Aplikacje mobilne dla systemu iOS

"Weather"

Jakub Karmański

rok 3 semestr 6 - letni
Wydział Matematyki Stosowanej
kierunek Informatyka

Spis treści:

Opis systematycznych testów

1. MainPage.xaml.cs

Przeprowadzono testy sprawdzające istnienie/zawartość klucza posiadającego ostatnio wybrane miasto:

```
if (Application.Current.Properties.ContainsKey("city"))
{
    var city = Application.Current.Properties["city"] as string;
    OnGetWeather(city);
}
```

Przeprowadzono testy: sprawdzające czy zawartość wprowadzana nie jest pusta lub nie zawiera tzw. białych znaków, sprawdzające czy odpowiedź z serwera nie pojawiła się ze względu na brak połączenia z internetem:

```
WeatherDataForecast weatherData = await _restService.GetForecastData(GenerateRequestUri(Constants.OpenWeatherMapEndpointForecast, city));
if (
{ class WeatherApp.WeatherDataForecast
    this.DisplayAlert("", "No internet connection", "OK");
}
else
{
    weatherData.List[0].Main.Temp = weatherData.List[0].Main.Temp + "°C ";
    weatherImage.Source = "http://openweathermap.org/img/w/" + weatherData.List[0].Weather[0].Icon + ".png";
    Day_0.Text = System.DateTime.Now.AddDays(1).DayOfWeek.ToString();
    Day_1.Text = System.DateTime.Now.AddDays(2).DayOfWeek.ToString();
    Day_2.Text = System.DateTime.Now.AddDays(3).DayOfWeek.ToString();
    BindingContext = weatherData;
    Application.Current.Properties["city"] = city;
}
```

w przypadku braku połączenia sieciowego zwrócony zostaje alert.

2. Search.xaml.cs

Przeprowadzono testy: sprawdzające czy zawartość wprowadzana nie jest pusta lub nie zawiera tzw. białych znaków, sprawdzające czy odpowiedź z serwera nie pojawiła się ze względu na brak połączenia z internetem:

```
if (!string.IsNullOrWhiteSpace(filter))
{
    WeatherData weatherData = await _restService.GetWeatherData(GenerateRequestUri(Constants.OpenWeatherMapEndpoint, filter));
    if (weatherData == null)
    {
        this.DisplayAlert("", "No internet connection", "OK");
    }
    else
    {
        Cities = new List<WeatherData>();
        if (weatherData != null)
        {
            Cities.Add(new WeatherData { Title = weatherData.Title });
            listOfCities.ItemsSource = Cities;
            listOfCities.ItemTapped += CityTapped;
            BindingContext = weatherData;
        }
    }
}
listOfCities.EndRefresh();
```

Przeprowadzono również testy podczas dodawania do listy wyszukiwanych miast czy: lista miast istnieje, oraz czy wybrane miasto nie znajduje się już w liście:

```
if (Application.Current.Properties.ContainsKey("list"))
{
    var list = Application.Current.Properties["list"] as string;
    List<string> listOfCities = JsonConvert.DeserializeObject<List<string>>(list);
    if (listOfCities.Contains(name))
    {
        this.DisplayAlert("", "Item was already in your list", "OK");
    }
    else
    {
        listOfCities.Add(name);
        Application.Current.Properties["list"] = JsonConvert.SerializeObject(listOfCities);
        this.DisplayAlert("", "Item added", "OK");
    }
}
else
{
    List<string> listOfCities = new List<string>();
    listOfCities.Add(name);
    Application.Current.Properties["list"] = JsonConvert.SerializeObject(listOfCities);
    this.DisplayAlert("", "Item addedd", "OK");
}
```

Przeprowadzno testy sprawdzające blokowanie pozawywania się wielu komunikatów/alertów w aplikacji:

```
if (this.disable)
    return;

this.disable = true;
Device.BeginInvokeOnMainThread(async () =>
{
    var result = await this.DisplayAlert("", "Do you want to add " + item.Title + "?", "Yes", "No");
    if (result)
    {
        AddToList(item.Title);
        this.disable = false;
        return;
    }
    else
    {
        this.disable = false;
        return;
    }

});
```

Z racji nasłuchiwania na paru eventach w trakcie wyszukiwania alerty wywoływały się kilkukrotnie (co stwarzało znaczny problem) przez co stworzona musiała być flaga ograniczająca ich pokazywanie. Działanie tej flagi zostało przetestowane.

3. Details.xaml.cs

Przeprowadzono testy sprawdzające istnienie/zawartość klucza posiadającego ostatnio wybrane miasto:

```
if (Application.Current.Properties.ContainsKey("city"))
{
    var cityName = Application.Current.Properties["city"] as string;
    BindingContext = this;
    GetAPIData(cityName);
}
```

Przeprowadzono testy: sprawdzające czy zawartość wprowadzana nie jest pusta lub nie zawiera tzw. białych znaków, sprawdzające czy odpowiedź z serwera nie pojawiła się ze względu na brak połączenia z internetem:

```
if (!string.IsNullOrWhiteSpace(cityName))
{
    WeatherData weatherData = await _restService.GetWeatherData(GenerateRequestUri(Constants.OpenWeatherMapEndpoint, cityName));
    if (weatherData == null)
    {
        this.DisplayAlert("", "No internet connection", "OK");
    }
    else
    {
        weatherImage.Source = "http://openweathermap.org/img/w/" + weatherData.Weather[0].Icon + ".png";
        weatherData.Main.Temperature = weatherData.Main.Temperature + "°C ";
        weatherData.Main.Humidity = weatherData.Main.Humidity + "%";
        weatherData.Main.Pressure = weatherData.Main.Pressure + "hPa";
        BindingContext = weatherData;
    }
}
```

