Test admin Spectator

Made by:

Diego Alexander Torres Forero

Software Engineering Specialist

Delivered to:

SPECTATOR

Summary

This development consists of a mobile application connected to Google's cloud service, "Google Fit." This application aims to display three key metrics: the number of steps, the distance traveled, and the calories burned. It implements OAuth 2.0 authentication provided by these Google services. The application is developed using Xamarin Forms.

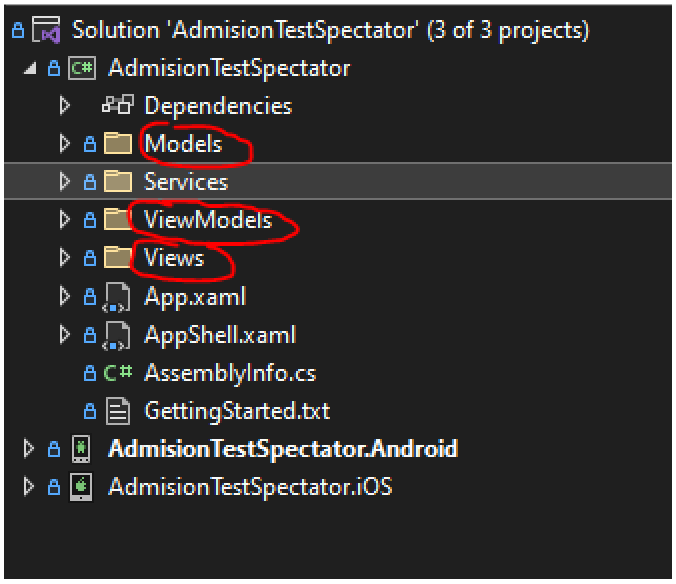
Architecture:

The application was developed using the Xamarin Forms MVVM architecture template as shown in Image 1.

Model: Contains data and business logic.

View: Is the user interface, usually defined in XAML.

ViewModel: Acts as a bridge between the Model and the View, facilitating mechanisms for notifications of data changes.



```
Solution 'AdmisionTestSpectator' (3 of 3 projects)
  AdmisionTestSpectator
    Dependencies
    Models
    Services
    ViewModels
    Views
    App.xaml
    AppShell.xaml
    AssemblyInfo.cs
    GettingStarted.txt
  AdmisionTestSpectator.Android
  AdmisionTestSpectator.iOS
```

Image 1 (Application Architecture)

Layer-Based Architecture

The architecture also follows a layered approach, thereby separating the logic into a library using .Net Standard 2.0. Configurations for Android are maintained in a separate layer, and the configuration for iOS is similarly kept in another layer as illustrated in Image 2.

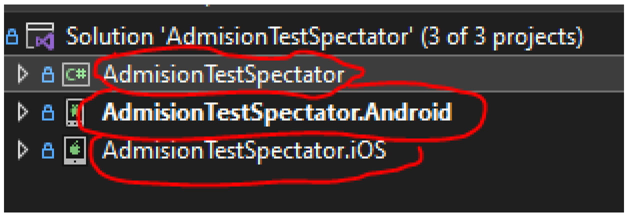
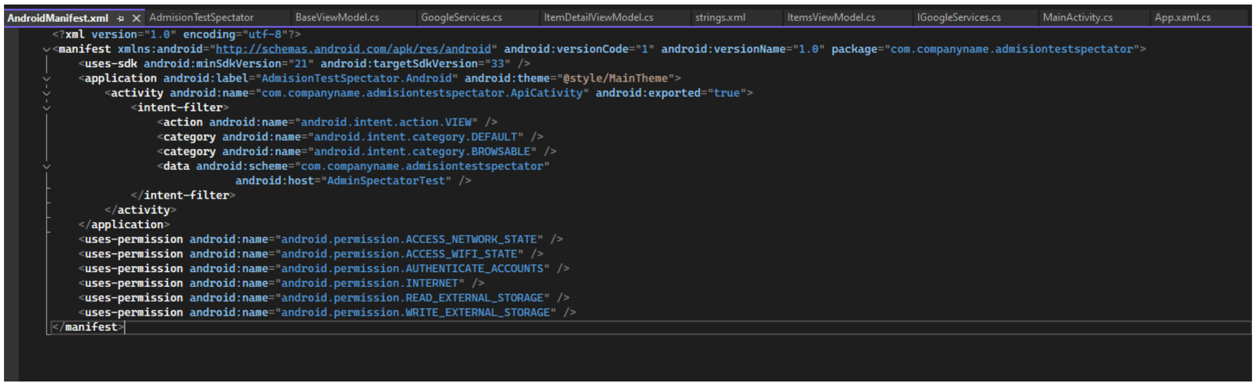

Solution 'AdmisionTestSpectator' (3 of 3 projects)
AdmisionTestSpectator
AdmisionTestSpectator.Android
AdmisionTestSpectator.iOS

Image 2 (Layer-Based Architecture)

Services

To retrieve data from Google and enable authentication, the following configurations were implemented:

