

PAŃSTWOWA WYŻSZA SZKOŁA ZAWODOWA
W NOWYM SĄCZU

Instytut Techniczny
Informatyka Stosowana

DOKUMENTACJA PROJEKTOWA
PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Zdrowy Spacer

Autorzy:
Kamil Pociecha
Nicolas Świątnik

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2021

Spis treści

1. Ogólne określenie wymagań	3
1.1. Podstawowe wymagania	3
1.2. Dodatkowe wymagania	3
2. Określenie wymagań szczegółowych	4
2.1. Opis wymagań	4
2.2. Layout aplikacji	4
3. Projektowanie	6
3.1. Przygotowanie narzędzi Git oraz Visual Studio	6
3.2. Biblioteki i klasy	6
3.3. Layouty dla opcji z menu	7
3.4. Stworzenie pliku apk na Androida z poziomu Visual Studio	9
4. Implementacja	12
5. Testowanie	21
5.1. Tworzenie nowego konta oraz logowanie do aplikacji	21
5.2. Test działania geolokalizacji i geokodowania na stronie głównej	23
5.3. Test działania mapy	24
5.4. Obsługa aparatu i dodawanie zdjęcia z galerii	25
6. Podręcznik użytkownika	27
Literatura	32
Spis rysunków	34
Spis tabel	35

1. Ogólne określenie wymagań

Aplikacja ma się nazywać Zdrowy Spacer a jej grupą docelową mają być osoby aktywne fizycznie, które chcą kontrolować swoje wyniki.

1.1. Podstawowe wymagania

Aplikacja powinna ona spełniać podstawowe funkcjonalności takie jak: pomiary przebytych odległości, liczbę przebytych kroków, liczbę spalonych kalorii.

1.2. Dodatkowe wymagania

Dodatkowo chcielibyśmy aby aplikacja rzutowała całą trasę na mapy, miała chociażby zapisywaną historię poprzednich tras do oglądu dla użytkownika, oczywiście umożliwiała ustalenie celu czy to podróży lub przebytych kroków/spalonej kalorii. Jako dodatkową funkcję przewidujemy też możliwość robienia zdjęć osiągniętego już celu, czyli np. zrobienie zdjęcia szczytu góry jako potwierdzenie osiągnięcia swojego celu. Chcielibyśmy również umożliwić naszym klientom logowanie się poprzez konta typu Facebook czy Google, co mogło by zautomatyzować proces logowania. Kolejną opcja którą przewidujemy by znalazła się w zamawianej przez nas aplikacji jest motyw typu jasny/ciemny dla uprzyjemnienia samego doświadczenia z korzystania z niej. Oczywiście aplikacja powinna mieć możliwość bezproblemowego działania w tle jak i również ewentualnego wykrycia ruchu bez jej inicjacji co przełoży się na to że sama zacznie rejestrować nasz spacer oraz aktywności.

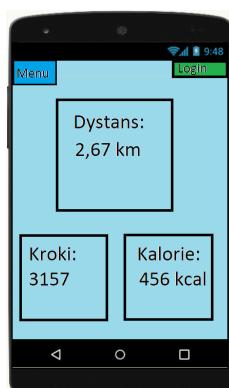
2. Określenie wymagań szczegółowych

2.1. Opis wymagań

Aplikacja korzystać będzie z czujnika GPS, który umożliwi pomiar pokonanej odległości oraz rzutowanie trasy na mapę - w tym celu użyte zostaną mapy Google. Na podstawie przebytej odległości będzie można określić także liczbę kroków oraz ilość spalonych kalorii. Odległości przebyte w poprzednich dniach będą zapisywane w aplikacji co pozwoli użytkownikowi na porównywanie swoich wyników z poszczególnych dni. W przypadku osiągnięcia wyniku poniżej wybranej normy użytkownik zostanie powiadomiony o tym fakcie przez aplikację. W celu zmiany motywu aplikacji użyty zostanie czujnik światła, dzięki któremu w trakcie dnia będzie funkcjonował tryb ciemny a w nocy tryb jasny. Użytkownik będzie miał możliwość zaznaczenia na mapie swojego celu, do którego chce danego dnia dotrzeć. Następnie będzie miał możliwość by przy użyciu aparatu w telefonie wykonać zdjęcie osiągniętego miejsca docelowego. Zdjęcie to będzie zapisywane w folderze aparatu a użytkownik będzie miał wgląd w zdjęcia swoich poprzednich celów. Aby ułatwić proces logowania Użytkownik będzie miał możliwość logowania do aplikacji za pomocą Facebook'a lub konta Google. Aplikacja będzie działała cały czas bez potrzeby inicjowania. Możliwe będzie jej działanie w tle a także zwijanie do paska.

