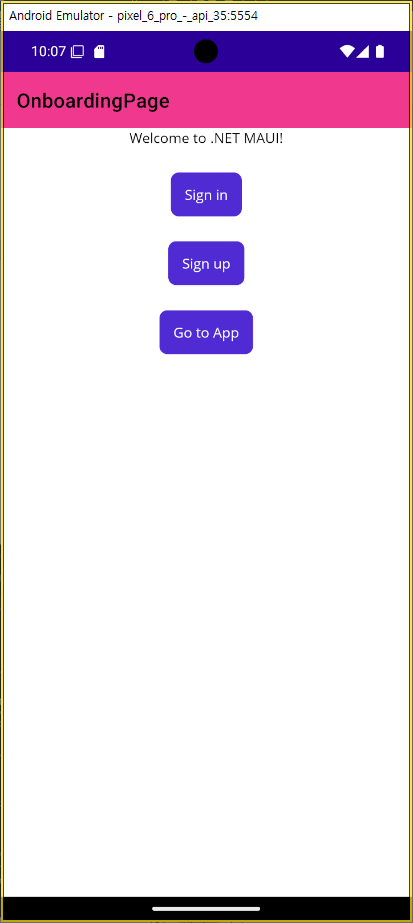
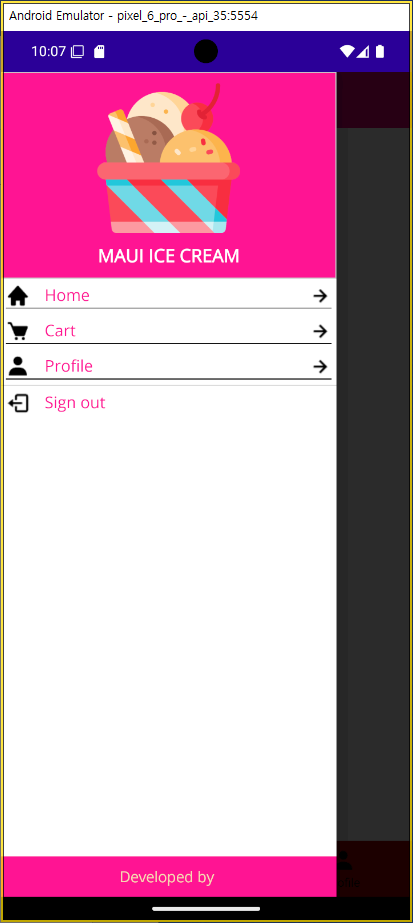
|  |
| --- |
| using IcecreamShop.APP.Pages;  namespace IcecreamShop.APP  {  public partial class AppShell : Shell  {  public AppShell()  {  InitializeComponent();  RegisterRoutes();  }  private readonly static Type[] \_routablePageTypes =  [  typeof(SigninPage),  typeof(SignupPage),  typeof(MyOrdersPage),  typeof(OrderDetailPage),  typeof(DetailsPage),  ];  private static void RegisterRoutes()  {  foreach(var pageType in \_routablePageTypes)  {  Routing.RegisterRoute(pageType.Name, pageType);  }  //Routing.RegisterRoute(nameof(SigninPage), typeof(SigninPage));  //Routing.RegisterRoute(nameof(SignupPage), typeof(SignupPage));  }  private async void FooterTapGestureRecognizer\_Tapped(object sender, TappedEventArgs e)  {  await Launcher.OpenAsync("https://myhome/");  }  private async void SignoutMenuItem\_Clicked(object sender, EventArgs e)  {  await Shell.Current.DisplayAlert("Alert", "Signout menu item clicked", "OK");  }  }  } |

5. OnboardingPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.OnboardingPage"  >  <VerticalStackLayout Spacing="25">  <Label  Text="Welcome to .NET MAUI!"  VerticalOptions="Center"  HorizontalOptions="Center" />  <Button Text="Sign in"  Clicked="Signin\_Clicked"  HorizontalOptions="Center"  VerticalOptions="Center"/>  <Button Text="Sign up"  Clicked="Signup\_Clicked"  HorizontalOptions="Center"  VerticalOptions="Center"/>  <Button Text="Go to App"  Clicked="Button\_Clicked"  HorizontalOptions="Center"  VerticalOptions="Center"/>    </VerticalStackLayout>  </ContentPage> |

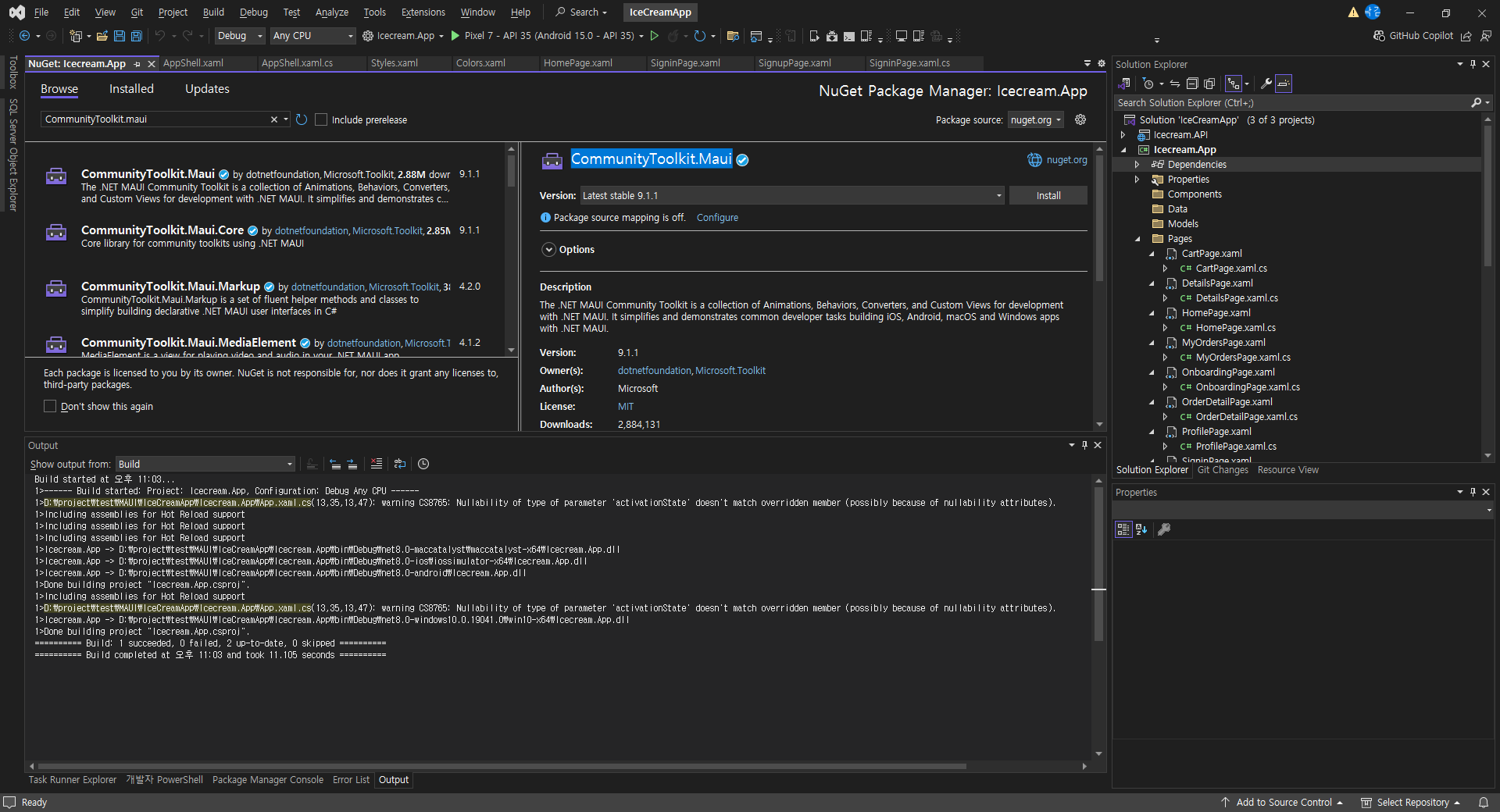
6. OnboardingPage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class OnboardingPage : ContentPage  {  public OnboardingPage()  {  InitializeComponent();  }  private async void Button\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync($"//{nameof(HomePage)}");  }  private async void Signin\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync(nameof(SigninPage));  }  private async void Signup\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync(nameof(SignupPage));  }  } |

7. 아이콘의 컬러를 변경하기 위해서 패키지 추가

1) Nuget으로 CommunityToolkit.Maui 추가

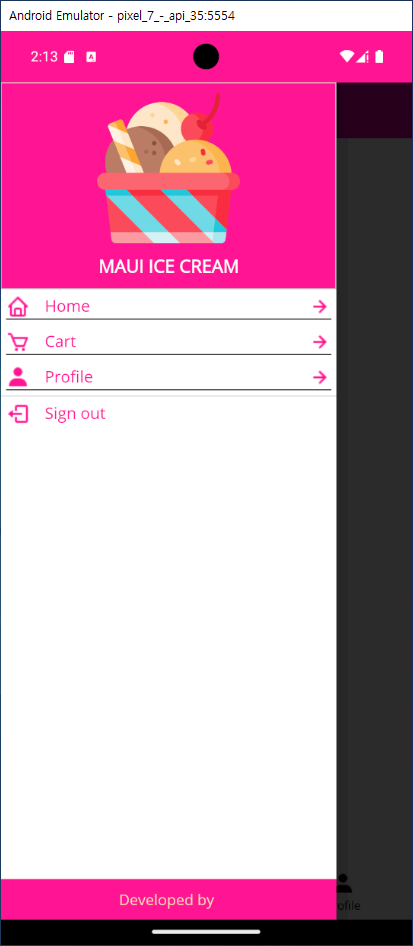


2) mauiProgram.cs

|  |
| --- |
| namespace IcecreamShop.APP  {  public static class MauiProgram  {  public static MauiApp CreateMauiApp()  {  var builder = MauiApp.CreateBuilder();  builder  .UseMauiApp<App>()  .UseMauiCommunityToolkit()  .ConfigureFonts(fonts =>  {  fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");  fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");  });  #if DEBUG  builder.Logging.AddDebug();  #endif  return builder.Build();  }  }  } |

3) AppShell.xaml 수정

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <Shell  x:Class="IcecreamShop.APP.AppShell"  xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:IcecreamShop.APP"  xmlns:pages="clr-namespace:IcecreamShop.APP.Pages"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  Shell.FlyoutBehavior="Flyout"  Title="IcecreamShop.APP">  <Shell.Behaviors>  <toolkit:StatusBarBehavior StatusBarColor="DeepPink"/>  </Shell.Behaviors>  <ShellContent  Title="Home"  ContentTemplate="{DataTemplate pages:OnboardingPage}"  Shell.FlyoutBehavior="Disabled"  Shell.FlyoutItemIsVisible="False"/>  <TabBar FlyoutDisplayOptions="AsMultipleItems">  <ShellContent  Title="Home"  ContentTemplate="{DataTemplate pages:HomePage}"  Icon="home.png"  Route="HomePage"/>  <ShellContent  Title="Cart"  ContentTemplate="{DataTemplate pages:CartPage}"  Icon="cart.png"  Route="CartPage"/>  <ShellContent  Title="Profile"  ContentTemplate="{DataTemplate pages:ProfilePage}"  Icon="user.png"  Route="Profile"/>  </TabBar>  <MenuItem IconImageSource="logout.png" Text="Sign out" Clicked="SignoutMenuItem\_Clicked"/>    <Shell.ItemTemplate>  <DataTemplate>  <Grid ColumnDefinitions="Auto, \*,Auto"  RowDefinitions="Auto, Auto"  Padding="5" ColumnSpacing="15">  <Image Grid.Column="0" Source="{Binding *FlyoutIcon*}"  HeightRequest="24" WidthRequest="24">  <Image.Behaviors>  <toolkit:IconTintColorBehavior TintColor="DeepPink"/>  </Image.Behaviors>  </Image>  <Label Grid.Column="1"  Text="{Binding *Title*}" TextColor="DeepPink"  FontSize="16"/>  <Image Grid.Column="2"  Source="right\_arrow.png"  WidthRequest="24">  <Image.Behaviors>  <toolkit:IconTintColorBehavior TintColor="DeepPink"/>  </Image.Behaviors>  </Image>  <BoxView Grid.Row="1"  Grid.ColumnSpan="3"  HeightRequest="1"/>  </Grid>  </DataTemplate>  </Shell.ItemTemplate>  <Shell.MenuItemTemplate>  <DataTemplate>  <Grid ColumnDefinitions="Auto, \*,Auto"  Padding="5" ColumnSpacing="15">  <Image Grid.Column="0" Source="{Binding *Icon*}"  HeightRequest="24" WidthRequest="24">  <Image.Behaviors>  <toolkit:IconTintColorBehavior TintColor="DeepPink"/>  </Image.Behaviors>  </Image>  <Label Grid.Column="1"  Text="{Binding *Text*}" TextColor="DeepPink"  FontSize="16"/>  </Grid>  </DataTemplate>  </Shell.MenuItemTemplate>  <Shell.FlyoutHeader>  <Border BackgroundColor="DeepPink" Padding="10">  <VerticalStackLayout HorizontalOptions="Center" Spacing="10">  <Image Source="ice\_cream.png" HeightRequest="150"  WidthRequest="150" HorizontalOptions="Center"/>  <Label Text="MAUI Ice Cream"  HorizontalOptions="Center"  FontSize="18"  TextColor="White"  TextTransform="Uppercase"  FontAttributes="Bold"/>  </VerticalStackLayout>  </Border>  </Shell.FlyoutHeader>  <Shell.FlyoutFooter>  <Border BackgroundColor="DeepPink"  StrokeThickness="0"  Padding="10">  <Label Text="Developed by "  HorizontalOptions="Center"  FontSize="15"  TextColor="Wheat"/>  <Border.GestureRecognizers>  <TapGestureRecognizer Tapped="FooterTapGestureRecognizer\_Tapped"/>  </Border.GestureRecognizers>  </Border>  </Shell.FlyoutFooter>    </Shell> |



8. Flayout 메뉴 이벤트 연결

1) OnboardingPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.OnboardingPage"  Shell.NavBarIsVisible="False">  <Grid RowDefinitions="\*, Auto"  Padding="0,0,0,40">  <VerticalStackLayout Grid.Row="0" Spacing="20"  HorizontalOptions="Center"  VerticalOptions="Center">  <Image Source="ice\_cream.png"  HeightRequest="250"  WidthRequest="250"  Aspect="AspectFit"/>  <Label Text="Welcome"  FontSize="20"  HorizontalOptions="Center"  FontAttributes="Bold"/>  <Label Text="MAUI ICE CREAMS"  FontSize="25"  HorizontalOptions="Center"/>  <Label Text="Home of delicious Ice Creams"  FontSize="18"  HorizontalOptions="Center"/>  </VerticalStackLayout>  <VerticalStackLayout Grid.Row="1"  Spacing="20"  HorizontalOptions="Center">  <Button Text="Signup" Clicked="Signup\_Clicked"/>  <Button Text="Signin" Clicked="Signin\_Clicked"/>  </VerticalStackLayout>  </Grid>  </ContentPage> |

2) OnboardingPage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class OnboardingPage : ContentPage  {  public OnboardingPage()  {  InitializeComponent();  }  private async void Button\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync($"//{nameof(HomePage)}");  }  private async void Signin\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync(nameof(SigninPage));  }  private async void Signup\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync(nameof(SignupPage));  }  } |

3) SigninPage.xaml( app프로그램 흐름상 임시적으로 사용)

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.SigninPage"  Title="Signin"  Shell.FlyoutBehavior="Disabled">  <VerticalStackLayout VerticalOptions="Center">  <Label  Text="Welcome to .NET MAUI!"  VerticalOptions="Center"  HorizontalOptions="Center" />  <Button Text="Go to App" Clicked="Button\_Clicked"/>  </VerticalStackLayout>  </ContentPage> |

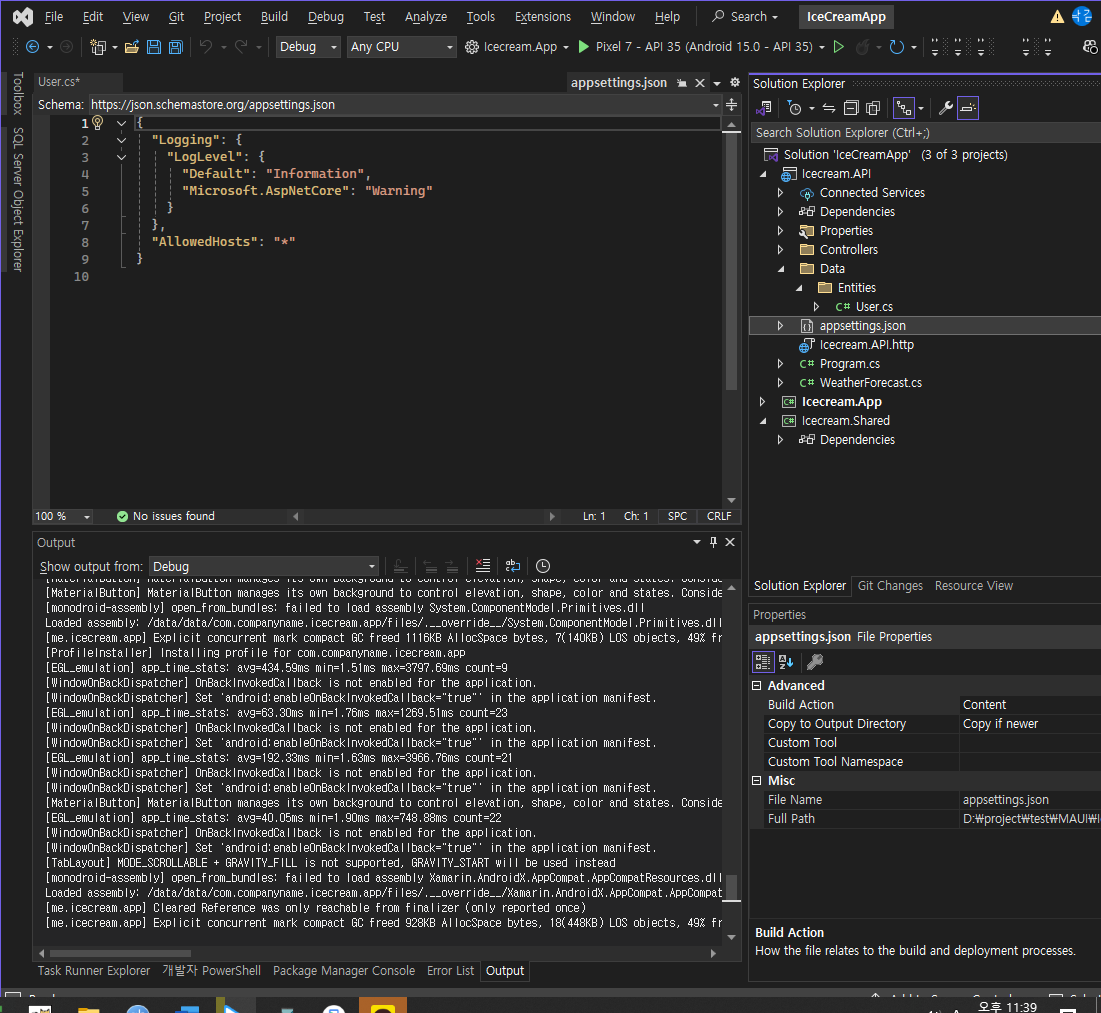
4) SigninPage.xaml.cs( app프로그램 흐름상 임시적으로 사용)

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class SigninPage : ContentPage  {  public SigninPage()  {  InitializeComponent();  }  private async void Button\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync($"//{nameof(HomePage)}");  }  } |

**III. 로그인/로그아웃 처리를 위한 API 서버 만들기.**

**1. 프로젝트 생성**

Back-end을 위한 IcecreamShop.API 프로젝트로 이동하여 Data폴더를 생성하고 그 하부에 Entities폴더를 생성한다. 그리고 User.cs 클래스를 생성한다.