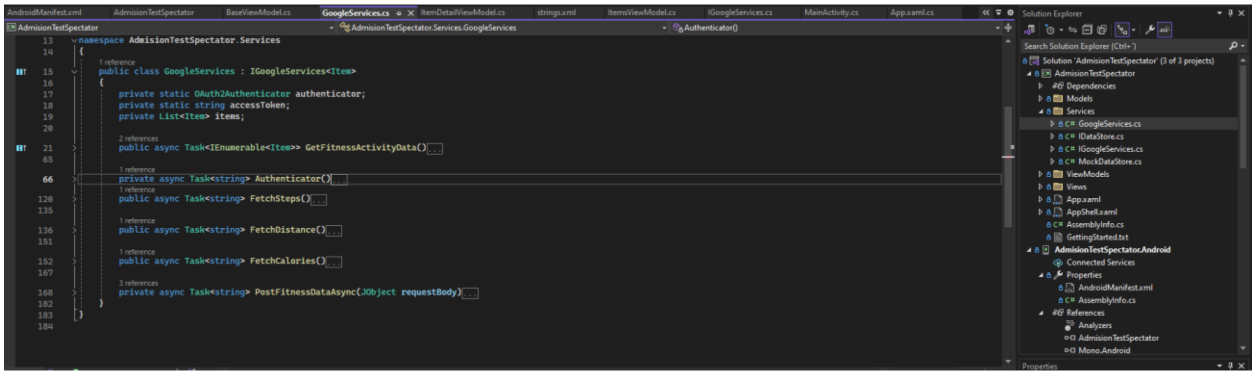
1. **Configure the AndroidManifest.xml File:** In this file, the necessary permissions were configured on the client's device to access resources such as Wi-Fi and internet, among others. The redirection for accessing Google authentication was also configured as illustrated in Image 3.



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="com.companyname.admisiontestspectator">
    <uses-sdk android:minSdkVersion="21" android:targetSdkVersion="33" />
    <application android:label="AdmisionTestSpectator.Android" android:theme="@style/MainTheme">
        <activity android:name="com.companyname.admisiontestspectator.ApiCativity" android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="com.companyname.admisiontestspectator"
                      android:host="AdminSpectatorTest" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.AUTHENTICATE_ACCOUNTS" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
</manifest>
```

Image 3: Manifest Configuration

2. **Configuring the Service for Authentication and Data Retrieval:** The service for authentication and for retrieving data from Google was set up in a C# class as shown in Image 4.

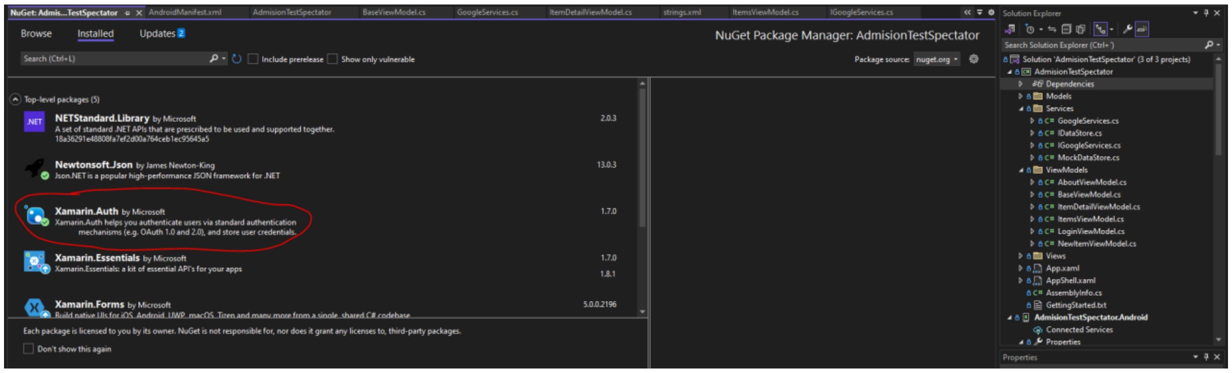


```csharp
namespace AdmisionTestSpectator.Services
{
    1 reference
    public class GoogleServices : IGoogleServices<Item>
    {
        private static OAuth2Authenticator authenticator;
        private static string accessToken;
        private List<Item> items;

        2 references
        public async Task<IEnumerable<Item>> GetFitnessActivityData()[...]

        1 reference
        private async Task<string> Authenticator()[...]
        1 reference
        public async Task<string> FetchSteps()[...]

        1 reference
        public async Task<string> FetchDistance()[...]

        1 reference
        public async Task<string> FetchCalories()[...]