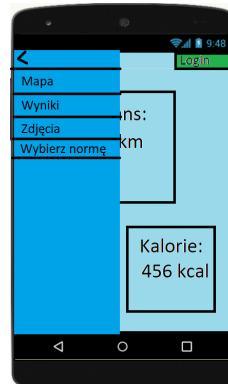
2.2. Layout aplikacji

Na stronie głównej aplikacja będzie wyświetlać przebytą odległość, liczbę wykonanych kroków oraz liczbę spalonych kalorii. W prawym górnym rogu znajduje się przycisk, który po jego naciśnięciu umożliwi zalogowanie się do aplikacji. W lewym górnym rogu znajduje się przycisk otwierający menu co widać na rysunku 2.1.



Rys. 2.1. podstawowy layout aplikacji

Na rysunku 2.2 widzimy, że po otwarciu menu pojawi się lista w której znajdują się elementy takie jak: mapa - pokazuje przebytą trasę oraz umożliwia ustalenie swojego celu, wyniki - wyświetla wyniki osiągnięte w poprzednich dniach, Zdjęcia - otwiera galerię ze zdjęciami miejsc docelowych, wybierz normę - pozwala ustalić użytkownikowi jaki wynik chciałby osiągnąć.



Rys. 2.2. layout aplikacji po otwarciu menu

3. Projektowanie

3.1. Przygotowanie narzędzi Git oraz Visual Studio

W celu korzystania z narzędzia Git należy utworzyć na swoim komputerze repozytorium lokalne oraz dodać do niego pliki projektu. Na platformie GitHub utworzyć repozytorium zdalne, które umożliwi wszystkim autorom projektu współpracę przy tworzeniu aplikacji.

W Visual Studio należy zainstalować dodatek opracowywanie aplikacji mobilnych za pomocą środowiska Xamarin oraz zestaw Android SDK, który umożliwia korzystanie z emulatora. Do tworzenia projektu wybraliśmy szablon aplikacji mobilnej Xamarin.Forms. Xamarin.Forms ma duży ekosystem bibliotek, które dodają różne funkcje do aplikacji. W tej sekcji opisano niektóre z tych dodatkowych funkcji¹.

3.2. Biblioteki i klasy

Biblioteka Xamarin.Essentials udostępnia międzyplatformowy interfejs programowania aplikacji mobilnych (API). Jest ona dostępna jako pakiet NuGet i jest uwzględniana w każdym nowym projekcie w programie Visual Studio. Biblioteka ta oferuje klasy, które zostaną przez nas użyte podczas tworzenia projektu².

Geolokalizacja:

Klasa Geolocation udostępnia interfejsy API do pobierania bieżących współrzędnych geolokalizacji urządzenia.

Motyw aplikacji:

Interfejs API RequestedTheme jest częścią klasy AppInfo i zawiera informacje dotyczące tematu żądanego dla uruchomionej aplikacji przez system.

Akcelerometr:

Klasa Accelerometer umożliwia monitorowanie czujnika przyspieszeniomierza urządzenia, który wskazuje przyspieszenie urządzenia w trójwymiarowej przestrzeni.

GoogleMap:

Firma Google oferuje natywny interfejs API mapowania dla systemu Android. Pozwala on na zmienianie punktu widzenia mapy, dodawanie i dostosowywanie znaczników, oznaczanie mapy za pomocą nakładek.

Wymagania wstępne Mapy API usługi Google: uzyskanie klucza Mapy API, zainstalowanie pakietu Xamarin.GooglePlayServices i Mapy pakietu z NuGet, określenie wymaganych uprawnień.

¹<https://docs.microsoft.com/pl-pl/xamarin/get-started/what-is-xamarin-forms>[1].