4. CityList.xaml.cs

Przeprowadzono testy sprawdzające istnienie/zawartość klucza posiadającego ostatnio wybrane miasta:

```
if (Application.Current.Properties.ContainsKey("list"))
{
    listOfCities.BeginRefresh();
    string list = Application.Current.Properties["list"] as string;
    List<string> cityList = JsonConvert.DeserializeObject<List<string>>(list);

    foreach (string item in cityList.ToArray())
    {
        Cities.Add(new WeatherData { Title = item });
    }
    listOfCities.ItemsSource = Cities;
    listOfCities.ItemTapped += CityTapped;
    BindingContext = this;
    listOfCities.EndRefresh();
}
```

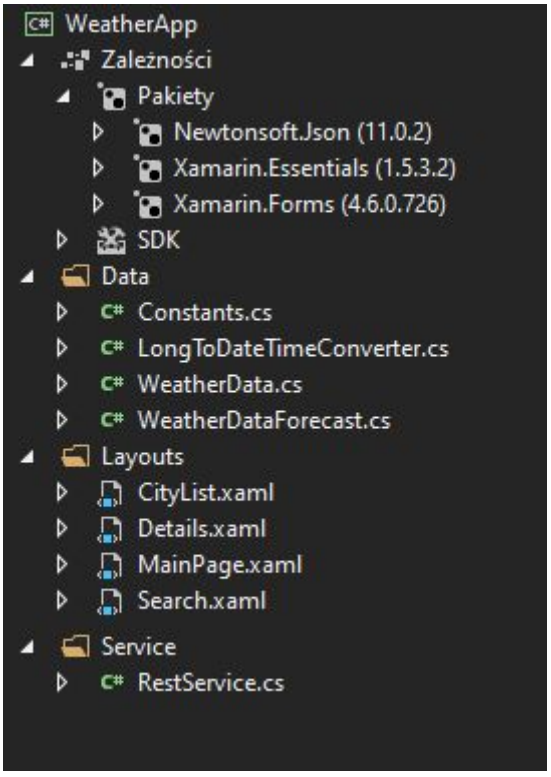
5. RestService.cs

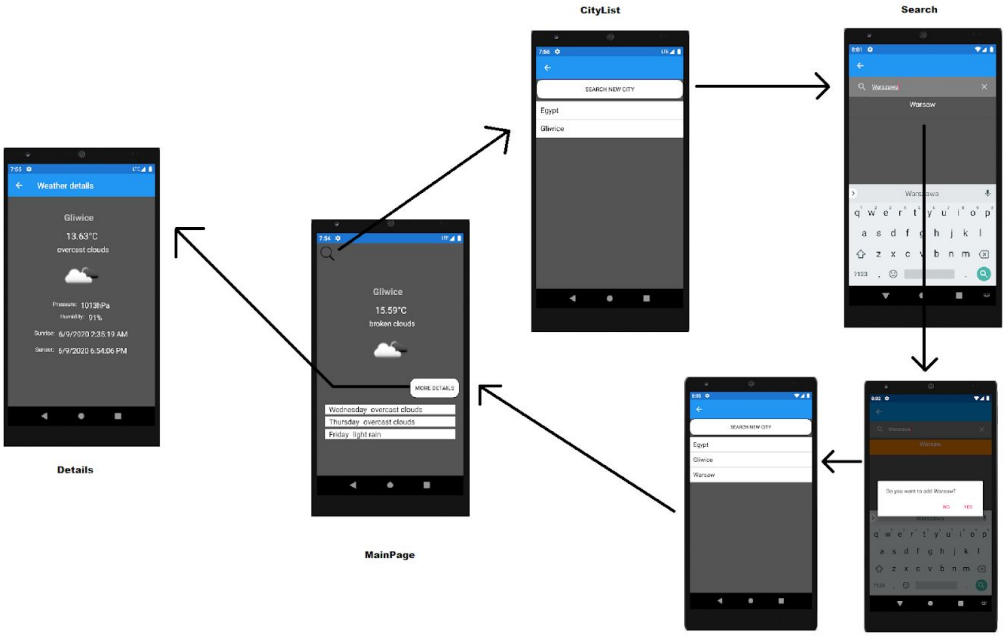
Przeprowadzono testy sprawdzające: połączenia z internetem przed wysłaniem zapytania na serwer, oraz sprawdzenie czy zapytanie zwróciło pozytywnie odpowiedź.

```
if (current == NetworkAccess.Internet)
{
    // Connection to internet is available
    WeatherData weatherData = null;
    try
    {
        var response = await _client.GetAsync(query);
        if (response.IsSuccessStatusCode)
        {
            var content = await response.Content.ReadAsStringAsync();
            weatherData = JsonConvert.DeserializeObject<WeatherData>(content);
        }
    }
    catch (Exception ex)
    {
        Debug.WriteLine("\t\tERROR {0}", ex.Message);
    }

    return weatherData;
}
else
{
    return null;
}
```

Schemat struktury wewnętrznej aplikacji





Zrzut tablicy Trello