**2. 엔티티 클래스 생성**

1) Data/Entities/User.cs

|  |
| --- |
| namespace IcecreamShop.API.Data.Entities;  public class User  {  [Key]  public Guid Id { get; set; }  [Required, MaxLength(30)]  public string Name { get; set; }  [Required, MaxLength(100)]  public string Email { get; set; }  [Required, MaxLength(150)]  public string Address { get; set; }  [Required, MaxLength(20)]  public string Salt { get; set; }  [Required, MaxLength(180)]  public string Hash { get; set; }  } |

2) Data/Entities/Icecream.cs

|  |
| --- |
| namespace IcecreamShop.API.Data.Entities;  public class Icecream  {  [Key]  public int Id { get; set; }  [Required, MaxLength(100)]  public string Name { get; set; }  [Range(0.1, double.MaxValue)]  public double Price { get; set; }  [Required, MaxLength(180)]  public string Image { get; set; }  public virtual ICollection<IcecreamOption> Options { get; set; }  } |

3) Data/Entities/IcecreamOption.cs

|  |
| --- |
| namespace IcecreamShop.API.Data.Entities;  public class IcecreamOption  {  public int IcecreamId { get; set; }  [Required, MaxLength(50)]  public string Flavor { get; set; }  [Required, MaxLength(50)]  public string Topping { get; set; }  public virtual Icecream Icecream { get; set; }  } |

4) Data/Entities/Order.cs

|  |
| --- |
| namespace IcecreamShop.API.Data.Entities;  public class Order  {  [Key]  public long Id { get; set; }  public DateTime OrderAt { get; set; } = DateTime.Now;  public Guid CustomerId { get; set; }  [Required, MaxLength(30)]  public string CustomerName { get; set; }  [Required, MaxLength(100)]  public string CustomerEmail { get; set; }  [Required, MaxLength(150)]  public string CustomerAddress { get; set; }  [Range(0.1, double.MaxValue)]  public double TotalPrice { get; set; }  public ICollection<OrderItem> Items { get; set; }  } |

5) Data/Entities/OrderItem.cs

|  |
| --- |
| namespace IcecreamShop.API.Data.Entities;  public class OrderItem  {  [Key]  public long Id { get; set; }  public long OrderId { get; set; }  public int IcecreamId { get; set; }  [Required, MaxLength(50)]  public string Name { get; set; }  [Range(0.1, double.MaxValue)]  public double Price { get; set; }  [Range(1, 100)]  public int Quantity { get; set; }  [Required, MaxLength(50)]  public string Flavor { get; set; }  [Required, MaxLength(50)]  public string Topping { get; set; }  [Range(0.1, double.MaxValue)]  public double TotalPrice { get; set; }  public virtual Order Order { get; set; }  } |

6) 데이터베이스를 위한 패키지 추가.

|  |
| --- |
| Microsoft.EntityFrameworkCore.SqlServer  Microsoft.EntityFrameworkCore.Tools |

7) dbcontext작성

참고: 원래 이미지 경로는 <https://raw.githubusercontent.com/Abhayprince/Images-Icons/main/Icecreams/small/> 이다.

Data/DataContext.cs

|  |
| --- |
| namespace IcecreamShop.API.Data;  public class DataContext : DbContext  {  public DataContext(DbContextOptions<DataContext> options) : base(options)  {  }  public DbSet<User> Users { get; set; }  public DbSet<Icecream> Icecreams { get; set; }  public DbSet<IcecreamOption> IcecreamOptions { get; set; }  public DbSet<Order> Orders { get; set; }  public DbSet<OrderItem> OrderItems { get; set; }  protected override void OnModelCreating(ModelBuilder modelBuilder)  {  //base.OnModelCreating(modelBuilder);  modelBuilder.Entity<IcecreamOption>()  .HasKey(io => new { io.IcecreamId, io.Flavor, io.Topping });  AddSeedData(modelBuilder);  }  private static void AddSeedData(ModelBuilder modelBuilder)  {  Icecream[] icecreams =  [  new Icecream{Id=1, Name = "Vanilla Delight", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_0.jpg", Price = 6.25},  new Icecream{Id=2, Name = "ChocoBerry Bliss", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_1.jpg", Price = 7.5},  new Icecream{Id=3, Name = "Minty Cookie Crunch", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_2.jpg", Price = 8.75},  new Icecream{Id=4, Name = "Strawberry Dream", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_3.jpg", Price = 6.95},  new Icecream{Id=5, Name = "Cookie Dough Extravaganza", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_4.jpg", Price = 9.2},  new Icecream{Id=6, Name = "Caramel Swirl Symphony", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_5.jpg", Price = 7.75},  new Icecream{Id=7, Name = "Peanut Butter Paradise", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_6.jpg", Price = 8.5},  new Icecream{Id=8, Name = "Mango Tango Tango", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_7.jpg", Price = 9.8},  new Icecream{Id=9, Name = "Hazelnut Heaven", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_8.jpg", Price = 8.95},  new Icecream{Id=10, Name = "Blueberry Bliss Bonanza", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_9.jpg", Price = 7.2},  new Icecream{Id=11, Name = "Toffee Twist Temptation", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_10.jpg", Price = 7.95},  new Icecream{Id=12, Name = "Rocky Road Revelry", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_11.jpg", Price = 9.5},  new Icecream{Id=13, Name = "Passionfruit Paradise", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_12.jpg", Price = 8.75},  new Icecream{Id=14, Name = "Cotton Candy Carnival", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_13.jpg", Price = 7.2},  new Icecream{Id=15, Name = "Lemon Sorbet Serenity", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_14.jpg", Price = 6.9},  new Icecream{Id=16, Name = "Maple Pecan Pleasure", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_15.jpg", Price = 8.25},  new Icecream{Id=17, Name = "Raspberry Ripple Rhapsody", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_16.jpg", Price = 7.45},  new Icecream{Id=18, Name = "Almond Joyful Jubilee", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_17.jpg", Price = 9.95},  new Icecream{Id=19, Name = "Blue Lagoon Blists", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_18.jpg", Price = 8.5},  new Icecream{Id=20, Name = "Coconut Caramel Carnival", Image = "https://raw.githubusercontent.com/KimNakHyun2/Image-Icons/main/icecream/main/ic\_19.jpg", Price = 7.8},  ];  IcecreamOption[] icecreamOptions = [  new IcecreamOption { IcecreamId = 1, Flavor = "Vanilla", Topping = "Default"},  new IcecreamOption { IcecreamId = 1, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 1, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 2, Flavor = "Vanilla", Topping = "Default"},  new IcecreamOption { IcecreamId = 2, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 2, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 3, Flavor = "Mint Chocolate Chip", Topping = "Default"},  new IcecreamOption { IcecreamId = 3, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 3, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 4, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 4, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 4, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 5, Flavor = "Cooke Dough", Topping = "Default"},  new IcecreamOption { IcecreamId = 5, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 5, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 6, Flavor = "Vanilla", Topping = "Default"},  new IcecreamOption { IcecreamId = 6, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 6, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 7, Flavor = "Chocolate", Topping = "Default"},  new IcecreamOption { IcecreamId = 7, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 7, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 8, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 8, Flavor = "Cooke Dough", Topping = "Default"},  new IcecreamOption { IcecreamId = 8, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 9, Flavor = "Mint Chocolate Chip", Topping = "Default"},  new IcecreamOption { IcecreamId = 9, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 9, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 10, Flavor = "Chocolate", Topping = "Default"},  new IcecreamOption { IcecreamId = 10, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 10, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 11, Flavor = "Vanilla", Topping = "Default"},  new IcecreamOption { IcecreamId = 11, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 11, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 12, Flavor = "Chocolate", Topping = "Default"},  new IcecreamOption { IcecreamId = 12, Flavor = "Mint Chocolate Chip", Topping = "Default"},  new IcecreamOption { IcecreamId = 12, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 13, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 13, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 13, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 14, Flavor = "Cooke Dough", Topping = "Default"},  new IcecreamOption { IcecreamId = 14, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 14, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 15, Flavor = "Vanilla", Topping = "Default"},  new IcecreamOption { IcecreamId = 15, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 15, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 16, Flavor = "Chocolate", Topping = "Default"},  new IcecreamOption { IcecreamId = 16, Flavor = "Mint Chocolate Chip", Topping = "Default"},  new IcecreamOption { IcecreamId = 16, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 17, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 17, Flavor = "Cooke Dough", Topping = "Default"},  new IcecreamOption { IcecreamId = 17, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 18, Flavor = "Vanilla", Topping = "Default"},  new IcecreamOption { IcecreamId = 18, Flavor = "Default", Topping = "Sprinkles"},  new IcecreamOption { IcecreamId = 18, Flavor = "Default", Topping = "Cherries"},  new IcecreamOption { IcecreamId = 19, Flavor = "Chocolate", Topping = "Default"},  new IcecreamOption { IcecreamId = 19, Flavor = "Strawberry", Topping = "Default"},  new IcecreamOption { IcecreamId = 19, Flavor = "Default", Topping = "Whipped Cream"},  new IcecreamOption { IcecreamId = 20, Flavor = "Mint Chocolate Chip", Topping = "Default"},  new IcecreamOption { IcecreamId = 20, Flavor = "Default", Topping = "Chocolate Sauce"},  new IcecreamOption { IcecreamId = 20, Flavor = "Default", Topping = "Sprinkles"},  ];  modelBuilder.Entity<Icecream>()  .HasData(icecreams);  modelBuilder.Entity<IcecreamOption>()  .HasData(icecreamOptions);  }  } |

**3. MS-SQL 에서 데이터베이스 생성.**

데이터베이스 이름: IcecreamYT

1) appsettings.Development.json

|  |
| --- |
| {  "Logging": {  "LogLevel": {  "Default": "Information",  "Microsoft.AspNetCore": "Warning"  }  },  "ConnectionStrings": {  "IcecreamDB": "Data Source=127.0.0.1;Initial Catalog=IcecreamYT;Persist Security Info=True;User ID=donkim;Password=1234;MultipleActiveResultSets=True;Encrypt=false;TrustServerCertificate=true;Connect Timeout=60"  }  } |

2) Program.cs 에 추가

|  |
| --- |
| builder.Services.AddDbContext<DataContext>(options =>  {  options.UseSqlServer(builder.Configuration.GetConnectionString("IcecreamDB") ?? throw new InvalidOperationException("Database connection string 'MyDB' is not found"));  }); |

3) Package Manager Concole에 다음과 같이 입력

Add-Migration Initial -o Data/Migrations

Update-Database

**4. JWT**

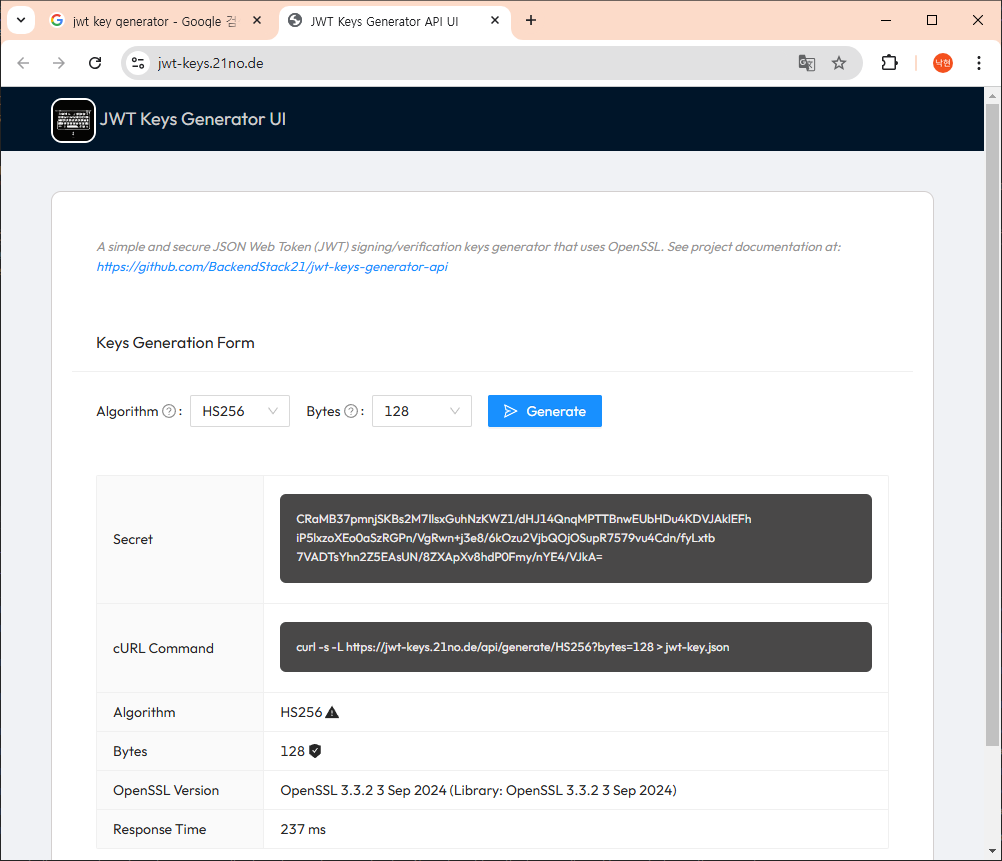
Microsoft.IdentityModel.Tokens

System.IdentityModel.Tokens.Jwt

Microsoft.AspNetCore.Authentication.JwtBearer

1) jwt 키값 생성.

<https://jwt-keys.21no.de/> 를 통해서 JWT secret key 값 생성.



2) appsettings.Development.json 에 설정

|  |
| --- |
| "Jwt": {  "SecretKey": "CRaMB37pmnjSKBs2M7IlsxGuhNzKWZ1/dHJ14QnqMPTTBnwEUbHDu4KDVJAklEFhiP5lxzoXEo0aSzRGPn/VgRwn+j3e8/6kOzu2VjbQOjOSupR7579vu4Cdn/fyLxtb7VADTsYhn2Z5EAsUN/8ZXApXv8hdP0Fmy/nYE4/VJkA=",  "Issuer": "https://localhost:7020",  "ExpireInMinute": 480  } |

3) 토큰 서비스를 위한 클래스 생성.

Services/TokenService.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class TokenService  {  private readonly IConfiguration \_configuration;  public TokenService(IConfiguration configuration)  {  this.\_configuration = configuration;  }  public static TokenValidationParameters GetTokenValidationParameters(IConfiguration configuration) =>  new TokenValidationParameters()  {  ValidateAudience = false,  ValidateIssuer = true,  ValidateLifetime = true,  ValidateIssuerSigningKey = true,  ValidIssuer = configuration["Jwt:Issuer"],  IssuerSigningKey = GetSecurityKey(configuration)  };    public string GenerateJwt(Guid userId, string userName, string email, string address)  {  //var secretKey = \_configuration["Jwt:SecretKey"]; //Guid.NewGuid().ToString();  //var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey!));  var securityKey = GetSecurityKey(\_configuration);  var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256 );  Claim[] claims = [  new Claim(ClaimTypes.NameIdentifier, userId.ToString()),  new Claim(ClaimTypes.Name, userName),  new Claim(ClaimTypes.Email, email),  new Claim(ClaimTypes.StreetAddress, address),  ];  var issuer = \_configuration["Jwt:Issuer"];  var expireInMinutes = Convert.ToInt32( \_configuration["Jwt:ExpireInMinute"]);  var token = new JwtSecurityToken(  issuer: issuer,  audience: "\*",  claims: claims,  expires: DateTime.Now.AddMinutes(expireInMinutes),  signingCredentials: credentials  );  var jwt = new JwtSecurityTokenHandler().WriteToken(token);  return jwt;  }  private static SymmetricSecurityKey GetSecurityKey(IConfiguration configuration)  {  var secretKey = configuration["Jwt:SecretKey"]; //Guid.NewGuid().ToString();  var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey!));  return securityKey;  }  } |

4) 토큰 서비스 등록하기.

Program.cs

|  |
| --- |
| using IcecreamShop.API.Data;  using IcecreamShop.API.Services;  using Microsoft.AspNetCore.Authentication.JwtBearer;  using Microsoft.EntityFrameworkCore;  using System;  var builder = WebApplication.CreateBuilder(args);  // Add services to the container.  builder.Services.AddControllers();  // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen();  builder.Services.AddDbContext<DataContext>(options =>  {  options.UseSqlServer(builder.Configuration.GetConnectionString("IcecreamDB") ?? throw new InvalidOperationException("Database connection string 'MyDB' is not found"));  });  builder.Services.AddAuthentication(options =>  {  options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;  options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;  })  .AddJwtBearer(jwtOptions =>  {  jwtOptions.TokenValidationParameters = TokenService.GetTokenValidationParameters(builder.Configuration);  });  ;  builder.Services.AddAuthorization();  builder.Services.AddTransient<TokenService>();  var app = builder.Build();  #if DEBUG  MigrateDatabase(app.Services);  #endif  // Configure the HTTP request pipeline.  if (app.Environment.IsDevelopment())  {  app.UseSwagger();  app.UseSwaggerUI();  }  app.UseHttpsRedirection();  app.UseAuthentication();  app.UseAuthorization();  app.MapControllers();  app.Run();  #if DEBUG  static void MigrateDatabase(IServiceProvider sp)  {  var scope = sp.CreateScope();  var dataContext = scope.ServiceProvider.GetRequiredService<DataContext>();  if(dataContext.Database.GetPendingMigrations().Any())  {  dataContext.Database.Migrate();  }  }  #endif |

5) 패스워드 관리

Service/ PasswordService.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class PasswordService  {  private const int SaltSize = 10;  public (string salt, string hashedPassword) GenerateSaltAndHash(string plainPassword)  {  if (string.IsNullOrWhiteSpace(plainPassword))  throw new ArgumentNullException(nameof(plainPassword));  var buffer = RandomNumberGenerator.GetBytes(10);  var salt = Convert.ToBase64String(buffer);  var hashedPassword = GenerateHashedPassword(plainPassword, salt);  return (salt, hashedPassword);  }  public bool AreEqual(string plainPassword, string salt, string hashedPassword)  {  var newHashedPassword = GenerateHashedPassword(plainPassword, salt);  return newHashedPassword == hashedPassword;  }  private string GenerateHashedPassword(string plainPassword, string salt)  {  var bytes = Encoding.UTF8.GetBytes(plainPassword + salt);  var hash = SHA256.HashData(bytes);  return Convert.ToBase64String(hash);  }  } |