²<https://docs.microsoft.com/pl-pl/xamarin/essentials/?context=xamarin/xamarin-forms>[2].

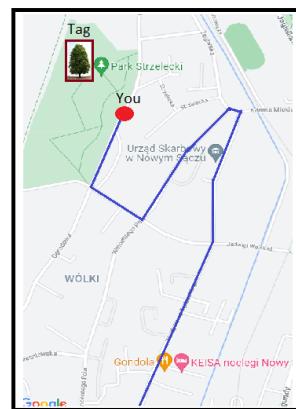
Klasa GoogleMaps poprzez aplikację platformy Xamarin.Android będzie współdziałała z aplikacją Google Maps.

Biblioteka Microsoft Authentication Library (MSAL) pozwala na dodawanie uwierzytelniania do aplikacji. Umożliwi to logowanie do aplikacji przy użyciu Google lub Facebook.

Klasa WebAuthenticator umożliwia inicjowanie przepływów opartych na przeglądarce, które nasłuchują wywołania zwrotnego do określonego adresu URL zarejestrowanego w aplikacji.

3.3. Layouty dla opcji z menu

Po wybraniu z menu opcji **Mapa** na ekranie wyświetli się mapa pokazująca przebytą trasę oraz zaznaczająca miejsce w którym użytkownik w danym momencie się znajduje co jest pokazane na rysunku 3.1. Na mapie będą się także wyświetlały zdjęcia miejsc docelowych zrobione przez użytkownika.



Rys. 3.1. Layout - Mapa

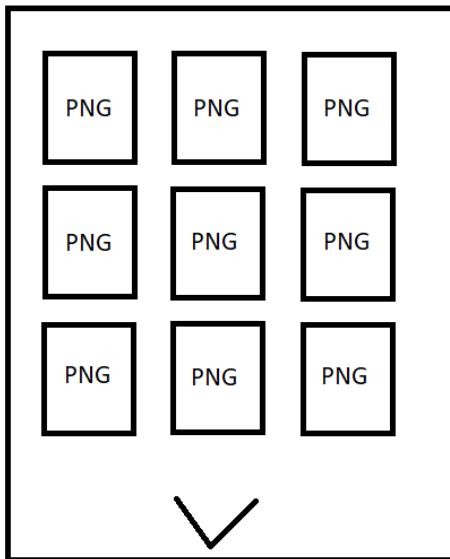
Opcja **Wyniki** wyświetli stronę zawierającą wyniki z ostatnich 7 dni oraz najlepszy wynik z całej historii. Wśród wyświetlanych wyników znajdą się: Przybyta odległość, liczba kroków oraz liczba spalonych kalorii jak na rysunku 3.2. Pozwoli to użytkownikowi na porównywanie swoich wyników oraz zmotywuje do poprawiania

rekordowych wyników.

Dzień	Dystans	Kroki	Kcal
czwartek			
środa			
wtorek			
poniedziałek			
niedziela			
sobota			
piątek			
Najlepszy wynik	15.028 km	20 037	987

Rys. 3.2. Layout - Wyniki

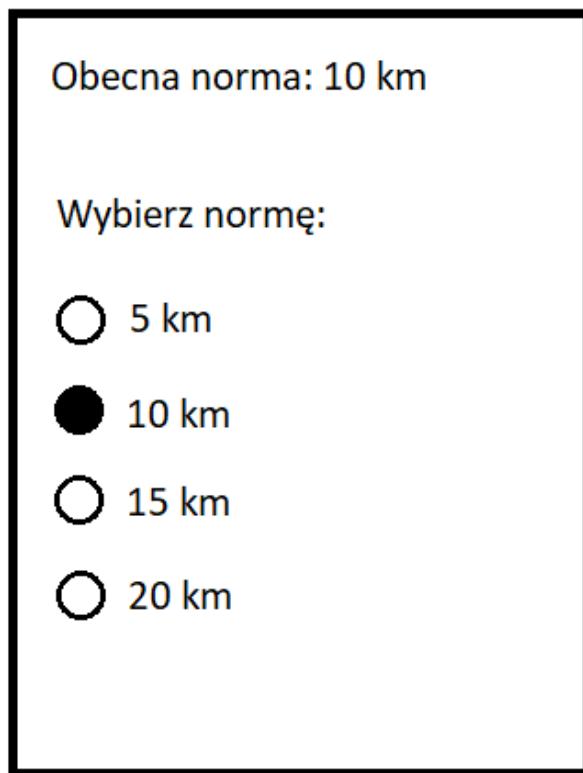
Wybór opcji **Zdjęcia** otworzy galerie zdjęć wykonanych przez użytkownika jak na rysunku 3.3. Zdjęcia będą przedstawiać miejsca docelowe uwiecznione aparatem telefonu.