        3 references
        private async Task<string> PostFitnessDataAsync(JObject requestBody)[...]
    }
}
```

Image 5: Service Configuration for Accessing Google Fit

For authentication with Google Cloud, the Xamarin.Auth NuGet package was used to enable OAuth 2.0 authentication.

Additional external libraries such as Newtonsoft.Json were utilized for serializing and deserializing JSON objects, as shown in Image 6.



Top-level packages (5)

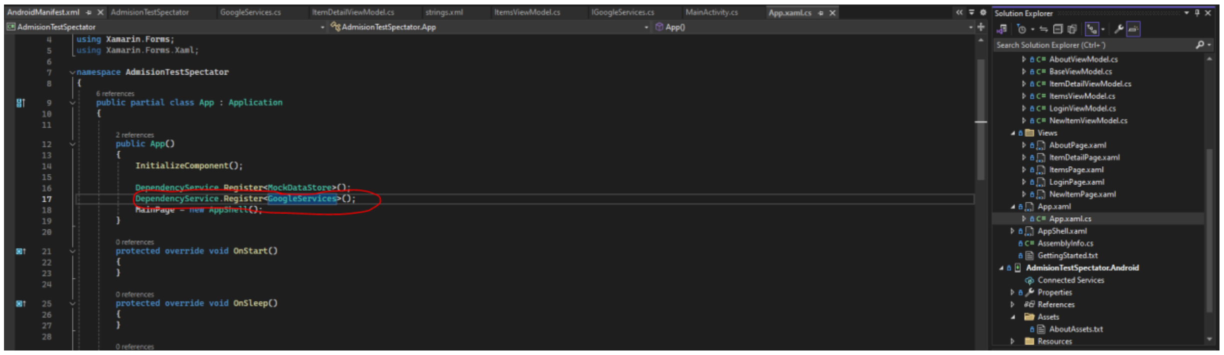
NETStandard.Library by Microsoft A set of standard .NET APIs that are prescribed to be used and supported together. 18a36291e48808fa7ef2d00a764ceb1ec95645a5	2.0.3
Newtonsoft.Json by James Newton-King Json.NET is a popular high-performance JSON framework for .NET	13.0.3
Xamarin.Auth by Microsoft Xamarin.Auth helps you authenticate users via standard authentication mechanisms (e.g. OAuth 1.0 and 2.0), and store user credentials.	1.7.0
Xamarin.Essentials by Microsoft Xamarin.Essentials: a kit of essential API's for your apps	1.7.0 1.8.1
Xamarin.Forms by Microsoft Build native UIs for iOS, Android, UWP, macOS, Tizen and many more from a single, shared C# codebase	5.0.0.2196

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

Don't show this again

Image 6: NuGet Configuration

3. Lastly, for the usage of the service, it was configured in the App.xaml.cs file to ensure that the library could recognize the service, thereby enabling the use of its logic as depicted in Image 7.

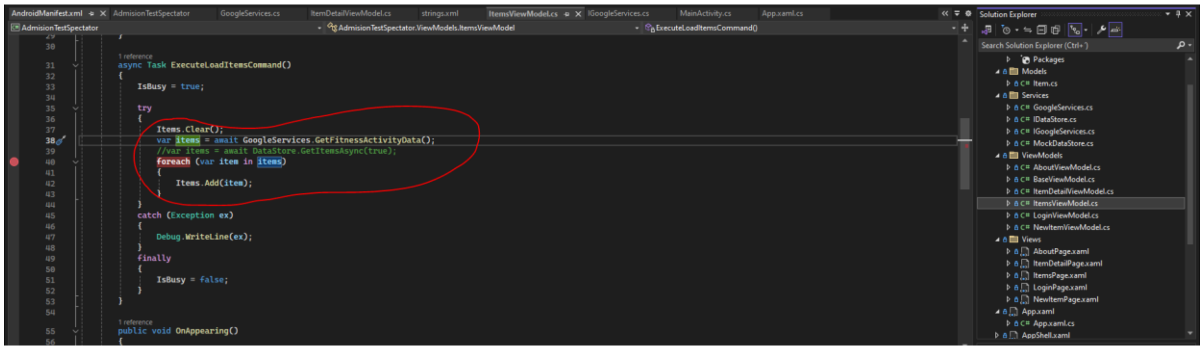


```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace AdmisionTestSpectator
{
    6 references
    public partial class App : Application
    {

        2 references
        public App()
        {
            InitializeComponent();

            DependencyService.Register<MockDataStore>();
            DependencyService.Register<GoogleServices>();
            MainPage = new AppShell();
        }

        0 references
        protected override void OnStart()
        {
        }

        0 references
        protected override void OnSleep()
        {
        }

        0 references
```

Image 7: Service Configuration with Access Logic for Google Fit

ViewModel Configuration

After configuring the service, the ViewModel was set up to display the records in the view as illustrated in Image 8.



```
        1 reference
        async Task ExecuteLoadItemsCommand()
        {
            IsBusy = true;

            try
            {
                Items.Clear();
                var items = await GoogleServices.GetFitnessActivityData();
                //var items = await DataStore.GetItemsAsync(true);
                foreach (var item in items)
                {
                    Items.Add(item);
                }
            }
            catch (Exception ex)
            {
                Debug.WriteLine(ex);
            }
            finally
            {
                IsBusy = false;
            }
        }

        1 reference
        public void OnAppearing()
        {
```

Image 8: ViewModel Configuration