5) 패스워드 서비스를 위한 등록

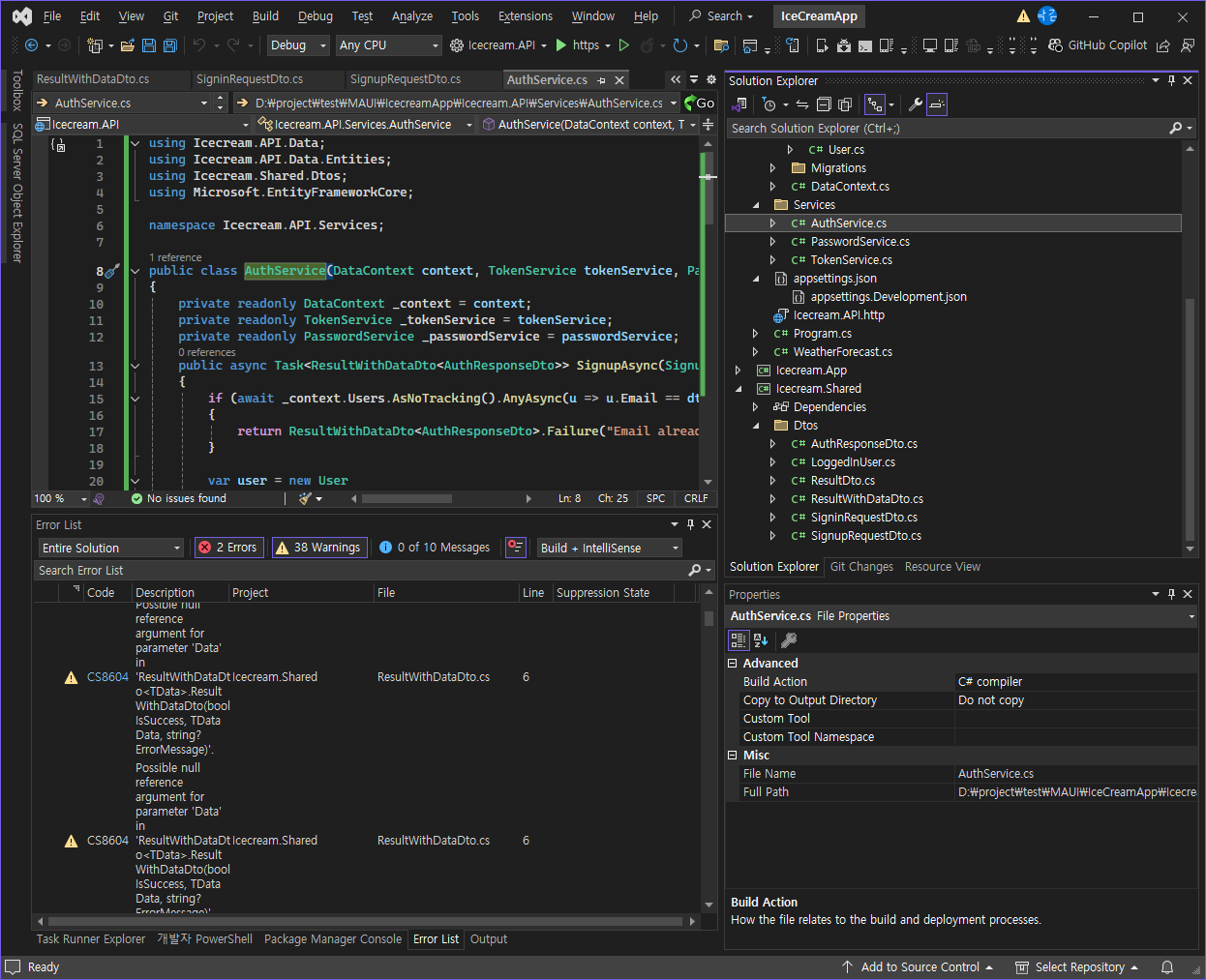
Program.cs에 다음을 추가

|  |
| --- |
| builder.Services.AddTransient<TokenService>()  .AddTransient<PasswordService>(); |

**5. 유저인증을 위한 APP과 API 서버간 통신**

1) AuthService를 위한 프로젝트 IcecreamShop.Shared에 클래스 추가.

프로젝트 IcecreamShop.Shared에 다음과 같이 추가한다.



Dtos/AuthResponseDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record AuthResponseDto**(**LoggedInUser User, string Token**)**; |

Dtos/LoggedInUser.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record LoggedInUser(Guid Id, string Name, string Email, string Address); |

Dtos/ResultDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record ResultDto(bool IsSuccess, string? ErrorMessage)  {  public static ResultDto Success() => new(true, null);  public static ResultDto Failure(string? errorMessage) => new(false, errorMessage);  } |

Dtos/ResultWithDataDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record ResultWithDataDto<TData>(bool IsSuccess, TData Data, string? ErrorMessage)  {  public static ResultWithDataDto<TData> Success(TData data) => new(true, data, null);  public static ResultWithDataDto<TData> Failure(string? errorMessage) => new(false, default, errorMessage);  } |

Dtos/SigninRequestDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record SigninRequestDto(string Email, string Password); |

SignupRequestDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record SignupRequestDto(string Name, string Email, string Password, string Address); |

2) AuthService

먼저 TokenService.cs 에서 GenerateJwt 함수 변경. 여러 개의 인자값을 받는 것에서부터 LoggedInUser로부 데이터를 받는것으로 변경한다.

|  |
| --- |
| public string GenerateJwt(LoggedInUser user)  {  //var secretKey = \_configuration["Jwt:SecretKey"]; //Guid.NewGuid().ToString();  //var securityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey!));  var securityKey = GetSecurityKey(\_configuration);  var credentials = new SigningCredentials(securityKey, SecurityAlgorithms.HmacSha256 );  Claim[] claims = [  new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),  new Claim(ClaimTypes.Name, user.Name),  new Claim(ClaimTypes.Email, user.Email),  new Claim(ClaimTypes.StreetAddress, user.Address),  ];  var issuer = \_configuration["Jwt:Issuer"];  var expireInMinutes = Convert.ToInt32( \_configuration["Jwt:ExpireInMinute"]);  var token = new JwtSecurityToken(  issuer: issuer,  audience: "\*",  claims: claims,  expires: DateTime.Now.AddMinutes(expireInMinutes),  signingCredentials: credentials  );  var jwt = new JwtSecurityTokenHandler().WriteToken(token);  return jwt;  } |

3) Services/AuthService.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class AuthService(DataContext context, TokenService tokenService, PasswordService passwordService)  {  private readonly DataContext \_context = context;  private readonly TokenService \_tokenService = tokenService;  private readonly PasswordService \_passwordService = passwordService;  public async Task<ResultWithDataDto<AuthResponseDto>> SignupAsync(SignupRequestDto dto)  {  if (await \_context.Users.AsNoTracking().AnyAsync(u => u.Email == dto.Email))  {  return ResultWithDataDto<AuthResponseDto>.Failure("Email already exists");  }  var user = new User  {  Email = dto.Email,  Address = dto.Address,  Name = dto.Name,  };  (user.Salt, user.Hash) = \_passwordService.GenerateSaltAndHash(dto.Password);  try  {  await \_context.Users.AddAsync(user);  await \_context.SaveChangesAsync();  return GenerateAuthResponse(user);  }  catch (Exception ex)  {  return ResultWithDataDto<AuthResponseDto>.Failure(ex.Message);  }  }  public async Task<ResultWithDataDto<AuthResponseDto>> SigninAsync(SigninRequestDto dto)  {  var dbUser = await \_context.Users.AsNoTracking().FirstOrDefaultAsync(u=>u.Email == dto.Email);  if (dbUser is null)  return ResultWithDataDto<AuthResponseDto>.Failure("User does not exist");  if(!\_passwordService.AreEqual(dto.Password, dbUser.Salt, dbUser.Hash))  return ResultWithDataDto<AuthResponseDto>.Failure("Incorrect password");  return GenerateAuthResponse(dbUser);  }  private ResultWithDataDto<AuthResponseDto> GenerateAuthResponse(User user)  {  var loggedInUser = new LoggedInUser(user.Id, user.Name, user.Email, user.Address);  var token = \_tokenService.GenerateJwt(loggedInUser);  var authResponse = new AuthResponseDto(loggedInUser, token);  return ResultWithDataDto<AuthResponseDto>.Success(authResponse);  }  } |

4) Program.cs 추가

|  |
| --- |
| builder.Services.AddTransient<TokenService>()  .AddTransient<PasswordService>()  .AddTransient< AuthService>(); |

5) API Endpoint 생성.

IcecreamShop.API 프로젝트에서 EndPoints를 추가한 다음에 EndPoints.cs 추가한다.

EndPoints/Endpoints.cs

|  |
| --- |
| namespace IcecreamShop.API.EndPoints;  public static class Endpoints  {  public static IEndpointRouteBuilder MapEndPoints(this IEndpointRouteBuilder app)  {  app.MapPost("/api/signup",  async (SignupRequestDto dto, AuthService authService) =>  TypedResults.Ok(await authService.SignupAsync(dto)));  app.MapPost("/api/signin",  async (SigninRequestDto dto, AuthService authService) =>  TypedResults.Ok(await authService.SigninAsync(dto)));  return app;  }  } |

Program.cs 추가

|  |
| --- |
| app.UseHttpsRedirection();  app.UseAuthentication();  app.UseAuthorization();  app.MapControllers();  app.MapEndPoints();  app.Run(); |

6. 앱에서 APP 호출하여 Signup과 Signin 구현하기.

1) 패키지를 추가한다.

IcecreamShop.APP(MAUI) 프로젝트에 다음의 2개의 패키지 추가.

CommunityToolkit.Mvvm

Refit.HttpClientFactory

2) IcecreamShop.APP(MAUI) 프로젝트에 Services/IAuthApi.cs를 추가한다.

|  |
| --- |
| namespace IcecreamShop.APP.Services;  public interface IAuthApi  {  [Post("/api/signup")]  Task<ResultWithDataDto<AuthResponseDto>> SignupAsync(SignupRequestDto dto);  [Post("/api/signin")]  Task<ResultWithDataDto<AuthResponseDto>> SigninAsync(SigninRequestDto dto);  } |

3) BaseViewModel 생성

ViewModels/BaseViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class BaseViewModel : ObservableObject  {  [ObservableProperty]  private bool \_isBusy;  protected async Task GotoAsync(string url, bool animate)  {  await Shell.Current.GoToAsync(url, animate);  }  protected async Task ShowErrorAlertAsync(string errorMessage)  {  await Shell.Current.DisplayAlert("Error", errorMessage, "Ok");  }  protected async Task ShowAlertAsync(string message)  {  await Shell.Current.DisplayAlert("Alert", message, "Ok");  }  } |

4) 유저 Sigin-in & Sigin-up ViewModel 생성.

ViewModels/AuthViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class AuthViewModel(IAuthApi authApi) : BaseViewModel  {  private readonly IAuthApi \_authApi = authApi;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_name;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignin)), NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_email;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignin)), NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_password;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_address;  public bool CanSignup => !string.IsNullOrEmpty(Name)  && !string.IsNullOrEmpty(Email)  && !string.IsNullOrEmpty(Password)  && !string.IsNullOrEmpty(Address);  public bool CanSignin => !string.IsNullOrEmpty(Password)  && !string.IsNullOrEmpty(Email);  [RelayCommand]  private async Task SignupAsync()  {  IsBusy = true;  try  {  var signupDto = new SignupRequestDto(Name, Email, Password, Address);  // Make Api Call  var result = await \_authApi.SignupAsync(signupDto);  if (result.IsSuccess)  {  // Navigate to HomePage  await ShowAlertAsync(result.Data.User.Name);  await GotoAsync($"//{nameof(HomePage)}", animate: true);  }  else  {  // Display Error Alert  await ShowErrorAlertAsync(result.ErrorMessage??"Unknown error in signing up");  }  }  catch (Exception ex)  {  await ShowErrorAlertAsync(ex.Message);  }  finally  {  IsBusy = false;  }  }  [RelayCommand]  private async Task SigninAsync()  {  IsBusy = true;  try  {  var signinDto = new SigninRequestDto(Email, Password);  // Make Api Call  var result = await \_authApi.SigninAsync(signinDto);  if (result.IsSuccess)  {  // Navigate to HomePage  await ShowAlertAsync(result.Data.User.Name);  await GotoAsync($"//{nameof(HomePage)}", animate: true);  }  else  {  // Display Error Alert  await ShowErrorAlertAsync(result.ErrorMessage ?? "Unknown error in signing in");  }  }  catch (Exception ex)  {  await ShowErrorAlertAsync(ex.Message);  }  finally  {  IsBusy = false;  }  }  } |

5) Refit 설정 및 서비스 등록

MauiProgram.cs

|  |
| --- |
| using CommunityToolkit.Maui;  using IcecreamShop.APP.Pages;  using IcecreamShop.APP.Services;  using IcecreamShop.APP.ViewModels;  using Microsoft.Extensions.Logging;  using Refit;  #if ANDROID  using System.Net.Security;  using Xamarin.Android.Net;  #elif IOS  using Security;  #endif  namespace IcecreamShop.APP  {  public static class MauiProgram  {  public static MauiApp CreateMauiApp()  {  var builder = MauiApp.CreateBuilder();  builder  .UseMauiApp<App>()  .UseMauiCommunityToolkit()  .ConfigureFonts(fonts =>  {  fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");  fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");  });  #if DEBUG  builder.Logging.AddDebug();  #endif  builder.Services.AddTransient<AuthViewModel>()  .AddTransient<SignupPage>()  .AddTransient<SigninPage>();  ConfigureRefit(builder.Services);  return builder.Build();  }  private static void ConfigureRefit(IServiceCollection services)  {  var refitSettings = new RefitSettings  {  HttpMessageHandlerFactory = () =>  {  #if ANDROID  return new AndroidMessageHandler  {  ServerCertificateCustomValidationCallback = (httpRequestMessage, certificate, chain, sslPolicyErrors) =>  {  return certificate?.Issuer == "CN=localhost" || sslPolicyErrors == SslPolicyErrors.None;  }  };  #elif IOS  return new NSUrlSessionHandler  {  TrustOverrideForUrl = (NSUrlSessionHandler sender, string url, SecTrust trust) =>  url.StartsWith("https://localhost")  };  #endif  return null;  }  };  services.AddRefitClient<IAuthApi>(refitSettings)  .ConfigureHttpClient(httpClient=>  {  var baseUrl = DeviceInfo.Platform == DevicePlatform.Android  ?"https://10.0.2.2:7282": "https://localhost:7282";  httpClient.BaseAddress = new Uri(baseUrl);  });  }  }  } |

Pages/Signin.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class SigninPage : ContentPage  {  public SigninPage(AuthViewModel authViewModel)  {  InitializeComponent();  BindingContext = authViewModel;  }  private async void Button\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync($"//{nameof(HomePage)}");  }  } |

Pages/Signup.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class SignupPage : ContentPage  {  public SignupPage(AuthViewModel authViewModel)  {  InitializeComponent();  BindingContext = authViewModel;  }  } |

6) Signup 페이지 디자인

App.xaml

|  |
| --- |
| <?xml version = "1.0" encoding = "UTF-8" ?>  <Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:IcecreamShop.APP"  x:Class="IcecreamShop.APP.App">  <Application.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="Resources/Styles/Colors.xaml" />  <ResourceDictionary Source="Resources/Styles/Styles.xaml" />  </ResourceDictionary.MergedDictionaries>  <Style TargetType="Button">  <Setter Property="BackgroundColor" Value="DeepPink"/>  <Setter Property="TextColor" Value="White"/>  <Setter Property="FontSize" Value="18"/>  <Setter Property="CornerRadius" Value="25"/>  <Setter Property="HeightRequest" Value="50"/>  <Setter Property="WidthRequest" Value="250"/>  </Style>  <Style TargetType="Entry">  <Setter Property="FontSize" Value="18"/>  <Setter Property="TextColor" Value="DeepPink"/>  <Setter Property="PlaceholderColor" Value="Pink"/>  </Style>  <Style TargetType="Editor">  <Setter Property="FontSize" Value="18"/>  <Setter Property="TextColor" Value="DeepPink"/>  <Setter Property="PlaceholderColor" Value="Pink"/>  </Style>  </ResourceDictionary>  </Application.Resources>  </Application> |

Pages/SignupPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  x:DataType="vm:AuthViewModel"  x:Class="IcecreamShop.APP.Pages.SignupPage"  Title="Signup"  Shell.FlyoutBehavior="Disabled">  <Grid>  <VerticalStackLayout VerticalOptions="Center"  Padding="20"  Spacing="20">  <Image Source="ice\_cream.png"  HeightRequest="150"  WidthRequest="150"  Aspect="AspectFit"/>  <Entry Text="{Binding Name}" Placeholder="Name"/>  <Entry Text="{Binding Email}" Keyboard="Email" Placeholder="Email"/>  <Entry Text="{Binding *Password*}" IsPassword="True" Placeholder="Password"/>  <Editor AutoSize="TextChanges"  MaxLength="150"  Placeholder="Address"  Text="{Binding Address}"/>  <Label Text="Existing User? Signin"  FontSize="15"  FontAttributes="Bold"  TextColor="DeepPink">  <Label.GestureRecognizers>  <TapGestureRecognizer Tapped="SigninLabel\_Tapped"/>  </Label.GestureRecognizers>  </Label>  <Button Text="Sign up" IsEnabled="{Binding CanSignup}" Command="{Binding SignupCommand}"/>  </VerticalStackLayout>  <ActivityIndicator HorizontalOptions="Center"  VerticalOptions="Center"  Color="DeepPink"  IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

Pages/ SigninPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  x:Class="IcecreamShop.APP.Pages.SigninPage"  x:DataType="vm:AuthViewModel"  Title="Signin"  Shell.FlyoutBehavior="Disabled">  <Grid>  <VerticalStackLayout VerticalOptions="Center"  Padding="20"  Spacing="20">  <Image Source="ice\_cream.png"  HeightRequest="150"  WidthRequest="150"  Aspect="AspectFit"/>  <Entry Text="{Binding Email}" Keyboard="Email" Placeholder="Email"/>  <Entry Text="{Binding *Password*}" IsPassword="True" Placeholder="Password"/>  <Label Text="new User? Signup"  FontSize="15"  FontAttributes="Bold"  TextColor="DeepPink">  <Label.GestureRecognizers>  <TapGestureRecognizer Tapped="Signup\_Tapped"/>  </Label.GestureRecognizers>  </Label>  <Button Text="Sign up" IsEnabled="{Binding CanSignin}" Command="{Binding SigninCommand}"/>  </VerticalStackLayout>  <ActivityIndicator HorizontalOptions="Center"  VerticalOptions="Center"  Color="DeepPink"  IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

Pages/ SigninPage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class SigninPage : ContentPage  {  public SigninPage(AuthViewModel authViewModel)  {  InitializeComponent();  BindingContext = authViewModel;  }  private async void Signup\_Tapped(object sender, TappedEventArgs e)  {  await Shell.Current.GoToAsync(nameof(SignupPage));  }  } |

26) 로그인 성공 후에 로그인 정보 남기기.