Rys. 3.3. Layout - Zdjęcia

Opcja **Wybierz** normę pozwala na wybranie dystansu jaki użytkownik planuje prze-

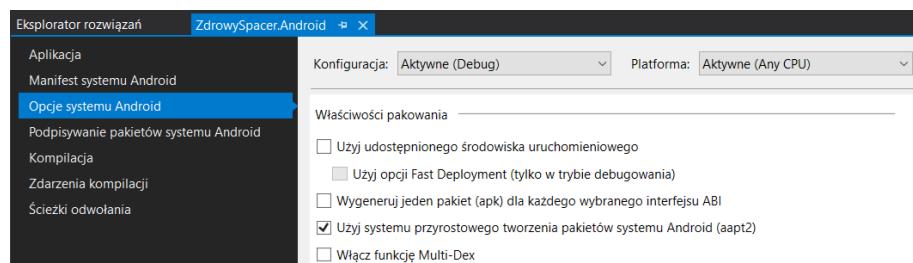
być w ciągu dnia. Polega to na zaznaczeniu checkboxa obok wybranej odległości zo jest widoczne na rysunku 3.4.



Rys. 3.4. Layout - Wybierz normę

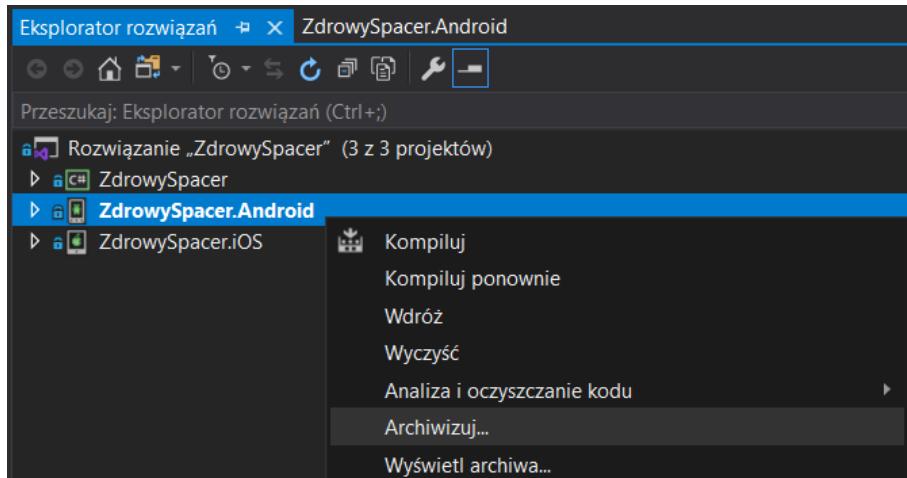
3.4. Stworzenie pliku apk na Androida z poziomu Visual Studio

Aby umożliwić stworzenie pliku apk w Visual Studio należy wejść w właściwości rozwiązania dla systemu Android. Następnie w opcjach systemu Android trzeba odznaczyć opcję **Użyj udostępnionego środowiska uruchomieniowego** jak na rysunku 3.5.



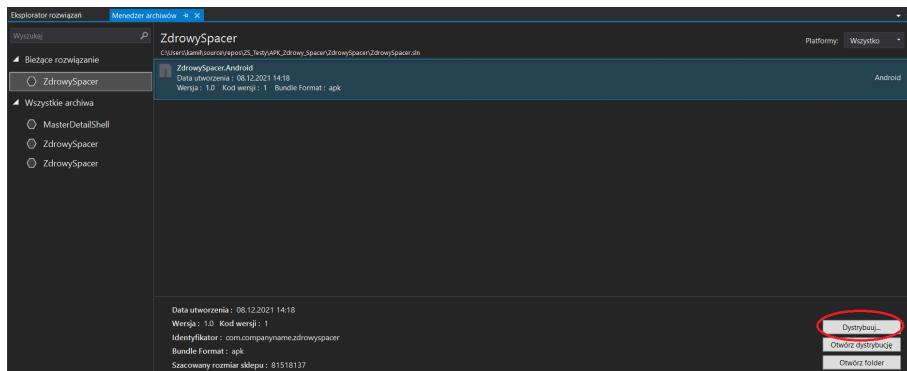
Rys. 3.5. Odznaczanie właściwości w Opcjach systemu Android