Services/AuthService.cs

|  |
| --- |
| namespace IcecreamShop.APP.Services;  public class AuthService  {  private const string AuthKey = "AuthKey";  public LoggedInUser User { get; private set; }  public string? Token { get; private set; }  public void Signin(AuthResponseDto dto)  {  var serialized = JsonSerializer.Serialize(dto);  Preferences.Default.Set(AuthKey, serialized);  (User, Token) = dto;  }  public void Initialize()  {  if (Preferences.Default.ContainsKey(AuthKey))  {  var serialized = Preferences.Default.Get<string?>(AuthKey, null);  if (string.IsNullOrWhiteSpace(serialized))  {  Preferences.Default.Remove(AuthKey);  }  else  {  (User, Token) = JsonSerializer.Deserialize<AuthResponseDto>(serialized)!;  }  }  }  public void Signout()  {  Preferences.Default.Remove(AuthKey);  (User, Token) = (null, null);  }  } |

MauiProgram.cs

|  |
| --- |
| builder.Services.AddSingleton<AuthService>();  builder.Services.AddTransient<OnboardingPage>(); |

App.xaml.cs

|  |
| --- |
| public App(AuthService authService)  {  InitializeComponent();  authService.Initialize();  MainPage = new AppShell(authService);  } |

ViewModels/AuthViewModel.cs 수정

|  |
| --- |
| public partial class AuthViewModel(IAuthApi authApi, AuthService authService) : BaseViewModel  {  private readonly IAuthApi \_authApi = authApi;  private readonly AuthService \_authService = authService;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_name;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignin)), NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_email;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignin)), NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_password;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanSignup))]  private string? \_address;  public bool CanSignup => !string.IsNullOrEmpty(Name)  && !string.IsNullOrEmpty(Email)  && !string.IsNullOrEmpty(Password)  && !string.IsNullOrEmpty(Address);  public bool CanSignin => !string.IsNullOrEmpty(Password)  && !string.IsNullOrEmpty(Email);  [RelayCommand]  private async Task SignupAsync()  {  IsBusy = true;  try  {  var signupDto = new SignupRequestDto(Name, Email, Password, Address);  // Make Api Call  var result = await \_authApi.SignupAsync(signupDto);  if (result.IsSuccess)  {  // Navigate to HomePage  \_authService.Signin(result.Data);  // await ShowAlertAsync(result.Data.User.Name);  await GotoAsync($"//{nameof(HomePage)}", animate: true);  }  else  {  // Display Error Alert  await ShowErrorAlertAsync(result.ErrorMessage??"Unknown error in signing up");  }  }  catch (Exception ex)  {  await ShowErrorAlertAsync(ex.Message);  }  finally  {  IsBusy = false;  }  }  [RelayCommand]  private async Task SigninAsync()  {  IsBusy = true;  try  {  var signinDto = new SigninRequestDto(Email, Password);  // Make Api Call  var result = await \_authApi.SigninAsync(signinDto);  if (result.IsSuccess)  {  // Navigate to HomePage  \_authService.Signin(result.Data);  //await ShowAlertAsync(result.Data.User.Name);  await GotoAsync($"//{nameof(HomePage)}", animate: true);  }  else  {  // Display Error Alert  await ShowErrorAlertAsync(result.ErrorMessage ?? "Unknown error in signing in");  }  }  catch (Exception ex)  {  await ShowErrorAlertAsync(ex.Message);  }  finally  {  IsBusy = false;  }  }  } |

OnboardingPage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class OnboardingPage : ContentPage  {  private readonly AuthService \_authService;  public OnboardingPage(AuthService authService)  {  InitializeComponent();  \_authService = authService;  }  protected async override void OnAppearing()  {  if(\_authService.User is not null && \_authService.User.Id != default(Guid)  && !string.IsNullOrWhiteSpace(\_authService.Token))  {  await Shell.Current.GoToAsync($"//{nameof(HomePage)}");  }  }  private async void Button\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync($"//{nameof(HomePage)}");  }  private async void Signin\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync(nameof(SigninPage));  }  private async void Signup\_Clicked(object sender, EventArgs e)  {  await Shell.Current.GoToAsync(nameof(SignupPage));  }  } |

AppShell.xaml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <Shell  x:Class="IcecreamShop.APP.AppShell"  xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:IcecreamShop.APP"  xmlns:pages="clr-namespace:IcecreamShop.APP.Pages"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  Shell.FlyoutBehavior="Flyout"  Title="IcecreamShop.APP">  <Shell.Behaviors>  <toolkit:StatusBarBehavior StatusBarColor="DeepPink"/>  </Shell.Behaviors>  <ShellContent  Title="Home"  ContentTemplate="{DataTemplate pages:OnboardingPage}"  Shell.FlyoutBehavior="Disabled"  Shell.FlyoutItemIsVisible="False"  Route="OnBoardingPage"/>  --- 이하 생략 ---- |

테스트)

로그인 성공 후 app 종료 후 다시 실행했을 때 로그인 화면이 뜨지 않으면 성공이다.

II. 아이스크림 관련

1. IcecreamShop.Shared 프로젝트에 IcecreamDto.cs를 추가한다.

Dtos/IcecreamDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record struct IcecreamOptionDto(string Flavor, string Topping);  public record IcecreamDto(int Id, string Name, string Image, double Price, IcecreamOptionDto[] Options); |

2. IcecreamShop.API 프로젝트에 IcecreamService.cs를 추가한다.

Services/IcecreamService.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class IcecreamService(DataContext context)  {  private readonly DataContext \_context = context;  public async Task<IcecreamDto[]> GetIcecreamsAsync() =>  await \_context.Icecreams.AsNoTracking()  .Select(i =>  new IcecreamDto(  i.Id,  i.Name,  i.Image,  i.Price,  i.Options  .Select(o => new IcecreamOptionDto(o.Flavor, o.Topping))  .ToArray()  )  )  .ToArrayAsync();    } |

3. IcecreamShop.API 프로젝트에 Program.cs에 아래와 같이 추가를 한다.

|  |
| --- |
| builder.Services.AddTransient<TokenService>()  .AddTransient<PasswordService>()  .AddTransient<AuthService>()  .AddTransient<IcecreamService>(); |

4. IcecreamShop.APP프로젝트에서 ViewModels폴더 안에 HomeViewModel.cs를 추가를 한다.

ViewModels/HomeViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class HomeViewModel(IIcecreamApi icecreamApi, AuthService authService) : BaseViewModel  {  private readonly IIcecreamApi \_icecreamApi = icecreamApi;  private readonly AuthService \_authService = authService;  [ObservableProperty]  private IcecreamDto[] \_icecreams = [];  [ObservableProperty]  private string \_userName = string.Empty;  private bool \_isInitialized;  public async Task InitializeAsync()  {  UserName = \_authService.User.Name;  if (\_isInitialized)  return;  IsBusy = true;  try  {  \_isInitialized = true;  // Make Api Call to fetch Icecreams  Icecreams = await \_icecreamApi.GetIcecreamsAsync();  }  catch (Exception ex)  {  await ShowErrorAlertAsync(ex.Message);  }  finally  {  IsBusy = false;  }  }  } |

5. IcecreamShop.APP프로젝트에서 Services 폴더에 IIcecreamApi.cs 추가를 한다.

|  |
| --- |
| namespace IcecreamShop.APP.Services;  public interface IIcecreamApi  {  [Get("/api/icecreams")]  Task<IcecreamDto[]> GetIcecreamsAsync();  } |

6. MauiProgram.cs

|  |
| --- |
| namespace IcecreamShop.APP  {  public static class MauiProgram  {  public static MauiApp CreateMauiApp()  {  var builder = MauiApp.CreateBuilder();  builder  .UseMauiApp<App>()  .UseMauiCommunityToolkit()  .ConfigureFonts(fonts =>  {  fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");  fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");  });  #if DEBUG  builder.Logging.AddDebug();  #endif  builder.Services.AddTransient<AuthViewModel>()  .AddTransient<SignupPage>()  .AddTransient<SigninPage>();  builder.Services.AddSingleton<AuthService>();  builder.Services.AddTransient<OnboardingPage>();  builder.Services.AddSingleton<HomeViewModel>()  .AddSingleton<HomePage>();  ConfigureRefit(builder.Services);  return builder.Build();  }  private static void ConfigureRefit(IServiceCollection services)  {  var refitSettings = new RefitSettings  {  HttpMessageHandlerFactory = () =>  {  #if ANDROID  return new AndroidMessageHandler  {  ServerCertificateCustomValidationCallback = (httpRequestMessage, certificate, chain, sslPolicyErrors) =>  {  return certificate?.Issuer == "CN=localhost" || sslPolicyErrors == SslPolicyErrors.None;  }  };  #elif IOS  return new NSUrlSessionHandler  {  TrustOverrideForUrl = (NSUrlSessionHandler sender, string url, SecTrust trust) =>  url.StartsWith("https://localhost")  };  #endif  return null;  }  };  services.AddRefitClient<IAuthApi>(refitSettings)  .ConfigureHttpClient(SetHttpClient);  services.AddRefitClient<IIcecreamApi>(refitSettings)  .ConfigureHttpClient(SetHttpClient);  static void SetHttpClient(HttpClient httpClient)  {  var baseUrl = DeviceInfo.Platform == DevicePlatform.Android  ? "https://10.0.2.2:7282" : "https://localhost:7282";  httpClient.BaseAddress = new Uri(baseUrl);  }  }  }  } |

7. Pages/HomePage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class HomePage : ContentPage  {  private readonly HomeViewModel \_homeViewModel;  public HomePage(HomeViewModel homeViewModel)  {  InitializeComponent();  BindingContext = \_homeViewModel = homeViewModel;  }  protected async override void OnAppearing()  {  await \_homeViewModel.InitializeAsync();  }  } |

8. ActivityIndicator 모든 페이지에 똑같이 사용하므로 전형화 시킨다.

App.xaml

|  |
| --- |
| <?xml version = "1.0" encoding = "UTF-8" ?>  <Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:local="clr-namespace:IcecreamShop.APP"  x:Class="IcecreamShop.APP.App">  <Application.Resources>  <ResourceDictionary>    **--- 중간 생략 ---**  <Style TargetType="ActivityIndicator">  <Setter Property="HorizontalOptions" Value="Center"/>  <Setter Property="VerticalOptions" Value="Center"/>  <Setter Property="Color" Value="DeepPink"/>  </Style>  </ResourceDictionary>  </Application.Resources>  </Application> |

Pages/Signin.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  x:Class="IcecreamShop.APP.Pages.SigninPage"  x:DataType="vm:AuthViewModel"  Title="Signin"  Shell.FlyoutBehavior="Disabled">  <Grid>  <VerticalStackLayout VerticalOptions="Center"  Padding="20"  Spacing="20">    **--- 중간 생략 ---**  </VerticalStackLayout>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

Pages/Signup.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  x:DataType="vm:AuthViewModel"  x:Class="IcecreamShop.APP.Pages.SignupPage"  Title="Signup"  Shell.FlyoutBehavior="Disabled">  <Grid>  <VerticalStackLayout VerticalOptions="Center"  Padding="20"  Spacing="20">    **--- 중간 생략 ---**  </VerticalStackLayout>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

Pages/HomePage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.HomePage"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  x:DataType="vm:HomeViewModel"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  Title="HomePage"  Shell.NavBarIsVisible="False"  Background="#EEE">  <Grid>  <ScrollView>  <VerticalStackLayout Padding="10, 25, 10, 10">  <VerticalStackLayout Spacing="10" Margin="0,0,0,25">  <Label  FontAttributes="Bold"  FontSize="35"  Text="{Binding *UserName*, *StringFormat*='Hello {}'}"/>  <Label  FontAttributes="Bold"  FontSize="18"  Text="What would you like to have today?"/>  </VerticalStackLayout>  <CollectionView ItemsSource="{Binding Icecreams}">  <CollectionView.ItemsLayout>  <GridItemsLayout  Orientation="Vertical"  Span="2"  VerticalItemSpacing="15"  HorizontalItemSpacing="15"/>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="dto:IcecreamDto">  <Border Background="#FBDBDE"  StrokeThickness="0"  StrokeShape="RoundRectangle 10">  <Grid RowDefinitions="\*, Auto, Auto">  <Image Grid.Row="0"  Source="{Binding Image}"  HeightRequest="200"  Aspect="AspectFill"/>  <Label Grid.Row="1"  Text="{Binding Name}"  FontSize="15"  HorizontalOptions="Center"  HorizontalTextAlignment="Center"  Padding="5, 10"  TextColor="DeepPink"  FontAttributes="Bold"/>  <Label Grid.Row="2"  Text="{Binding Price, *StringFormat*='{0:C}'"  FontSize="16"  HorizontalOptions="End"  FontAttributes="Bold"  Padding="5"/>  </Grid>  </Border>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </VerticalStackLayout>  </ScrollView>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

9. 품목의 세부정보 보기.

BaseViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class BaseViewModel : ObservableObject  {  **--- 중간 생략 ----**  protected async Task GotoAsync(string url, bool animate, IDictionary<string, object> parameters)  {  await Shell.Current.GoToAsync(url, animate, parameters);  }  } |

HomeViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class HomeViewModel(IIcecreamApi icecreamApi, AuthService authService) : BaseViewModel  {  **--- 중간 생략 ---**  private async Task GoToDetailsPageAsync(IcecreamDto icecream)  {  var parameter = new Dictionary<string, object>  {  [nameof(DetailsViewModel.Icecream)] = icecream,  };  //await Shell.Current.GoToAsync(nameof(DetailsPage), animate: true, parameter);  await GotoAsync(nameof(DetailsPage), animate: true, parameter);  }} |

HomePage.xaml

|  |
| --- |
| --- 중간 생략 ---  <Label Grid.Row="2"  Text="{Binding Price, *StringFormat*='{0:C}'"  FontSize="16"  HorizontalOptions="End"  FontAttributes="Bold"  Padding="5"/>  </Grid>  <Border.GestureRecognizers>  <TapGestureRecognizer Command="{Binding *Source*={x:RelativeSource *AncestorType*={x:Type vm:HomeViewModel}}}" CommandParameter="{Binding .}"/>  </Border.GestureRecognizers>  </Border>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </VerticalStackLayout>  </ScrollView>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

MauiProgram.cs를 추가한다.

|  |
| --- |
| builder.Services.AddTransient<DetailsViewModel>()  .AddTransient<DetailsPage>(); |

Pages/DetailsPage.xaml

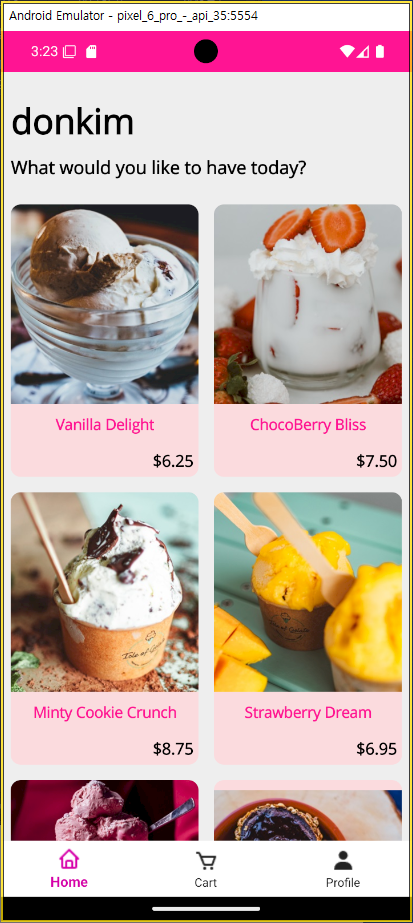
|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.DetailsPage"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:DetailsViewModel"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  Shell.PresentationMode="ModalAnimated"  Shell.NavBarIsVisible="False">  <!--<Shell.Behaviors>  <toolkit:StatusBarBehavior StatusBarColor="DeepPink"/>  </Shell.Behaviors>-->    <ContentPage.Resources>  <ResourceDictionary>  <Style TargetType="ImageButton" x:Key="quantityBtn">  <Setter Property="BackgroundColor" Value="DeepPink"/>  <Setter Property="VerticalOptions" Value="Center"/>  <Setter Property="HorizontalOptions" Value="Center"/>  <Setter Property="HeightRequest" Value="40"/>  <Setter Property="WidthRequest" Value="40"/>  <Setter Property="CornerRadius" Value="20"/>  <Setter Property="Padding" Value="5"/>  </Style>  </ResourceDictionary>  </ContentPage.Resources>  <Grid>  <Grid RowDefinitions="Auto, \*, Auto"  Padding="10, 30, 10, 10">  <VerticalStackLayout Grid.Row="0"  HorizontalOptions="Center"  Spacing="10">  <Image Source="{Binding Icecream.Image}"  HeightRequest="250"  WidthRequest="250"  HorizontalOptions="Center"  Aspect="AspectFit"/>  <Label Text="{Binding Icecream.Name}"  HorizontalOptions="Center"  HorizontalTextAlignment="Center"  FontAttributes="Bold"  TextColor="DeepPink"/>  <Label Text="{Binding Icecream.Price, *StringFormat*='{0:C}'}"  TextColor="DeepPink"  FontSize="20"  FontAttributes="Bold"  HorizontalOptions="Center"/>  </VerticalStackLayout>  <CollectionView Grid.Row="1"  ItemsSource="{Binding Icecream.Options}"  VerticalOptions="Center">  <CollectionView.ItemsLayout>  <GridItemsLayout Span="2"  Orientation="Vertical"  VerticalItemSpacing="10"  HorizontalItemSpacing="10">  </GridItemsLayout>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="dto:IcecreamOptionDto">  <Border StrokeShape="RoundRectangle 5"  Stroke="Gray"  Padding="5">  <VerticalStackLayout Spacing="5">  <VerticalStackLayout>  <Label Text="{Binding Flavor}"  TextColor="DeepPink"  FontSize="16"  FontAttributes="Bold"/>  <Label Text="Flavor" FontSize="12"/>  </VerticalStackLayout>  <VerticalStackLayout>  <Label Text="{Binding Topping}"  TextColor="DeepPink"  FontSize="16"  FontAttributes="Bold"/>  <Label Text="Topping" FontSize="12"/>  </VerticalStackLayout>  </VerticalStackLayout>  </Border>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  <VerticalStackLayout Grid.Row="2"  VerticalOptions="End"  Spacing="15">  <Grid ColumnDefinitions="\*, 100, \*">  <ImageButton Grid.Column="0" Source="minus.png"  Style="{x:StaticResource quantityBtn}"  Command="{Binding DecreaseQuantityCommand}">  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="White"/>  </ImageButton.Behaviors>  <!-- 0이면 "-"버튼이 비활성화 -->  <!--<ImageButton.Triggers>  <DataTrigger Binding="{Binding Quantity}"  TargetType = "ImageButton"  Value="0">  <Setter Property="IsEnabled" Value="False"/>  </DataTrigger>  </ImageButton.Triggers>-->  </ImageButton>  <Label Grid.Column="1"  Text="{Binding Quantity}"  VerticalOptions="Center"  HorizontalOptions="Center"  FontAttributes="Bold"  FontSize="25"/>  <ImageButton Grid.Column="2" Source="plus.png"  Style="{x:StaticResource quantityBtn}"  Command="{Binding IncreaseQuantityCommand}">  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="White"/>  </ImageButton.Behaviors>  </ImageButton>  </Grid>  <Button Text="Add to Cart"/>  </VerticalStackLayout>  </Grid>  <ImageButton Source="cross"  HorizontalOptions="Start"  VerticalOptions="Start"  BackgroundColor="DeepPink"  HeightRequest="50"  WidthRequest="50"  CornerRadius="25"  Padding="5"  Margin="10"  Command="{Binding GoBackCommand}">  <ImageButton.Shadow>  <Shadow/>  </ImageButton.Shadow>  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="White"/>  </ImageButton.Behaviors>  </ImageButton>  </Grid>  </ContentPage> |

Pages/DetailsPage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class DetailsPage : ContentPage  {  public DetailsPage(DetailsViewModel detailViewModel)  {  InitializeComponent();  BindingContext = detailViewModel;  }  } |

ViewModels/DetailsViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  [QueryProperty(nameof(Icecream), nameof(Icecream))]  public partial class DetailsViewModel : BaseViewModel  {  [ObservableProperty]  private IcecreamDto? \_icecream;  [ObservableProperty]  private int \_quantity;  [RelayCommand]  private void IncreaseQuantity()  {  Quantity++;  }  [RelayCommand]  private void DecreaseQuantity()  {  if (Quantity > 0)  {  Quantity--;  }  }  [RelayCommand]  private async Task GoBackAsync() => await GotoAsync("..", true);  } |

10. 옵션 선택하기.

Models/IcecreamOption.cs

|  |
| --- |
| namespace IcecreamShop.APP.Models;  public partial class IcecreamOption : BaseViewModel  {  public string Flavor { get; set; }  public string Topping { get; set; }  [ObservableProperty]  private bool \_isSelected;  } |

ViewModels/DetailsViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  [QueryProperty(nameof(Icecream), nameof(Icecream))]  public partial class DetailsViewModel : BaseViewModel  {  **--- 중간 생략 ---**  [RelayCommand]  private async Task GoBackAsync() => await GotoAsync("..", animate:true);  [RelayCommand]  private void SelectOption(IcecreamOption newOption)  {  var newIsSelected = !newOption.IsSelected;  // Deselect all options.  Options = [.. Options.Select(o => { o.IsSelected = false; return o; })];  newOption.IsSelected = newIsSelected;  }  } |

Pages/DetailsPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.DetailsPage"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:DetailsViewModel"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  Shell.PresentationMode="ModalAnimated"  Shell.NavBarIsVisible="False">  <!--<Shell.Behaviors>  <toolkit:StatusBarBehavior StatusBarColor="DeepPink"/>  </Shell.Behaviors>-->    <ContentPage.Resources>  <ResourceDictionary>  <Style TargetType="ImageButton" x:Key="quantityBtn">  <Setter Property="BackgroundColor" Value="DeepPink"/>  <Setter Property="VerticalOptions" Value="Center"/>  <Setter Property="HorizontalOptions" Value="Center"/>  <Setter Property="HeightRequest" Value="40"/>  <Setter Property="WidthRequest" Value="40"/>  <Setter Property="CornerRadius" Value="20"/>  <Setter Property="Padding" Value="5"/>  </Style>  </ResourceDictionary>  </ContentPage.Resources>  <Grid>  <Grid RowDefinitions="Auto, \*, Auto"  Padding="10, 30, 10, 10">  <VerticalStackLayout Grid.Row="0"  HorizontalOptions="Center"  Spacing="10">  <Image Source="{Binding Icecream.Image}"  HeightRequest="250"  WidthRequest="250"  HorizontalOptions="Center"  Aspect="AspectFit"/>  <Label Text="{Binding Icecream.Name}"  HorizontalOptions="Center"  HorizontalTextAlignment="Center"  FontAttributes="Bold"  TextColor="DeepPink"/>  <Label Text="{Binding Icecream.Price, *StringFormat*='{0:C}'}"  TextColor="DeepPink"  FontSize="20"  FontAttributes="Bold"  HorizontalOptions="Center"/>  </VerticalStackLayout>  <CollectionView Grid.Row="1"  ItemsSource="{Binding Options}"  VerticalOptions="Center">  <CollectionView.ItemsLayout>  <GridItemsLayout Span="2"  Orientation="Vertical"  VerticalItemSpacing="10"  HorizontalItemSpacing="10">  </GridItemsLayout>  </CollectionView.ItemsLayout>  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="models:IcecreamOption">  <Border StrokeShape="RoundRectangle 5"  Stroke="Gray"  Padding="5">  <Border.Triggers>  <DataTrigger TargetType="Border"  Binding="{Binding *IsSelected*}"  Value="True">  <Setter Property="Stroke" Value="DeepPink"/>  <Setter Property="StrokeThickness" Value="3"/>  </DataTrigger>  </Border.Triggers>  <VerticalStackLayout Spacing="5">  <VerticalStackLayout>  <Label Text="{Binding Flavor}"  TextColor="DeepPink"  FontSize="16"  FontAttributes="Bold"/>  <Label Text="Flavor" FontSize="12"/>  </VerticalStackLayout>  <VerticalStackLayout>  <Label Text="{Binding Topping}"  TextColor="DeepPink"  FontSize="16"  FontAttributes="Bold"/>  <Label Text="Topping" FontSize="12"/>  </VerticalStackLayout>  </VerticalStackLayout>  <Border.GestureRecognizers>  <TapGestureRecognizer  Command="{Binding *Source*={x:RelativeSource *AncestorType*={x:Type vm:DetailsViewModel}}, *Path*=SelectOptionCommand}"  CommandParameter="{Binding .}"/>  </Border.GestureRecognizers>  </Border>    </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  <VerticalStackLayout Grid.Row="2"  VerticalOptions="End"  Spacing="15">  <Grid ColumnDefinitions="\*, 100, \*">  <ImageButton Grid.Column="0" Source="minus.png"  Style="{x:StaticResource quantityBtn}"  Command="{Binding DecreaseQuantityCommand}">  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="White"/>  </ImageButton.Behaviors>  <!-- 0이면 "-"버튼이 비활성화 -->  <!--<ImageButton.Triggers>  <DataTrigger Binding="{Binding Quantity}"  TargetType = "ImageButton"  Value="0">  <Setter Property="IsEnabled" Value="False"/>  </DataTrigger>  </ImageButton.Triggers>-->  </ImageButton>  <Label Grid.Column="1"  Text="{Binding Quantity}"  VerticalOptions="Center"  HorizontalOptions="Center"  FontAttributes="Bold"  FontSize="25"/>  <ImageButton Grid.Column="2" Source="plus.png"  Style="{x:StaticResource quantityBtn}"  Command="{Binding IncreaseQuantityCommand}">  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="White"/>  </ImageButton.Behaviors>  </ImageButton>  </Grid>  <Button Text="Add to Cart"/>  </VerticalStackLayout>  </Grid>  <ImageButton Source="cross"  HorizontalOptions="Start"  VerticalOptions="Start"  BackgroundColor="DeepPink"  HeightRequest="50"  WidthRequest="50"  CornerRadius="25"  Padding="5"  Margin="10"  Command="{Binding GoBackCommand}">  <ImageButton.Shadow>  <Shadow/>  </ImageButton.Shadow>  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="White"/>  </ImageButton.Behaviors>  </ImageButton>  </Grid>  </ContentPage> |

11. 장바구니 구현

Models/CartItem.cs

|  |
| --- |
| namespace IcecreamShop.APP.Models;  public partial class CartItem : ObservableObject  {  public int Id { get; set; }  public int IcecreamId { get; set; }  public string Name { get; set; }  public double Price { get; set; }  public string FlavorName { get; set; }  public string ToppingName { get; set; }  [ObservableProperty, NotifyPropertyChangedFor(nameof (TotalPrice))]  public int \_quantity;  public double TotalPrice => Price \* Quantity;  } |

ViewModels/BaseViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class BaseViewModel : ObservableObject  {  --- 중간 생략 ------  protected async Task ShowToastAsync(string message)  {  await Toast.Make(message).Show();  }  } |

ViewModels/CartViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class CartViewModel : BaseViewModel  {  public ObservableCollection<CartItem> CartItems { get; set; } = [];  public static int TotalCartCount { get; set; }  public static event EventHandler<int>? TotalCartCountChanged;  public async void AddItemToCart(IcecreamDto icecream, int quantity, string flavor, string topping)  {  var existingItem = CartItems.FirstOrDefault(ci => ci.IcecreamId == icecream.Id);  if ((existingItem is not null))  {  if(quantity <= 0)  {  CartItems.Remove(existingItem);  await ShowToastAsync("Icecream removed from the cart");  }  else  {  existingItem.Quantity = quantity;  await ShowToastAsync("Quantity updated in the cart");  }  }  else  {  var cartItem = new CartItem  {  IcecreamId = icecream.Id,  Id = icecream.Id,  Name = icecream.Name,  Price = icecream.Price,  Quantity = quantity,  FlavorName = flavor,  ToppingName = topping,  };  CartItems.Add(cartItem);  await ShowToastAsync("Icecream added to cart");  }  TotalCartCount = CartItems.Sum(i => i.Quantity);  TotalCartCountChanged?.Invoke(null, TotalCartCount);  }  } |

Platforms/Android/TabbarBadgeRenderer.cs

|  |
| --- |
| namespace IcecreamShop.APP;  public class TabbarBadgeRenderer : ShellRenderer  {  protected override IShellBottomNavViewAppearanceTracker CreateBottomNavViewAppearanceTracker(ShellItem shellItem)  {  //return base.CreateBottomNavViewAppearanceTracker(shellItem);  return new BadgeShellBottomNavViewAppearanceTracker(this, shellItem);  }  class BadgeShellBottomNavViewAppearanceTracker : ShellBottomNavViewAppearanceTracker  {  private BadgeDrawable \_badgeDrawable;  public BadgeShellBottomNavViewAppearanceTracker(IShellContext shellContext, ShellItem shellItem) : base(shellContext, shellItem)  {  }  public override void SetAppearance(BottomNavigationView bottomView, IShellAppearanceElement appearance)  {  base.SetAppearance(bottomView, appearance);  const int cartTabbarItemIndex = 1;  \_badgeDrawable = bottomView.GetOrCreateBadge(cartTabbarItemIndex);  UpdateBadge(CartViewModel.TotalCartCount);  CartViewModel.TotalCartCountChanged += CartViewModel\_TotalCartCountChanged;  }  private void CartViewModel\_TotalCartCountChanged(object? sender, int newTotalCartCount) => UpdateBadge(newTotalCartCount);  private void UpdateBadge(int count)  {  if (count <= 0)  {  \_badgeDrawable.SetVisible(false);  }  else  {  \_badgeDrawable.Number = count;  \_badgeDrawable.BackgroundColor = Colors.DeepPink.ToPlatform();  \_badgeDrawable.BadgeTextColor = Colors.White.ToPlatform();  \_badgeDrawable.SetVisible(true);  }    }  protected override void Dispose(bool disposing)  {  CartViewModel.TotalCartCountChanged -= CartViewModel\_TotalCartCountChanged;  base.Dispose(disposing);  }  }  } |

Platforms/IOS/TabbarBadgeRenderer.cs

|  |
| --- |
| namespace IcecreamShop.APP;  public class TabbarBadgeRenderer : ShellRenderer  {  protected override IShellTabBarAppearanceTracker CreateTabBarAppearanceTracker()  {  return new BadgeShellTabbarAppearanceTracker();  }  class BadgeShellTabbarAppearanceTracker : ShellTabBarAppearanceTracker  {  private UITabBarItem? \_cartTabbarItem;  public override void UpdateLayout(UITabBarController controller)  {  base.UpdateLayout(controller);  if(\_cartTabbarItem is null)  {  const int cartTabbarItemIndex = 1;  \_cartTabbarItem = controller.TabBar.Items?[cartTabbarItemIndex];  UpdateBadge(CartViewModel.TotalCartCount);  CartViewModel.TotalCartCountChanged += CartViewModel\_TotalCartCountChanged;  }    }  private void CartViewModel\_TotalCartCountChanged(object? sender, int newCount)  {  UpdateBadge(newCount);  }  private void UpdateBadge(int count)  {  if (\_cartTabbarItem is not null)  {  if (count <= 0)  {  \_cartTabbarItem.BadgeValue = null;  }  else  {  \_cartTabbarItem.BadgeValue = count.ToString();  \_cartTabbarItem.BadgeColor = Colors.DeepPink.ToPlatform();  //\_cartTabbarItem.BadgeTextColor = Colors.White.ToPlatform();  }  }  }  protected override void Dispose(bool disposing)  {  CartViewModel.TotalCartCountChanged -= CartViewModel\_TotalCartCountChanged;  base.Dispose(disposing);  }  }  } |

MauiProgram.cs

|  |
| --- |
| namespace IcecreamShop.APP  {  public static class MauiProgram  {  public static MauiApp CreateMauiApp()  {  var builder = MauiApp.CreateBuilder();  builder  .UseMauiApp<App>()  .UseMauiCommunityToolkit()  .ConfigureMauiHandlers(h =>  {  #if ANDROID || IOS  h.AddHandler<Shell, TabbarBadgeRenderer>();  #endif  }  )  .ConfigureFonts(fonts =>  {  fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");  fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");  });  #if DEBUG  builder.Logging.AddDebug();  #endif  builder.Services.AddTransient<AuthViewModel>()  .AddTransient<SignupPage>()  .AddTransient<SigninPage>();  builder.Services.AddSingleton<AuthService>();  builder.Services.AddTransient<OnboardingPage>();  builder.Services.AddSingleton<HomeViewModel>()  .AddSingleton<HomePage>();  builder.Services.AddTransient<DetailsViewModel>()  .AddTransient<DetailsPage>();  builder.Services.AddSingleton<CartViewModel>();  ConfigureRefit(builder.Services);  return builder.Build();  }  --- 이하 생략 ----  }  } |

ViewModels/DetailsViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  [QueryProperty(nameof(Icecream), nameof(Icecream))]  public partial class DetailsViewModel : BaseViewModel  {  private readonly CartViewModel \_cartViewModel;  public DetailsViewModel(CartViewModel cartViewModel)  {  \_cartViewModel = cartViewModel;  }  [ObservableProperty]  private IcecreamDto? \_icecream;  [ObservableProperty]  private int \_quantity;  [ObservableProperty]  private IcecreamOption[] \_options = [];  partial void OnIcecreamChanged(IcecreamDto? value)  {  Options = [];  if(value is null)  {  return;  }  Options = value.Options.Select(o => new IcecreamOption  {  Flavor = o.Flavor,  Topping = o.Topping,  IsSelected = false  }).ToArray();  }  [RelayCommand]  private void IncreaseQuantity()  {  Quantity++;  }  [RelayCommand]  private void DecreaseQuantity()  {  if (Quantity > 0)  {  Quantity--;  }  }  [RelayCommand]  private async Task GoBackAsync() => await GotoAsync("..", animate:true);  [RelayCommand]  private void SelectOption(IcecreamOption newOption)  {  var newIsSelected = !newOption.IsSelected;  // Deselect all options.  Options = [.. Options.Select(o => { o.IsSelected = false; return o; })];  newOption.IsSelected = newIsSelected;  }  [RelayCommand]  private void AddToCart()  {  var selectedOption = Options.FirstOrDefault(o => o.IsSelected)??Options[0];  \_cartViewModel.AddItemToCart(Icecream, Quantity, selectedOption.Flavor, selectedOption.Topping);  }  } |

Pages/DetailsPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  **---- 중간 생략 ----**  </Grid>  <Button Text="Add to Cart" Command="{Binding AddToCartCommand}"/>  </VerticalStackLayout>  </Grid>  **---- 중간 생략 ----**  </Grid>  </ContentPage> |

12. 장바구니에 항목을 SQLite에 담기

1) 프로젝트 IcecreamShop.APP에서 2개의 패키지 추가

sqlite-net-pcl

SQLitePCLRaw.bundle\_green

2) 프로젝트 IcecreamShop.APP에서 Data폴더를 생성하고 CartItemEntity.cs를 추가한다.

Data/CartItemEntity.cs

|  |
| --- |
| namespace IcecreamShop.APP.Data;  public class CartItemEntity  {  [PrimaryKey, AutoIncrement]  public int Id { get; set; }  public int IcecreamId { get; set; }  public string Name { get; set; }  public double Price { get; set; }  public string FlavorName { get; set; }  public string ToppingName { get; set; }  public int Quantity { get; set; }  public CartItemEntity(CartItem cartItemModel)  {  IcecreamId = cartItemModel.IcecreamId;  Name = cartItemModel.Name;  Price = cartItemModel.Price;  FlavorName = cartItemModel.FlavorName;  ToppingName = cartItemModel.ToppingName;  Quantity = cartItemModel.Quantity;  }  public CartItemEntity() { }  public CartItem ToCartItemModel() =>  new()  {  Id = Id,  Name = Name,  Price = Price,  FlavorName = FlavorName,  ToppingName = ToppingName,  IcecreamId = IcecreamId,  Quantity = Quantity  };  } |

ViewModels/CartViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class CartViewModel : BaseViewModel  {  private readonly DatabaseService \_databaseService;  public CartViewModel(DatabaseService databaseService)  **{**  \_databaseService = databaseService;  **}**  public ObservableCollection<CartItem> CartItems { get; set; } = [];  public static int TotalCartCount { get; set; }  public static event EventHandler<int>? TotalCartCountChanged;  public async void AddItemToCart(IcecreamDto icecream, int quantity, string flavor, string topping)  {  var existingItem = CartItems.FirstOrDefault(ci => ci.IcecreamId == icecream.Id);  if ((existingItem is not null))  {  var dbCartItem = await \_databaseService.GetCartItemAsync(existingItem.Id);  if (quantity <= 0)  {  await \_databaseService.DeleteCartItem(dbCartItem);  CartItems.Remove(existingItem);  await ShowToastAsync("Icecream removed from the cart");  }  else  {    dbCartItem.Quantity = quantity;  await \_databaseService.UpdateCartItem(dbCartItem);  existingItem.Quantity = quantity;  await ShowToastAsync("Quantity updated in the cart");  }  }  else  {  var cartItem = new CartItem  {  IcecreamId = icecream.Id,  Id = icecream.Id,  Name = icecream.Name,  Price = icecream.Price,  Quantity = quantity,  FlavorName = flavor,  ToppingName = topping,  };    // sqlite 에 기록  var entity = new Data.CartItemEntity(cartItem);  await \_databaseService.AddCartItem(entity);  // sqlite id를 저장.  cartItem.Id = entity.Id;  CartItems.Add(cartItem);  await ShowToastAsync("Icecream added to cart");  }  NotifyCartCountChange();  }  private void NotifyCartCountChange()  {  TotalCartCount = CartItems.Sum(i => i.Quantity);  TotalCartCountChanged?.Invoke(null, TotalCartCount);  }  public int GetItemCartCount(int icecreamId)  {  var existingItem = CartItems.FirstOrDefault(i => i.IcecreamId == icecreamId);  return existingItem?.Quantity?? 0;  }  public async Task InitializeCartAsync()  {  var dbItems = await \_databaseService.GetAllCartItemAsync();  foreach (var dbItem in dbItems)  {  CartItems.Add(dbItem.ToCartItemModel());  }  NotifyCartCountChange();  }  } |

ViewModels/DetailsViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  [QueryProperty(nameof(Icecream), nameof(Icecream))]  public partial class DetailsViewModel : BaseViewModel  {  private readonly CartViewModel \_cartViewModel;  public DetailsViewModel(CartViewModel cartViewModel)  {  \_cartViewModel = cartViewModel;  }  [ObservableProperty]  private IcecreamDto? \_icecream;  [ObservableProperty]  private int \_quantity;  [ObservableProperty]  private IcecreamOption[] \_options = [];  partial void OnIcecreamChanged(IcecreamDto? value)  {  Options = [];  if(value is null)  {  return;  }  Options = value.Options.Select(o => new IcecreamOption  {  Flavor = o.Flavor,  Topping = o.Topping,  IsSelected = false  }).ToArray();  Quantity = \_cartViewModel.GetItemCartCount(value.Id);  }  [RelayCommand]  private void IncreaseQuantity()  {  Quantity++;  }  [RelayCommand]  private void DecreaseQuantity()  {  if (Quantity > 0)  {  Quantity--;  }  }  [RelayCommand]  private async Task GoBackAsync() => await GotoAsync("..", animate:true);  [RelayCommand]  private void SelectOption(IcecreamOption newOption)  {  var newIsSelected = !newOption.IsSelected;  // Deselect all options.  Options = [.. Options.Select(o => { o.IsSelected = false; return o; })];  newOption.IsSelected = newIsSelected;  }  [RelayCommand]  private Task AddToCart()  {  var selectedOption = Options.FirstOrDefault(o => o.IsSelected)??Options[0];  \_cartViewModel.AddItemToCart(Icecream, Quantity, selectedOption.Flavor, selectedOption.Topping);  }  } |

Services/DatabaseService.cs

|  |
| --- |
| namespace IcecreamShop.APP.Services;  public class DatabaseService : IAsyncDisposable  {  private const string DatabaseName = "Icecream.db3";  private static readonly string \_databasePath=Path.Combine(FileSystem.AppDataDirectory, DatabaseName);  private SQLiteAsyncConnection? \_connection;  private bool disposedValue;  private SQLiteAsyncConnection Database =>  \_connection??=new SQLiteAsyncConnection(\_databasePath, SQLiteOpenFlags.Create | SQLiteOpenFlags.ReadWrite | SQLiteOpenFlags.SharedCache);  private async Task<TResult> ExecuteAsync<TResult>(Func<Task<TResult>> actionOnDb)  {  await Database.CreateTableAsync<CartItemEntity>();  return await actionOnDb.Invoke();  }  public async Task<int> AddCartItem(CartItemEntity entity) =>  await ExecuteAsync(async () => await Database.InsertAsync(entity));  public async Task<int> UpdateCartItem(CartItemEntity cartItem) =>  await ExecuteAsync(async () => await Database.UpdateAsync(cartItem));  public async Task<int> DeleteCartItem(CartItemEntity entity) =>  await ExecuteAsync(async () => await Database.DeleteAsync(entity));  public async Task<CartItemEntity> GetCartItemAsync(int id) =>  await ExecuteAsync(async () => await Database.GetAsync<CartItemEntity> (id));  public async Task<List<CartItemEntity>> GetAllCartItemAsync() =>  await ExecuteAsync(async () => await Database.Table<CartItemEntity>().ToListAsync());  public async ValueTask DisposeAsync()  {  await \_connection?.CloseAsync();  }  } |

App.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP  {  public partial class App : Application  {  private readonly CartViewModel \_cartViewModel;  public App(AuthService authService, CartViewModel cartViewModel)  {  InitializeComponent();  authService.Initialize();  \_cartViewModel = cartViewModel;  MainPage = new AppShell(authService);  }  protected override async void OnStart()  {  await \_cartViewModel.InitializeCartAsync();  }  #if WINDOWS  protected override Window CreateWindow(IActivationState activationState)  {  var window = base.CreateWindow(activationState);  const int newWidth = 540;  const int newHeight = 960;  window.Width = newWidth;  window.Height = newHeight;  return window;  }  #endif  }  } |

MauiProgram.cs 아래와 같이 서비스 등록한다.

|  |
| --- |
| builder.Services.AddSingleton<DatabaseService>();  --- 중간 생략 ---  builder.Services.AddSingleton<CartViewModel>()  .AddTransient<CartPage>(); |

13. 장바구니의 아이템 리스트 보여주기.