Na rysunku 3.6 widzimy, że kolejny krok to Archiwizacja rozwiązania dla systemu Android



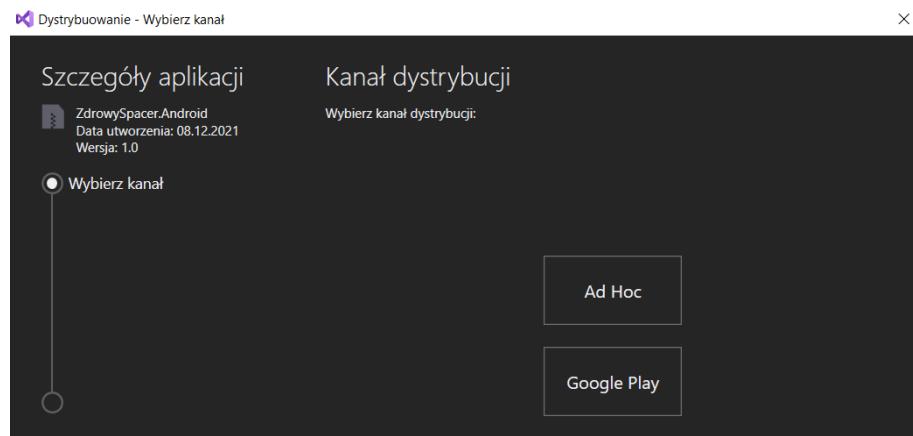
Rys. 3.6. Archiwizacja rozwiązania dla systemu Android

W Menedżer archiwów należy wybrać wersje aplikacji i nacisnąć przycisk **Dstrybuuj...** jak na rysunku 3.7.



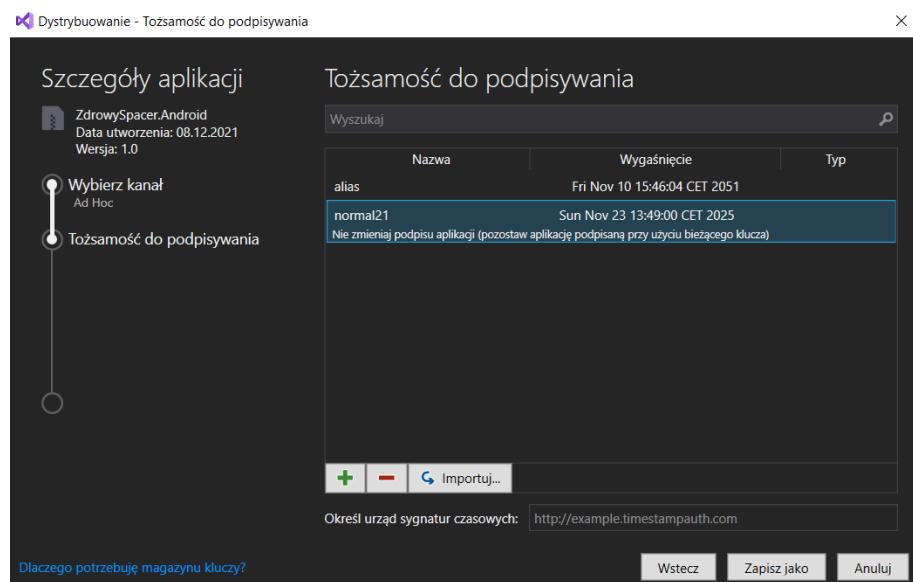
Rys. 3.7. Dstrybucja pliku

Na rysunku 3.8 zaprezentowano wybór kanału dystrybucji jako opcji **Ad Hoc**



Rys. 3.8. Wybranie kanału dystrybucji

Ostatni krok to podpisanie pliku za pomocą wcześniej utworzonego aliasu jak widać na rysunku 3.9 i wybranie miejsca w którym chcemy zapisać plik apk



Rys. 3.9. Podpis za pomocą aliasu

4. Implementacja

Tworzenie Menu:

Do stworzenia menu użyty został **Xamarin.forms Shell**, który zmniejsza złożoność tworzenia aplikacji, oferując podstawowe funkcje. Obejmuje on wspólne środowisko użytkownika nawigacji, schematu nawigacji i zintegrowanej procedury obsługi wyszukiwania.

Dodawanie strony do menu na przykładzie elementu wyniki:

W pliku **MainPage.xaml**:

```
<FlyoutItem Title="MyTabAPP"
    Shell.TabBarIsVisible="False"
    FlyoutDisplayOptions="AsMultipleItems">
    <ShellContent Title="Wyniki" Icon="ic_wyniki" ContentTemplate="{DataTemplate local:Wyniki}"/>
</FlyoutItem>
```

Rys. 4.1. Dodanie strony Menu do menu bocznego

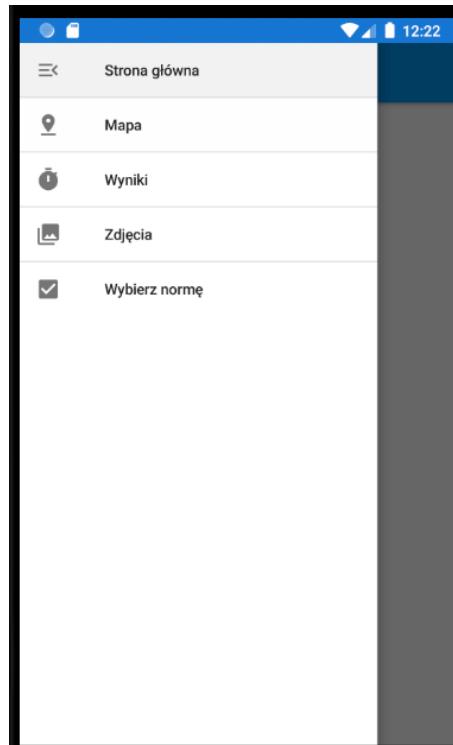
Na rysunku 4.1 **ic_wyniki** to nazwa ikonki a **local:Wyniki** to odnośnik do plików strony "Wyniki", które znajdują się w folderze głównym projektu.

Dodawanie ikonek do projektu:

Ikonki pobrane zostały z **Android Asset Studio** w formacie ".png".

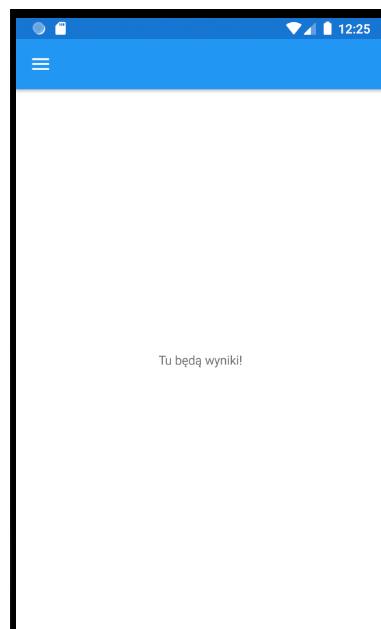
Aby użyć ikonki w projekcie należy umieścić je w dwóch osobnych miejscach. Dla androida jest to folder **drawable** znajdujący się w folderze resources a dla systemu iOS folder **resources**.

Działanie menu bocznego w emulatorze Android 8.1:



Rys. 4.2. Widok menu bocznego

Na rysunku 4.2 można zobaczyć menu boczne z opcjami do wyboru.



Rys. 4.3. Widok strony otwartej po wybraniu danej opcji z menu

Na rysunku 4.3 widzimy nowo otwartą stronę po wybraniu opcji z menu bocz-

nego.

Dodanie pól wyboru na stronie "Wybierz normę":

Do stworzenia pól wyboru z których można wybrać tylko jedną opcję potrzebne jest zainstalowanie pakietu Nuget **Xamarin.Forms.InputKit**. Przy użyciu tego pakietu można użyć opcji **RadioButton**, dzięki której tworzona jest lista z polami do wyboru.