ViewModels/DetailsViewModel.cs 추가하기

|  |
| --- |
| --- 이전 내용 생략 됨 -----  [RelayCommand]  private async Task AddToCart()  {  var selectedOption = Options.FirstOrDefault(o => o.IsSelected)??Options[0];  \_cartViewModel.AddItemToCart(Icecream!, Quantity, selectedOption.Flavor, selectedOption.Topping);  await GoBackAsync();  } |

ViewModes/BaseViewModels.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class BaseViewModel : ObservableObject  {  [ObservableProperty]  private bool \_isBusy;  protected async Task GotoAsync(string url, bool animate)  {  await Shell.Current.GoToAsync(url, animate);  }  protected async Task GotoAsync(string url, bool animate, IDictionary<string, object> parameters)  {  await Shell.Current.GoToAsync(url, animate, parameters);  }  protected async Task ShowErrorAlertAsync(string errorMessage)  {  await Shell.Current.DisplayAlert("Error", errorMessage, "Ok");  }  protected async Task ShowAlertAsync(string message)  {  await Shell.Current.DisplayAlert("Alert", message, "Ok");  }  protected async Task ShowAlertAsync(string title, string message)  {  await Shell.Current.DisplayAlert(title, message, "Ok");  }  protected async Task ShowToastAsync(string message)  {  await Toast.Make(message).Show();  }  protected async Task<bool> ConfirmAsync(string title, string message)  {  return await Shell.Current.DisplayAlert(title, message, "Yes", "No");  }  } |

ViewModes/CartViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class CartViewModel : BaseViewModel  {  private readonly DatabaseService \_databaseService;  public CartViewModel(DatabaseService databaseService)  {  \_databaseService = databaseService;  }  public ObservableCollection<CartItem> CartItems { get; set; } = [];  public static int TotalCartCount { get; set; }  public static event EventHandler<int>? TotalCartCountChanged;  public async void AddItemToCart(IcecreamDto icecream, int quantity, string flavor, string topping)  {  var existingItem = CartItems.FirstOrDefault(ci => ci.IcecreamId == icecream.Id);  if ((existingItem is not null))  {  var dbCartItem = await \_databaseService.GetCartItemAsync(existingItem.Id);  if (quantity <= 0)  {  await \_databaseService.DeleteCartItem(dbCartItem);  CartItems.Remove(existingItem);  await ShowToastAsync("Icecream removed from the cart");  }  else  {    dbCartItem.Quantity = quantity;  await \_databaseService.UpdateCartItem(dbCartItem);  existingItem.Quantity = quantity;  await ShowToastAsync("Quantity updated in the cart");  }  }  else  {  var cartItem = new CartItem  {  IcecreamId = icecream.Id,  Id = icecream.Id,  Name = icecream.Name,  Price = icecream.Price,  Quantity = quantity,  FlavorName = flavor,  ToppingName = topping,  };    // sqlite 에 기록  var entity = new Data.CartItemEntity(cartItem);  await \_databaseService.AddCartItem(entity);  // sqlite id를 저장.  cartItem.Id = entity.Id;  CartItems.Add(cartItem);  await ShowToastAsync("Icecream added to cart");  }  NotifyCartCountChange();  }  private void NotifyCartCountChange()  {  TotalCartCount = CartItems.Sum(i => i.Quantity);  TotalCartCountChanged?.Invoke(null, TotalCartCount);  }  public int GetItemCartCount(int icecreamId)  {  var existingItem = CartItems.FirstOrDefault(i => i.IcecreamId == icecreamId);  return existingItem?.Quantity?? 0;  }  public async Task InitializeCartAsync()  {  var dbItems = await \_databaseService.GetAllCartItemAsync();  foreach (var dbItem in dbItems)  {  CartItems.Add(dbItem.ToCartItemModel());  }  NotifyCartCountChange();  }  [RelayCommand]  private async Task ClearCartAsync()  {  if (CartItems.Count == 0)  {  await ShowAlertAsync("Empty Cart", "There's no items in the cart");  return;  }  if(await ConfirmAsync("Clear Cart?", "Do you really want to clear all the items from the cart"))  {  await \_databaseService.ClearCartAsync();  CartItems.Clear();  await ShowToastAsync("Cart cleared");  NotifyCartCountChange();  }  }  [RelayCommand]  private async Task RemoveCartItemAsync(int cartItemId)  {  if (await ConfirmAsync("Remove item from Cart?", "Do you really want to delete this item items from the cart"))  {  var existingItem = CartItems.FirstOrDefault(i => i.Id == cartItemId);  if ((existingItem is null))  {  return;  }  CartItems.Remove(existingItem);  var dbCartItem = await \_databaseService.GetCartItemAsync(existingItem.Id);  if ((dbCartItem is null))  {  return;  }  await \_databaseService.DeleteCartItem(dbCartItem);  await ShowToastAsync("Icecream removed from cart");  NotifyCartCountChange();  }  }  } |

Services/DatabaseService.s

|  |
| --- |
| namespace IcecreamShop.APP.Services;  public class DatabaseService : IAsyncDisposable  {  private const string DatabaseName = "Icecream.db3";  private static readonly string \_databasePath=Path.Combine(FileSystem.AppDataDirectory, DatabaseName);  private SQLiteAsyncConnection? \_connection;  private bool disposedValue;  private SQLiteAsyncConnection Database =>  \_connection??=new SQLiteAsyncConnection(\_databasePath, SQLiteOpenFlags.Create | SQLiteOpenFlags.ReadWrite | SQLiteOpenFlags.SharedCache);  private async Task<TResult> ExecuteAsync<TResult>(Func<Task<TResult>> actionOnDb)  {  await Database.CreateTableAsync<CartItemEntity>();  return await actionOnDb.Invoke();  }  public async Task<int> AddCartItem(CartItemEntity entity) =>  await ExecuteAsync(async () => await Database.InsertAsync(entity));  public async Task<int> UpdateCartItem(CartItemEntity cartItem) =>  await ExecuteAsync(async () => await Database.UpdateAsync(cartItem));  public async Task<int> DeleteCartItem(CartItemEntity entity) =>  await ExecuteAsync(async () => await Database.DeleteAsync(entity));  public async Task<CartItemEntity> GetCartItemAsync(int id) =>  await ExecuteAsync(async () => await Database.GetAsync<CartItemEntity> (id));  public async Task<List<CartItemEntity>> GetAllCartItemAsync() =>  await ExecuteAsync(async () => await Database.Table<CartItemEntity>().ToListAsync());  public async Task<int> ClearCartAsync() =>  await ExecuteAsync(async() => await Database.DeleteAllAsync<CartItemEntity>());  public async ValueTask DisposeAsync()  {  await \_connection?.CloseAsync();  }  } |

Pages/CartPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:CartViewModel"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="IcecreamShop.APP.Pages.CartPage"  Title="CartPage"  BackgroundColor="#EEE">  <ContentPage.ToolbarItems>  <ToolbarItem Text="Clear Cart" Command="{Binding ClearCartCommand}"/>  </ContentPage.ToolbarItems>    <Grid Padding="10">  <ScrollView Padding="0, 0, 0, 60">  <CollectionView ItemsSource="{Binding CartItems}">  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="models:CartItem">  <Border StrokeShape="RoundRectangle 10"  Stroke="LightPink"  BackgroundColor="LightPink"  Padding="10">  <Grid ColumnDefinitions="40, \*, Auto, 50">  <Label Text="{Binding Quantity}"  VerticalOptions="Center"  FontSize="16"  FontAttributes="Bold"/>  <VerticalStackLayout Grid.Column="1">  <Label Text="{Binding Name}"  FontSize="15"  FontAttributes="Bold"  TextColor="DeepPink"/>  <Label FontSize="12">  <Label.FormattedText>  <FormattedString>  <Span Text="{Binding FlavorName}"/>  <Span Text=", "/>  <Span Text="{Binding ToppingName}"/>  </FormattedString>  </Label.FormattedText>  </Label>  </VerticalStackLayout>  <Label Grid.Column="2"  Text="{Binding Price, *StringFormat*='{0:C}'}"  FontAttributes="Bold"  FontSize="15"  VerticalOptions="Center"  HorizontalOptions="End"/>  <ImageButton Grid.Column="3" Source="trash"  HeightRequest="24"  WidthRequest="24"  VerticalOptions="Center"  HorizontalOptions="End"  Command="{Binding *Source*={x:RelativeSource *AncestorType*={x:Type vm:CartViewModel}}, *Path*=RemoveCartItemCommand}"  CommandParameter="{Binding Id}">  <ImageButton.Behaviors>  <toolkit:IconTintColorBehavior TintColor="Maroon"/>  </ImageButton.Behaviors>  </ImageButton>  </Grid>  </Border>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </ScrollView>  <Button Text="Place Order"  VerticalOptions="End"  HorizontalOptions="Center"/>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

Pages/CartPage.xaml.cs

|  |
| --- |
| public partial class CartPage : ContentPage  {  public CartPage(CartViewModel cartViewModel)  {  InitializeComponent();  BindingContext = cartViewModel;  }  } |

14. 주문하기

1) IcecreamShop.Shared 프로젝트에서 OrderDto.cs추가

Dtos/OrderDto.cs

|  |
| --- |
| public record OrderItemDto(long Id, int IcecreamId, string Name, int Quantity, double Price, string Flavor, string Topping)  {  public double TotalPrice => Quantity \* Price;  }  public record OrderPlaceDto(OrderDto Order, OrderItemDto[] Items);  public record OrderDto(long Id, DateTime OrderAt, double TotalPrice); |

2) IcecreamShop.API프로젝트에서 DB로부터 주문 정보를 처리하기 위해서 Service/OrderService.cs추가 한다.

Service/OrderService.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class OrderService(DataContext context)  {  private readonly DataContext \_context = context;  public async Task<ResultDto> PlaceOrderAsync(OrderPlaceDto dto, Guid customerId)  {  var customer = await \_context.Users.FirstOrDefaultAsync(u => u.Id == customerId);  if(customer is null)  {// error - no customer  return ResultDto.Failure("Customer does not exist");  }  var orderItems = dto.Items.Select(i =>  new OrderItem  {  Flavor = i.Flavor,  IcecreamId = i.IcecreamId,  Name = i.Name,  Price = i.Price,  Quantity = i.Quantity,  Topping = i.Topping,  TotalPrice = i.TotalPrice,  });  var order = new Order  {  CustomerId = customerId,  CustomerAddress = customer.Address,  CustomerEmail = customer.Email,  CustomerName = customer.Name,  OrderAt = DateTime.Now,  TotalPrice = orderItems.Sum(o=>o.TotalPrice),  };  try  {  await \_context.Orders.AddAsync(order);  await \_context.SaveChangesAsync();  return ResultDto.Success();  }  catch (Exception ex)  {  return ResultDto.Failure(ex.Message);  }  }  } |

3) IcecreamShop.API프로젝트에서 Program.cs에서의 일부코드에서 추가를 한다

|  |
| --- |
| builder.Services.AddTransient<TokenService>()  .AddTransient<PasswordService>()  .AddTransient<AuthService>()  .AddTransient<IcecreamService>()  .AddTransient<OrderService>(); |

4) IcecreamShop.APP프로젝트에서 EndPoints/AuthEndPoints.cs 일부분 코드 추가

|  |
| --- |
| namespace IcecreamShop.API.EndPoints;  public static class AuthEndpoints  {  private static Guid GetUserId(this ClaimsPrincipal principal) =>  Guid.Parse(principal.FindFirstValue(ClaimTypes.NameIdentifier)!);  public static IEndpointRouteBuilder MapEndPoints(this IEndpointRouteBuilder app)  {  app.MapPost("/api/signup",  async (SignupRequestDto dto, AuthService authService) =>  TypedResults.Ok(await authService.SignupAsync(dto)));  app.MapPost("/api/signin",  async (SigninRequestDto dto, AuthService authService) =>  TypedResults.Ok(await authService.SigninAsync(dto)));  app.MapGet("/api/icecreams",  async (IcecreamService icecreamService) =>  TypedResults.Ok(await icecreamService.GetIcecreamsAsync()));  var orderGroup = app.MapGroup("/api/order").RequireAuthorization();  orderGroup.MapPost("/place-order",  async(OrderPlaceDto dto, ClaimsPrincipal principal, OrderService orderService) =>  await orderService.PlaceOrderAsync(dto, principal.GetUserId()));  return app;  }  } |

5) IcecreamShop.APP프로젝트에서 Service/IOrderApi.cs 추가 한다.

|  |
| --- |
| namespace IcecreamShop.APP.Services;  [Headers("Authorization Bearer")]  public interface IOrderApi  {  [Post("/api/order/place-order")]  Task<ResultDto> PlaceOrderAsync(OrderPlaceDto dto);  } |

5) MauiProgram.cs

|  |
| --- |
| namespace IcecreamShop.APP  {  public static class MauiProgram  {  public static MauiApp CreateMauiApp()  {  var builder = MauiApp.CreateBuilder();  builder  .UseMauiApp<App>()  .UseMauiCommunityToolkit()  .ConfigureMauiHandlers(h =>  {  #if ANDROID || IOS  h.AddHandler<Shell, TabbarBadgeRenderer>();  #endif  }  )  .ConfigureFonts(fonts =>  {  fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");  fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");  });  #if DEBUG  builder.Logging.AddDebug();  #endif  builder.Services.AddSingleton<DatabaseService>();  builder.Services.AddTransient<AuthViewModel>()  .AddTransient<SignupPage>()  .AddTransient<SigninPage>();  builder.Services.AddSingleton<AuthService>();  builder.Services.AddTransient<OnboardingPage>();  builder.Services.AddSingleton<HomeViewModel>()  .AddSingleton<HomePage>();  builder.Services.AddTransient<DetailsViewModel>()  .AddTransient<DetailsPage>();  builder.Services.AddSingleton<CartViewModel>()  .AddTransient<CartPage>();  ConfigureRefit(builder.Services);  return builder.Build();  }  private static void ConfigureRefit(IServiceCollection services)  {    services.AddRefitClient<IAuthApi>(GetRefitSetting)  .ConfigureHttpClient(SetHttpClient);  services.AddRefitClient<IIcecreamApi>(GetRefitSetting)  .ConfigureHttpClient(SetHttpClient);  services.AddRefitClient<IOrderApi>(GetRefitSetting)  .ConfigureHttpClient(SetHttpClient);  static void SetHttpClient(HttpClient httpClient)  {  var baseUrl = DeviceInfo.Platform == DevicePlatform.Android  ? "https://10.0.2.2:7282" : "https://localhost:7282";  if (DeviceInfo.DeviceType == DeviceType.Physical)  {  baseUrl = "https://82s5gr41-7282.asse.devtunnels.ms";  }  httpClient.BaseAddress = new Uri(baseUrl);  }  static RefitSettings GetRefitSetting(IServiceProvider serviceProvider)  {  var authService = serviceProvider.GetRequiredService<AuthService>();  var refitSettings = new RefitSettings  {  HttpMessageHandlerFactory = () =>  {  #if ANDROID  return new AndroidMessageHandler  {  ServerCertificateCustomValidationCallback = (httpRequestMessage, certificate, chain, sslPolicyErrors) =>  {  return certificate?.Issuer == "CN=localhost" || sslPolicyErrors == SslPolicyErrors.None;  }  };  #elif IOS  return new NSUrlSessionHandler  {  TrustOverrideForUrl = (NSUrlSessionHandler sender, string url, SecTrust trust) =>  url.StartsWith("https://localhost")  };  #endif  return null;  },  AuthorizationHeaderValueGetter = (\_, \_\_) =>  Task.FromResult(authService.Token ?? string.Empty)  };  return refitSettings;  }  }  }  } |

6) IcecreamShop.APP프로젝트에서 ViewModels/CartViewModel.cs에 다음을 추가한다.

ViewModels/CartViewModel.cs

|  |
| --- |
| using CommunityToolkit.Mvvm.ComponentModel;  using CommunityToolkit.Mvvm.Input;  using IcecreamShop.APP.Models;  using IcecreamShop.APP.Pages;  using IcecreamShop.APP.Services;  using IcecreamShop.Shared.Dtos;  using Refit;  using System.Collections.ObjectModel;  namespace IcecreamShop.APP.ViewModels;  public partial class CartViewModel : BaseViewModel  {  private readonly DatabaseService \_databaseService;  private readonly IOrderApi \_orderApi;  private readonly AuthService \_authService;  public CartViewModel(DatabaseService databaseService, IOrderApi orderApi, AuthService authService)  {  \_databaseService = databaseService;  \_orderApi = orderApi;  \_authService = authService;  }  public ObservableCollection<CartItem> CartItems { get; set; } = [];  public static int TotalCartCount { get; set; }  public static event EventHandler<int>? TotalCartCountChanged;  public async void AddItemToCart(IcecreamDto icecream, int quantity, string flavor, string topping)  {  var existingItem = CartItems.FirstOrDefault(ci => ci.IcecreamId == icecream.Id);  if ((existingItem is not null))  {  var dbCartItem = await \_databaseService.GetCartItemAsync(existingItem.Id);  if (quantity <= 0)  {  await \_databaseService.DeleteCartItem(dbCartItem);  CartItems.Remove(existingItem);  await ShowToastAsync("Icecream removed from the cart");  }  else  {    dbCartItem.Quantity = quantity;  await \_databaseService.UpdateCartItem(dbCartItem);  existingItem.Quantity = quantity;  await ShowToastAsync("Quantity updated in the cart");  }  }  else  {  var cartItem = new CartItem  {  IcecreamId = icecream.Id,  Id = icecream.Id,  Name = icecream.Name,  Price = icecream.Price,  Quantity = quantity,  FlavorName = flavor,  ToppingName = topping,  };    // sqlite 에 기록  var entity = new Data.CartItemEntity(cartItem);  await \_databaseService.AddCartItem(entity);  // sqlite id를 저장.  cartItem.Id = entity.Id;  CartItems.Add(cartItem);  await ShowToastAsync("Icecream added to cart");  }  NotifyCartCountChange();  }  private void NotifyCartCountChange()  {  TotalCartCount = CartItems.Sum(i => i.Quantity);  TotalCartCountChanged?.Invoke(null, TotalCartCount);  }  public int GetItemCartCount(int icecreamId)  {  var existingItem = CartItems.FirstOrDefault(i => i.IcecreamId == icecreamId);  return existingItem?.Quantity?? 0;  }  public async Task InitializeCartAsync()  {  var dbItems = await \_databaseService.GetAllCartItemAsync();  foreach (var dbItem in dbItems)  {  CartItems.Add(dbItem.ToCartItemModel());  }  NotifyCartCountChange();  }  [RelayCommand]  private async Task ClearCartAsync()  {  await ClearCartInternalAsync(fromPlaceOrder:false);  }  private async Task ClearCartInternalAsync(bool fromPlaceOrder)  {  if (!fromPlaceOrder && CartItems.Count == 0)  {  await ShowAlertAsync("Empty Cart", "There's no items in the cart");  return;  }  if (fromPlaceOrder  || await ConfirmAsync("Clear Cart?", "Do you really want to clear all the items from the cart"))  {  await \_databaseService.ClearCartAsync();  CartItems.Clear();  if(!fromPlaceOrder)  await ShowToastAsync("Cart cleared");  NotifyCartCountChange();  }  }  [RelayCommand]  private async Task RemoveCartItemAsync(int cartItemId)  {  if (await ConfirmAsync("Remove item from Cart?", "Do you really want to delete this item items from the cart"))  {  var existingItem = CartItems.FirstOrDefault(i => i.Id == cartItemId);  if ((existingItem is null))  {  return;  }  CartItems.Remove(existingItem);  var dbCartItem = await \_databaseService.GetCartItemAsync(existingItem.Id);  if ((dbCartItem is null))  {  return;  }  await \_databaseService.DeleteCartItem(dbCartItem);  await ShowToastAsync("Icecream removed from cart");  NotifyCartCountChange();  }  }  [RelayCommand]  private async Task PlaceOrderAsync()  {  if(CartItems.Count == 0)  {  await ShowAlertAsync("Empty Cart", "Please, Add some items to cart");  return;  }  IsBusy = true;  try  {  var order = new OrderDto(0, DateTime.Now, CartItems.Sum(i => i.TotalPrice));  var orderItems = CartItems.Select(i => new OrderItemDto(0, i.IcecreamId,i.Name, i.Quantity, i.Price, i.FlavorName, i.ToppingName)).ToArray();  var orderPlaceDto = new OrderPlaceDto(order, orderItems);  var result = await \_orderApi.PlaceOrderAsync(orderPlaceDto);  if (!result.IsSuccess)  {  await ShowErrorAlertAsync(result.ErrorMessage!);  return;  }  // 성공이라면  // 카드 지우기  // 성공 메시지 보여주기.  await ShowToastAsync("Order placed successful!!");  await ClearCartInternalAsync(fromPlaceOrder: true);  }  catch (ApiException ex)  {  if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  {  // User is not logged in  await ShowErrorAlertAsync("Session Expired");  \_authService.Signout();  await GotoAsync($"//{nameof(OnboardingPage)}", animate:false);  return;  }  await ShowErrorAlertAsync(ex.Message);    }  finally  {  IsBusy = false;  }  }  } |

Page/CartPage.xaml

|  |
| --- |
| </ScrollView>  <Button Text="Place Order"  VerticalOptions="End"  HorizontalOptions="Center"  Command="{Binding PlaceOrderCommand}"/>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </ContentPage> |

15. 프로파일

1) IcecreamShop.APP 프로젝트에서 ViewModels/ProfileViewModel.cs 추가를 한다.

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class ProfileViewModel : BaseViewModel  {  private readonly AuthService \_authService;  public ProfileViewModel(AuthService authService)  {  \_authService = authService;  }  [ObservableProperty, NotifyPropertyChangedFor(nameof(Initials))]  private string \_name = "";  public string Initials  {  get  {  var parts = Name.Split(' ', StringSplitOptions.TrimEntries | StringSplitOptions.RemoveEmptyEntries);  if(parts.Length > 1)  {  return $"{parts[0][0]}{parts[1][0]}".ToUpper() ;  }  return Name.Length > 1 ? Name[..1]:Name.ToUpper() ;  }  }  public void Initialize()  {  Name = \_authService.User.Name;  }  [RelayCommand]  private async Task SignoutAsync()  {  \_authService.Signout();  await GotoAsync($"//{nameof(OnboardingPage)}", animate:false);  }  } |

MauiProgram.cs

|  |
| --- |
| builder.Services.AddTransient<ProfileViewModel>()  .AddTransient<ProfilePage>(); |

Pages/ProfilePage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class ProfilePage : ContentPage  {  private readonly ProfileViewModel \_profileViewModel;  public ProfilePage(ProfileViewModel profileViewModel)  {  InitializeComponent();  BindingContext = \_profileViewModel = profileViewModel;  }  protected override void OnAppearing()  {  \_profileViewModel.Initialize();  }  } |

Pages/ProfilePage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:ProfileViewModel"  x:Class="IcecreamShop.APP.Pages.ProfilePage"  Title="My Profile">  <ContentPage.Resources>  <ResourceDictionary>  <Style x:Key="MenuLbl" TargetType="Label" >  <Setter Property="FontSize" Value="20"/>  <Setter Property="TextColor" Value="DeepPink"/>  </Style>  </ResourceDictionary>  </ContentPage.Resources>  <Grid RowDefinitions="Auto, \*, Auto"  Padding="15, 25,15,15">  <HorizontalStackLayout Grid.Row="0"  Spacing="15"  HorizontalOptions="Center">  <Border StrokeShape="RoundRectangle 25"  HeightRequest="50"  WidthRequest="50"  BackgroundColor="DeepPink">  <Label Text="{Binding Initials}"  TextColor="White"  HorizontalOptions="Center"  VerticalOptions="Center"  FontAttributes="Bold"  FontSize="25"/>  </Border>  <Label Text="{Binding Name}"  FontSize="25"  FontAttributes="Bold"  VerticalOptions="Center"  TextColor="DeepPink"/>  </HorizontalStackLayout>  <VerticalStackLayout Grid.Row="1" VerticalOptions="Center">  <Label Style="{x:StaticResource MenuLbl}" Text="My Orders"/>  <BoxView HeightRequest="1" Color="Pink" Margin="0,10"/>  <Label Style="{x:StaticResource MenuLbl}" Text="Change Password"/>  <BoxView HeightRequest="1" Color="Pink" Margin="0,10"/>  </VerticalStackLayout>  <Button  Grid.Row="2"  Padding="0"  FontSize="15"  HeightRequest="30"  Text="Signout"  WidthRequest="120"  Command="{Binding SignoutCommand}"/>  </Grid>  </ContentPage> |

16. 프로파일에서 나의 주문 내역 보여주기.

1) IcecreamShop.API 프로젝트에서 OrderService.cs를 추가 한다.

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class OrderService(DataContext context)  **{**  private readonly DataContext \_context = context;  public async Task<ResultDto> PlaceOrderAsync(OrderPlaceDto dto, Guid customerId)  {  var customer = await \_context.Users.FirstOrDefaultAsync(u => u.Id == customerId);  if(customer is null)  {// error - no customer  return ResultDto.Failure("Customer does not exist");  }  var orderItems = dto.Items.Select(i =>  new OrderItem  {  Flavor = i.Flavor,  IcecreamId = i.IcecreamId,  Name = i.Name,  Price = i.Price,  Quantity = i.Quantity,  Topping = i.Topping,  TotalPrice = i.TotalPrice,  });  var order = new Order  {  CustomerId = customerId,  CustomerAddress = customer.Address,  CustomerEmail = customer.Email,  CustomerName = customer.Name,  OrderAt = DateTime.Now,  TotalPrice = orderItems.Sum(o=>o.TotalPrice),  };  try  {  await \_context.Orders.AddAsync(order);  await \_context.SaveChangesAsync();  return ResultDto.Success();  }  catch (Exception ex)  {  return ResultDto.Failure(ex.Message);  }  }  public async Task<OrderDto[]> GetUserOrdersAsync(Guid userId) =>  await \_context.Orders  .Where(o => o.CustomerId == userId)  .Select(o=> new OrderDto(o.Id, o.OrderAt, o.TotalPrice))  .ToArrayAsync();  public async Task<OrderItemDto[]> GetUserOrderItemsAsync(long orderId, Guid userId) =>  await \_context.OrderItems  .Where(i => i.OrderId == orderId && i.Order.CustomerId == userId)  .Select(i => new OrderItemDto(i.Id, i.IcecreamId, i.Name, i.Quantity, i.Price, i.Flavor, i.Topping))  .ToArrayAsync();  **}** |

2) IcecreamShop.API 프로젝트에서 EndPoints/AuthEndpoints.cs를 추가한다.

|  |
| --- |
| namespace IcecreamShop.API.EndPoints;  public static class AuthEndpoints  {  private static Guid GetUserId(this ClaimsPrincipal principal) =>  Guid.Parse(principal.FindFirstValue(ClaimTypes.NameIdentifier)!);  public static IEndpointRouteBuilder MapEndPoints(this IEndpointRouteBuilder app)  {    === 중간 생략 ===  orderGroup.MapGet("",  async (ClaimsPrincipal principal, OrderService orderService) =>  TypedResults.Ok(await orderService.GetUserOrdersAsync(principal.GetUserId())));  orderGroup.MapGet("/{orderId:long}/items",  async(long orderId, ClaimsPrincipal principal, OrderService orderService)=>  TypedResults.Ok(await orderService.GetUserOrderItemsAsync(orderId, principal.GetUserId()))  );  return app;  }  } |

3) IcecreamShop.APP 프로젝트에서 Services/IOrderApi.cs에 추가를 한다.

Services/IOrderApi.cs

|  |
| --- |
| namespace IcecreamShop.APP.Services;  [Headers("Authorization Bearer")]  public interface IOrderApi  **{**  [Post("/api/order/place-order")]  Task<ResultDto> PlaceOrderAsync(OrderPlaceDto dto);  [Get("/api/orders")]  Task<OrderDto[]> GetMyOrdersAsync();  [Get("/api/orders/{orderId}/items")]  Task<OrderItemDto[]> GetUserOrderItemsAsync(long orderId);  **}** |

4) IcecreamShop.APP 프로젝트에서 ViewModels/OrderViewModel.cs를 추가한다.

ViewModels/OrderViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class OrderViewModel : BaseViewModel  {  private readonly AuthService \_authService;  private readonly IOrderApi \_orderApi;  public OrderViewModel(AuthService authService, IOrderApi orderApi)  {  \_authService = authService;  \_orderApi = orderApi;  }  [ObservableProperty]  private OrderDto[] \_orders = [];  public async Task InitialzeAsync() => await LoadOrderAsync();  [RelayCommand]  private async Task LoadOrderAsync()  {  IsBusy = true;  try  {  Orders = await \_orderApi.GetMyOrdersAsync();  if (Orders.Length == 0)  {  await ShowToastAsync("No orders found");  }  }  catch (ApiException ex)  {  await HandleApiExceptionAsync(ex, () => \_authService.Signout());  }  finally  {  IsBusy = false;  }  }  } |

5) ViewModels/BaseViewModel.cs를 수정한다.

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class BaseViewModel : ObservableObject  {  === 중간 생략 ===  protected async Task HandleApiExceptionAsync(ApiException ex, Action? signout)  {  if (ex.StatusCode == System.Net.HttpStatusCode.Unauthorized)  {  // User is not logged in  await ShowErrorAlertAsync("Session Expired");  //\_authService.Signout();  signout?.Invoke();  await GotoAsync($"//{nameof(OnboardingPage)}", animate: false);  return;  }  await ShowErrorAlertAsync(ex.Message);  }  } |

5) Pages/MyOrdersPage.xaml.cs

|  |
| --- |
| public partial class MyOrdersPage : ContentPage  {  private readonly OrderViewModel \_orderViewModel  public MyOrdersPage(OrderViewModel orderViewModel)  {  InitializeComponent();  BindingContext = \_orderViewModel = orderViewModel;  }  protected override async void OnAppearing()  {  await \_orderViewModel.InitialzeAsync();  }  } |

6) MauiProgram.cs

|  |
| --- |
| builder.Services.AddTransient<OrderViewModel>()  .AddTransient<MyOrdersPage>(); |

7) ViewModels/ProfileViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class ProfileViewModel : BaseViewModel  {  ==== 생략 ====  [RelayCommand]  private async Task SignoutAsync()  **{**  \_authService.Signout();  await GotoAsync($"//{nameof(OnboardingPage)}", animate:false);  **}**  [RelayCommand]  private async Task GotoMyOrdersAsync() =>  await GotoAsync(nameof(MyOrdersPage), animate:true);  } |

8) Pages/ProfilePage.xaml에 일부 코드 수정

|  |
| --- |
| <Label Style="{x:StaticResource MenuLbl}" Text="My Orders">  <Label.GestureRecognizers>  <TapGestureRecognizer Command="{Binding GotoMyOrdersCommand}"/>  </Label.GestureRecognizers>  </Label> |

9) 주문한 아이템 개수를 가져오기 위해서 다음을 수정한다.

\* IcecreamShop.Shared프로젝트에서 Dtos/OrderDto.cs를 다음과 같이 수정한다.

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record OrderItemDto(long Id, int IcecreamId, string Name, int Quantity, double Price, string Flavor, string Topping)  {  public double TotalPrice => Quantity \* Price;  }  public record OrderPlaceDto(OrderDto Order, OrderItemDto[] Items);  public record OrderDto(long Id, DateTime OrderAt, double TotalPrice, int ItemsCount = 0)  {  public string ItemCountDisplay => ItemsCount + (ItemsCount > 1 ? " Items" : " Items");  } |

\*IcecreamShop.API 프로젝트에서 Services/OrderServices.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class OrderService(DataContext context)  {  === 중간 생략 ===  public async Task<OrderDto[]> GetUserOrdersAsync(Guid userId) =>  await \_context.Orders  .Where(o => o.CustomerId == userId)  .Select(o=> new OrderDto(o.Id, o.OrderAt, o.TotalPrice, o.Items.Count))  .ToArrayAsync();  === 이하 생략 ===  } |

10) Pages/MyOrderPage.xaml

Pages/MyOrderPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:OrderViewModel"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="IcecreamShop.APP.Pages.MyOrdersPage"  Title="My Orders"  BackgroundColor="#EEE">  <RefreshView Command="{Binding LoadOrderCommand}" IsRefreshing="{Binding *IsBusy*}">  <Grid>  <ScrollView Padding="10">  <CollectionView ItemsSource="{Binding Orders}">  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="dto:OrderDto">  <Border  StrokeShape="RoundRectangle 10"  Background="White"  Margin="0, 4"  Padding="10">  <Grid ColumnDefinitions="\*, Auto, Auto"  ColumnSpacing="10">  <Grid Grid.Column="0" RowDefinitions="\*, Auto">  <Label Grid.Row ="0"  Text="{Binding Id, *StringFormat*='Order Id: {0}'}"  FontSize="15"  TextColor="DeepPink"/>  <Label Grid.Row="1"  Text="{Binding ItemCountDisplay}"/>  </Grid>  <Grid Grid.Column="1" RowDefinitions="\*, \*"  RowSpacing="2">  <Label Grid.Row ="0"  Text="{Binding TotalPrice, *StringFormat*='{0:C}'}"  FontSize="15"  TextColor="DeepPink"  FontAttributes="Bold"  HorizontalOptions="End"/>  <Label Grid.Row="1"  FontSize="13"  Text="{Binding OrderAt, *StringFormat*='{0: MMM dd yyyy, hh:mm'}"/>  </Grid>  <Image Grid.Column="2"  Source="right\_arrow"  VerticalOptions="Center">  <Image.Behaviors>  <toolkit:IconTintColorBehavior TintColor="DeepPink"/>  </Image.Behaviors>  </Image>  </Grid>  </Border>    </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </ScrollView>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </RefreshView>  </ContentPage> |

17. 주문 내역 보기.

1) ViewModels/OrderDetailViewModel.cs를 추가한다.

ViewModels/OrderDetailViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  [QueryProperty(nameof(OrderId), nameof(OrderId))]  public partial class OrderDetailViewModel : BaseViewModel  {  private readonly AuthService \_authService;  private readonly IOrderApi \_orderApi;  public OrderDetailViewModel(AuthService authService, IOrderApi orderApi)  {  \_authService = authService;  \_orderApi = orderApi;  }  [ObservableProperty]  private long \_orderId;  [ObservableProperty]  private OrderItemDto[] \_orderItems = [];  [ObservableProperty]  private string \_title = "Order Items";  partial void OnOrderIdChanged(long value)  {  Title = $"Order #{value}";  LoadOrderItemsAsync(value);  }  private async Task LoadOrderItemsAsync(long orderId)  {  IsBusy = true;  try  {  OrderItems = await \_orderApi.GetUserOrderItemsAsync(orderId);  }  catch (ApiException ex)  {  await HandleApiExceptionAsync(ex, () => \_authService.Signout());  }  finally  {  IsBusy = false;  }  }  } |

2) Pages/OrderDetailPage.xaml.cs에 ViewModel 적용.