Fragment kodu z pliku **Wybierz_norme.xml** został pokazany na rysunku 4.4.

```
<input:RadioButtonGroupView>
    <input:RadioButton Text="5 km" />
    <input:RadioButton Text="10 km " />
    <input:RadioButton Text="15 km" />
    <input:RadioButton Text="20 km" />
</input:RadioButtonGroupView>
```

Rys. 4.4. Dodanie opcji z polami do wyboru

Dodanie mapy pokazującej obecną lokalizację

Użycie pakietu Nuget xamarin.forms.maps umożliwia wyświetlenie na stronie mapy ze znacznikiem pokazującym obecną lokalizację.

W pliku **AndroidManifest.xml** w sekcji application - meta-data należy dodać klucz interfejsu API, który można utworzyć na stronie developers.google.com. Wprowadzić należy także nazwe metadanych dla klucza interfejsu API. Kolejna linijka również posiada sekcję meta-data i określa numer wersji Google Play usługi. W sekcji uses-library deklarowana jest biblioteka Apache HTTP jak to zostało pokazane na rysunku 4.5.

```
<meta-data android:name="com.google.android.geo.API_KEY" android:value="AIzaSyBd-ag_GGwxdWLrk-gE_KXQaU36TMG9FZk" />
<meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version" />
<uses-library android:name="org.apache.http.legacy" android:required="false" />
```

Rys. 4.5. Dodanie nowych deklaracji w AndroidManifest.xaml

Na rysunku 4.6 widzimy fragment pliku **mapa.xaml.cs** w którym utworzona została klasa **DisplayCurLoc**, w której zawarte są instrukcje odpowiedzialne za

wykrycie obecnej lokalizacji.

```
public async void DisplayCurLoc()
{
    try
    {
        var request = new GeolocationRequest(GeolocationAccuracy.Medium, TimeSpan.FromSeconds(10));
        var location = await Geolocation.GetLocationAsync(request);
        if (location != null)
        {
            Position p = new Position(location.Latitude, location.Longitude);
            MapSpan mapSpan = MapSpan.FromCenterAndRadius(p, Distance.FromKilometers(.444));
            myMap.MoveToRegion(mapSpan);
            Console.WriteLine($"Latitude: {location.Latitude}, Longitude: {location.Longitude}, Altitude: {location.Altitude}");
        }
    }
}
```

Rys. 4.6. Dodanie funkcji w mapa.xaml.cs

W pliku **mapa.xaml** wartość opcji IsShowingUser ustawiona jest jako True, dzięki czemu na mapie wyświetlony zostanie znacznik wskazujący na obecną lokalizację. Opcja x:Name nadaje nazwę dla używanej mapy tak jak na rysunku 4.7.

```
<myMap:Map IsShowingUser="True" x:Name="myMap" />
```

Rys. 4.7. Ustawienie opcji w mapa.xaml

Dodanie obsługi aparatu

Do obsługi aparatu użyty został pakiet Nuget **Xam.Plugin.Media**. Po naciśnięciu przycisku otwiera się aparat telefonu i można wykonać zdjęcie, które następnie zostanie zapisane i wyświetcone na stronie.

```
<Image x:Name="imgCam"
       Grid.Row="0"
       Grid.Column="0"/>

<Button x:Name="btnCam"
        Text="Zdjęcie"
        Grid.Row="1"
        Grid.Column="0"
        Margin="2"
        Clicked="BtnCam_Clicked"
        BackgroundColor="#00a4fc"
        TextColor="#FFF"/>
```

Rys. 4.8. Plik Zdjecia.xml

Na rysunku 4.8 przedstawiony jest fragment kodu ze strony **Zdjecia.xml**. Opcja Image odpowiada za dodanie zrobionego zdjęcia na stronę. Opcja Button tworzy przycisk z napisem Zdjęcie, którego naciśnięcie otworzy aparat.

```
private async void BtnCam_Clicked(object sender, EventArgs e)
{
    try
    {

        var photo = await CrossMedia.Current.TakePhotoAsync(new StoreCameraMediaOptions()
        {
            DefaultCamera = Plugin.Media.Abstractions.CameraDevice.Rear,
            Directory = "Xamarin",
            SaveToAlbum = true
        });

        if (photo != null)
            imgCam.Source = ImageSource.FromStream(() => { return photo.GetStream(); });

    }
    catch (Exception ex)
    {
        await DisplayAlert("Error", ex.Message.ToString(), "Ok");
    }
}
```