Pages/OrderDetailPage.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Pages;  public partial class OrderDetailPage : ContentPage  {  private readonly OrderDetailViewModel \_orderDetailViewModel;  public OrderDetailPage(OrderDetailViewModel orderDetailViewModel)  {  InitializeComponent();  BindingContext = \_orderDetailViewModel = orderDetailViewModel;  }  } |

3) 서비스 등록

MauiProgram.cs

|  |
| --- |
| builder.Services.AddTransient<OrderDetailViewModel>()  .AddTransient**<**OrderDetailPage**>**(); |

4) ViewModels/OrderViewModel.cs에 다음을 추가

ViewModels/OrderViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class OrderViewModel : BaseViewModel  {  === 중간 생략 ===  [RelayCommand]  private async Task GoToOrderDetailsPageAsync(long orderId)  {  var parameter = new Dictionary<string, object>  {  [nameof(OrderDetailViewModel.OrderId)] = orderId,  };  await GotoAsync(nameof(OrderDetailPage), animate: true, parameter);  }  } |

5) 항목이 클릭하면 주면 내역페이지가 open되도록 Pages/MyOrderPage.xaml에 코드 추가.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:OrderViewModel"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  x:Class="IcecreamShop.APP.Pages.MyOrdersPage"  Title="My Orders"  BackgroundColor="#EEE">  <RefreshView Command="{Binding LoadOrderCommand}" IsRefreshing="{Binding *IsBusy*}">  <Grid>  <ScrollView Padding="10">  <CollectionView ItemsSource="{Binding Orders}">  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="dto:OrderDto">  <Border  StrokeShape="RoundRectangle 10"  Background="White"  Margin="0, 4"  Padding="10">    === 중 간 생 략 ===  </Grid>  <Border.GestureRecognizers>  <TapGestureRecognizer Command="{Binding *Source*={x:RelativeSource *AncestorType*={x:Type vm:OrderViewModel}}, *Path*=GoToOrderDetailsPageCommand}"  CommandParameter="{Binding Id}"/>  </Border.GestureRecognizers>  </Border>    </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </ScrollView>  <ActivityIndicator IsRunning="{Binding *IsBusy*}"/>  </Grid>  </RefreshView>  </ContentPage> |

6) UI를 위해서 Pages/OrderDetailPage.xaml 에 다음을 추가

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  x:Class="IcecreamShop.APP.Pages.OrderDetailPage"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:OrderDetailViewModel"  Title="{Binding *Title*}"  BackgroundColor="#EEE">  <Grid>  <ScrollView Padding="10">  <CollectionView ItemsSource="{Binding OrderItems}">  <CollectionView.ItemTemplate>  <DataTemplate x:DataType="dto:OrderItemDto">  <StackLayout BackgroundColor="DeepPink"  Padding="0,0,0,1"  Margin="0, 5">  <StackLayout BackgroundColor="White"  Padding="10">  <Grid ColumnDefinitions="40, \*, Auto">  <Label Grid.Column="0"  Text="{Binding Quantity}"  VerticalOptions="Center"  FontSize="16"  FontAttributes="Bold"/>  <VerticalStackLayout>  <Label Text="{Binding Name}"  FontSize="15"  TextColor="DeepPink"/>  <Label FontSize="12">  <Label.FormattedText>  <FormattedString>  <Span Text="{Binding Flavor}" TextColor="#333"/>  <Span Text=", "/>  <Span Text="{Binding Topping}" TextColor="#333"/>  </FormattedString>  </Label.FormattedText>  </Label>  </VerticalStackLayout>  <Label Grid.Column="2"  Text="{Binding TotalPrice, *StringFormat*='{0:C}'}"  VerticalOptions="Center"  FontSize="15"  FontAttributes="Bold"/>  </Grid>  </StackLayout>  </StackLayout>  </DataTemplate>  </CollectionView.ItemTemplate>  </CollectionView>  </ScrollView>  </Grid>  </ContentPage> |

18. 암호 변경

1) IcecreamShop.Shared에 Dtos/ChangePasswordDto.cs를 추가한다.

Dtos/ChangePasswordDto.cs

|  |
| --- |
| namespace IcecreamShop.Shared.Dtos;  public record ChangePasswordDto(string OldPassword, string NewPassword); |

2) IcecreamShop.API 프로젝트에서 Services/AuthService.cs를 열어서 다음과 같이 추가를 한다.

Services/AuthService.cs

|  |
| --- |
| namespace IcecreamShop.API.Services;  public class AuthService(DataContext context, TokenService tokenService, PasswordService passwordService)  {    === 중간 생략 ===  public async Task<ResultDto> ChangePasswordAsync(ChangePasswordDto dto, Guid userId)  {  var user = await \_context.Users.FirstOrDefaultAsync(u => u.Id == userId);  if (user is null)  return ResultDto.Failure("Invalid request");  if(!\_passwordService.AreEqual(dto.OldPassword, user.Salt, user.Hash))  {  return ResultDto.Failure("Incorrect password");  }  (user.Salt, user.Hash) = \_passwordService.GenerateSaltAndHash(dto.NewPassword);  await \_context.SaveChangesAsync();  return ResultDto.Success();  }  } |

3) IcecreamShop.API 프로젝트에서 Endpoints/AuthEndpoints.cs에 다음과 같이 추가를 한다.

|  |
| --- |
| namespace IcecreamShop.API.EndPoints;  public static class AuthEndpoints  {  private static Guid GetUserId(this ClaimsPrincipal principal) =>  Guid.Parse(principal.FindFirstValue(ClaimTypes.NameIdentifier)!);  public static IEndpointRouteBuilder MapEndPoints(this IEndpointRouteBuilder app)  {  === 중간 생략 ===  app.MapPost("/api/auth/change-password",  async(ChangePasswordDto dto, ClaimsPrincipal principal, AuthService authService) =>  TypedResults.Ok(await authService.ChangePasswordAsync(dto, principal.GetUserId() )))  .RequireAuthorization();  return app;  }  } |

4) IcecreamShop.APP 프로젝트에서 API를 호출하기 위해서 Services/IAuthApi.cs를 추가한다.

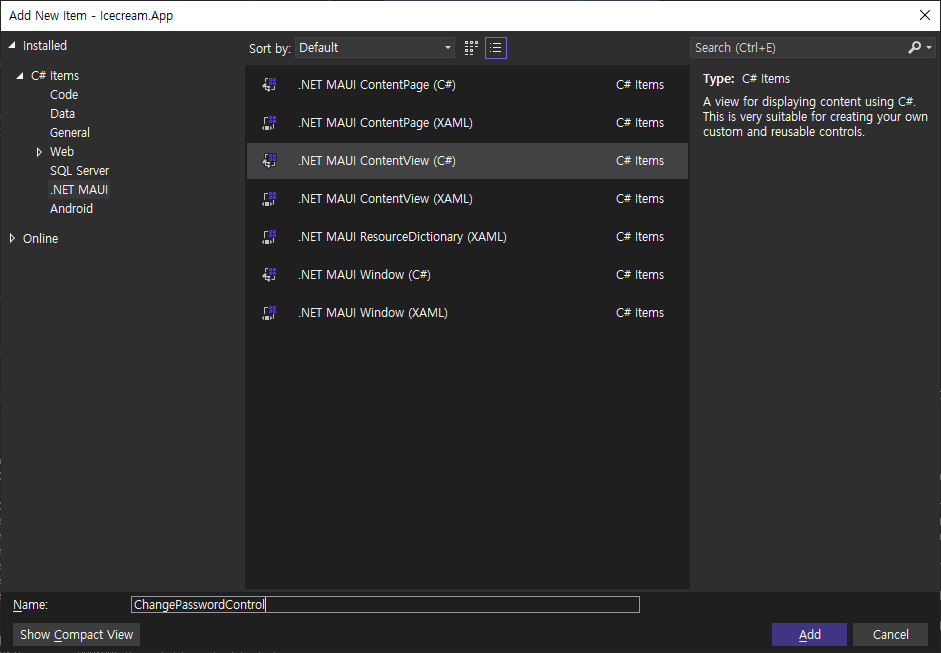
Services/IAuthApi.cs

|  |
| --- |
| namespace IcecreamShop.APP.Services;  public interface IAuthApi  {  === 중간 생 략 ======  [Headers("Authorization: Bearer")]  [Post("/api/auth/change-password")]  Task<ResultDto> ChangePasswordAsync(ChangePasswordDto dto);  } |

5) 암호 변경 ViewModel을 위해서 ViewModels/ChangePasswordViewModel.cs를 새로 생성한다.

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class ChangePasswordViewModel : BaseViewModel  {  private readonly AuthService \_authService;  private readonly IAuthApi \_authApi;  public ChangePasswordViewModel(AuthService authService, IAuthApi authApi)  {  this.\_authService = authService;  this.\_authApi = authApi;  }  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanChangePassword))]  private string? \_oldPassword;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanChangePassword))]  private string? \_newPassword;  [ObservableProperty, NotifyPropertyChangedFor(nameof(CanChangePassword))]  private string? \_confirmNewPassword;  public bool CanChangePassword => !string.IsNullOrWhiteSpace(OldPassword)  && !string.IsNullOrWhiteSpace(NewPassword)  && !string.IsNullOrWhiteSpace(ConfirmNewPassword);  [RelayCommand]  private async Task ChangePasswordAsync()  {  if (NewPassword != ConfirmNewPassword)  {  await ShowErrorAlertAsync("New password and confirm new password do not match");  return;  }  IsBusy = true;  try  {  var dto = new ChangePasswordDto(OldPassword!, NewPassword!);  var result = await \_authApi.ChangePasswordAsync(dto);  if (!result.IsSuccess)  {  await ShowErrorAlertAsync(result.ErrorMessage!);  return;  }  await ShowAlertAsync("Success", "Password changed successfully");  OldPassword = NewPassword = ConfirmNewPassword = null;  }  catch (ApiException ex)  {  await HandleApiExceptionAsync(ex, () => \_authService.Signout());  }  finally  {  IsBusy = false;  }  }  } |

6) 패스워드 변경은 팝업으로 하기 위해서 IcecreamShop.APP 프로젝트에서 IcecreamShop.APP에 Controls폴더를 추가하고 Add => New Item => .NET MAUI ContentView(C#)을 선택한다.



이름을 ChangePasswordControl이라고 한다.

Controls/ChangePasswordControl.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <toolkit:Popup xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:toolkit="http://schemas.microsoft.com/dotnet/2022/maui/toolkit"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:ChangePasswordViewModel"  x:Class="IcecreamShop.APP.Controls.ChangePasswordControl"  >  <Grid RowDefinitions="Auto, Auto" RowSpacing="10" BackgroundColor="White">  <Grid Grid.Row="0"  Padding="10, 5"  BackgroundColor="DeepPink"  ColumnDefinitions="\*, Auto"  HeightRequest="40">  <Label Grid.Column="0" Text="Change Password"  TextColor="White"  FontSize="18"  FontAttributes="Bold"  HorizontalOptions="Start"  VerticalOptions="Center"/>  <Label Grid.Column="2" Text="X"  BackgroundColor="White"  Padding="10, 5"  TextColor="Red"  FontSize="15"  FontAttributes="Bold"  HorizontalOptions="End"  VerticalOptions="Center">  <Label.GestureRecognizers>  <TapGestureRecognizer Tapped="ClosePopup\_Tapped"/>  </Label.GestureRecognizers>  </Label>  </Grid>  <VerticalStackLayout Grid.Row="1"  Padding="10"  Spacing="10"  VerticalOptions="Center"  BackgroundColor="White">  <Entry Text="{Binding *OldPassword*}" IsPassword="True" Placeholder="Old Password"/>  <Entry Text="{Binding *NewPassword*}" IsPassword="True" Placeholder="New Password"/>  <Entry Text="{Binding ConfirmNewPassword}" IsPassword="True" Placeholder="Confirm Password"/>  <Button Text="Change Password" Command="{Binding ChangePasswordCommand}" IsEnabled="{Binding CanChangePassword}"/>  </VerticalStackLayout>  </Grid>  </toolkit:Popup> |

Controls/ChangePasswordControl.xaml.cs

|  |
| --- |
| namespace IcecreamShop.APP.Controls;  public partial class ChangePasswordControl : Popup  {  private readonly ChangePasswordViewModel \_changePasswordViewModel;  public ChangePasswordControl(ChangePasswordViewModel changePasswordViewModel)  {  InitializeComponent();  BindingContext = this.\_changePasswordViewModel = changePasswordViewModel;  }  private async void ClosePopup\_Tapped(object sender, TappedEventArgs e)  {  await CloseAsync();  }  } |

7) MyProfile과 연결.

Pages/ProfilePage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:vm="clr-namespace:IcecreamShop.APP.ViewModels"  xmlns:models="clr-namespace:IcecreamShop.APP.Models"  xmlns:dto="clr-namespace:IcecreamShop.Shared.Dtos;assembly=IcecreamShop.Shared"  x:DataType="vm:ProfileViewModel"  x:Class="IcecreamShop.APP.Pages.ProfilePage"  Title="My Profile">  === 중간 생략 ===  <VerticalStackLayout Grid.Row="1" VerticalOptions="Center">  <Label Style="{x:StaticResource MenuLbl}" Text="My Orders">  <Label.GestureRecognizers>  <TapGestureRecognizer Command="{Binding GotoMyOrdersCommand}"/>  </Label.GestureRecognizers>  </Label>  <BoxView HeightRequest="1" Color="Pink" Margin="0,10"/>  <Label Style="{x:StaticResource MenuLbl}" Text="Change Password">  <Label.GestureRecognizers>  <TapGestureRecognizer Command="{Binding ChangePasswordCommand}"/>  </Label.GestureRecognizers>  </Label>  <BoxView HeightRequest="1" Color="Pink" Margin="0,10"/>  </VerticalStackLayout>  <Button  Grid.Row="2"  Padding="0"  FontSize="15"  HeightRequest="30"  Text="Signout"  WidthRequest="120"  Command="{Binding SignoutCommand}"/>  </Grid>  </ContentPage> |

8) ViewModels/ProfileViewModel.cs

|  |
| --- |
| namespace IcecreamShop.APP.ViewModels;  public partial class ProfileViewModel : BaseViewModel  {  === 중간 생략 ===  [RelayCommand]  private async Task ChangePasswordAsync()  {  await Shell.Current.CurrentPage.ShowPopupAsync(new ChangePasswordControl(\_changePasswordViewModel));  }  //await GotoAsync($"//{nameof()}")  } |

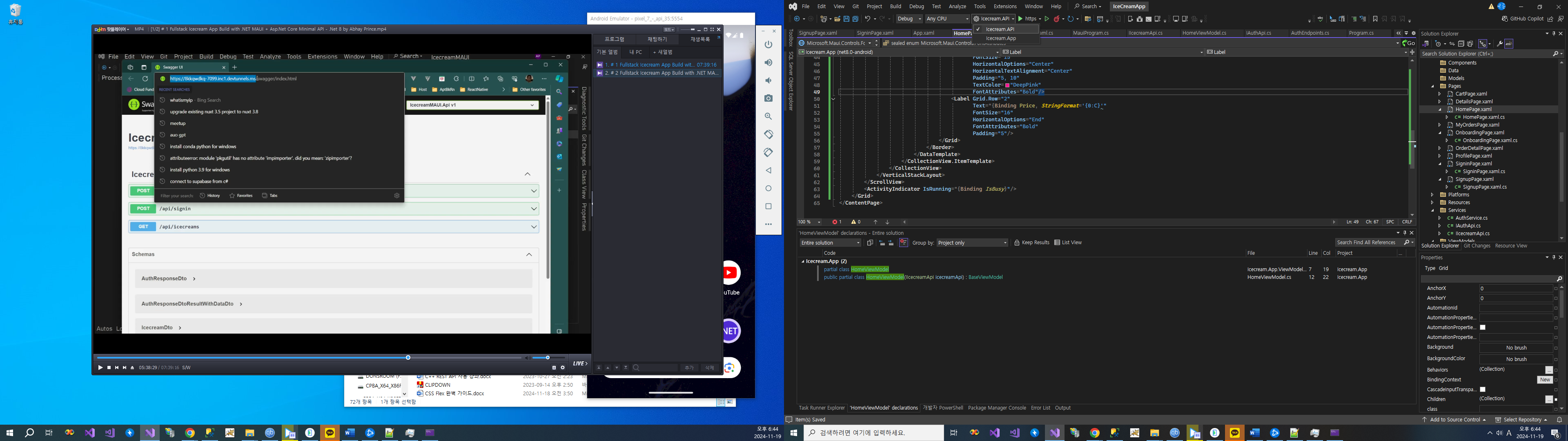
9) 서비스 등록

MauiProgram.cs

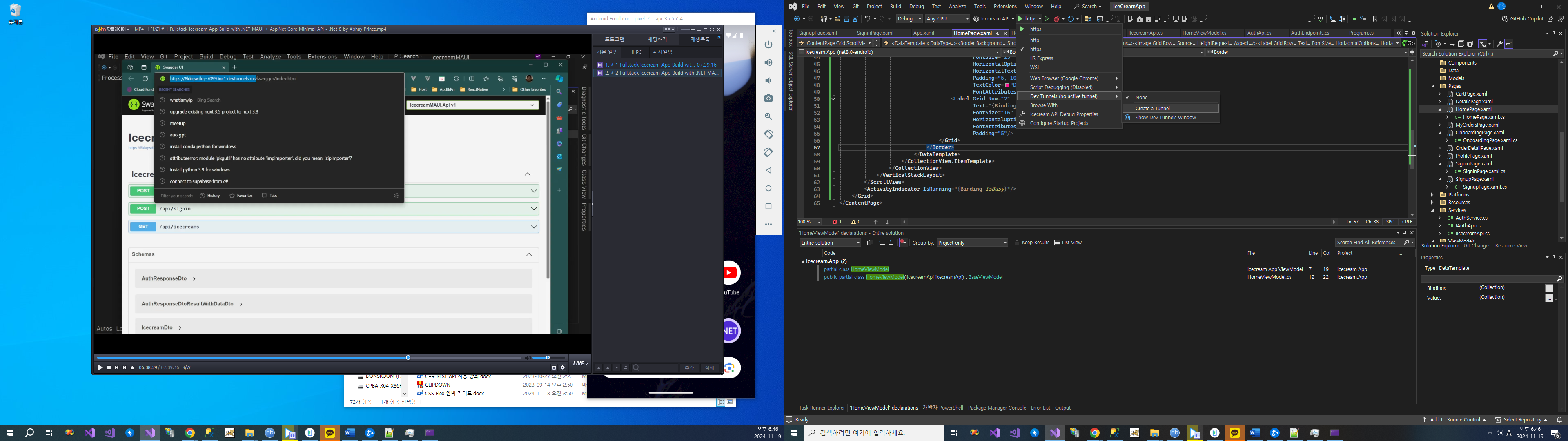
|  |
| --- |
| builder.Services.AddTransient<ChangePasswordViewModel>(); |

Appendix I. DevTunnel 사용하기.

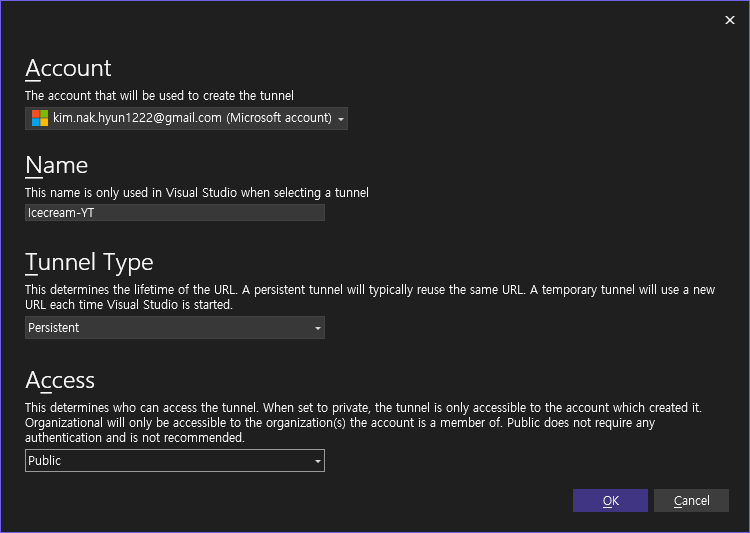
1) 프로젝트를 IcecreamShop.API로 선택한다.

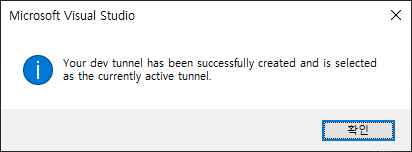


2) 실행 타켓을 Dev Tunnels(no active tunnel) => Create a Tunnel를 선택한다.



3) 아래와 같이 dev tunnel를 생성한다.





4) 터널 주소명으로