Rys. 4.9. Plik Zdjecia.xml.cs

Na rysunku 4.9 znajduje się fragment kodu znajdującego się na stronie **Zdjecia.xml.cs**. Funkcja **BtnCamClicked** ustawia domyślną kamerę na tylną kamerę naszego telefonu. Ścieżka pliku obrazu ustawiona jest do folderu ”Xamarin”. Atrybut zostaje przekazany jako true, aby zapisać obraz.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Rys. 4.10. Plik AndroidManifest.xml

Na rysunku 4.10 przedstawiony jest fragment pliku **AndroidManifest.xml**. Aby prawidłowa obsługa aparatu była możliwa zadeklarowane muszą być zezwolenia na dostęp do uprawnień aparatu i pamięci zewnętrznej.

Rejestracja i logowanie z wykorzystaniem Google Firebase

Baza danych Google Firebase umożliwia rejestrację oraz logowanie użytkowników do aplikacji. Użytkownicy, którzy utworzą nowe konto zostają zapisani w bazie danych

i mają możliwość logowania się do aplikacji za pomocą adresu Email i hasła³. Do korzystania z autoryzacji Firebase konieczne jest pobranie dwóch pakietów NuGet: **Xamarin.FirebaseAuth** i **Xamarin.Firebase.Core** oraz inicjalizacja usługi w pliku **MainActivity.cs** w rozwiążaniu Android co zostało pokazane na rysunku 4.11.

```
        FirebaseApp.InitializeApp(Context);
```

Rys. 4.11. Inicjalizacja w pliku MainActivity.cs

W rozwiązaniu głównym stworzony został plik **Iauth.cs** a w nim interfejs **Iauth**.

```
public interface IAuth
{
    Task<string> LoginWithEmailAndPassword(string email, string password);

    Task<string> SignUpWithEmailAndPassword(string email, string password);

    bool SignOut();

    bool IsSignIn();
}
```

Rys. 4.12. Interfejs Iauth w pliku Iauth.cs

Jak widać na rysunku 4.12 wewnątrz interfejsu utworzone zostały cztery funkcje. **LoginwithEmailAndPassword** i **SignUpWithEmailAndPassword**, które przyjmują email i hasło jako zmienne typu string i będą wykorzystywane przy tworzeniu nowego konta a także przy logowaniu. Funkcje bool **SignOut** oraz **IsSignIn** będą potrzebne do sprawdzania statusu użytkownika.

W rozwiązaniu dla Androida utworzono klasę **AuthDroid.cs** w której znajdują się instrukcje dla funkcji, które zostały pokazane w interfejsie.

Na rysunku 4.13 przedstawiono funkcję rejestracji. Przy użyciu autoryzacji Firebase tworzona jest nowe konto użytkownika. Pobrany zostaje Email i hasło a użytkownik otrzymuje unikalny numer Uid. Aby konto zostało poprawnie utworzone należy użyć poprawnej formy adresu email i hasła o długości minimum 6 znaków.

³[https://firebase.google.com/docs/auth\[3\]](https://firebase.google.com/docs/auth).

```
public async Task<string> SignUpWithEmailAndPassword(string email, string password)
{
    try
    {
        var newUser = await FirebaseAuth.Instance.CreateUserWithEmailAndPasswordAsync(email, password);
        var token = newUser.User.Uid;
        return token;
    }
    catch (FirebaseAuthInvalidUserException e)
    {
        e.PrintStackTrace();
        return string.Empty;
    }
    catch (FirebaseAuthInvalidCredentialsException e)
    {
        e.PrintStackTrace();
        return string.Empty;
    }
}
```

Rys. 4.13. Funkcja rejestracji

Podobnie wygląda to w funkcji odpowiadającej za logowanie przedstawionej na rysunku 4.14.

```
public async Task<string> LoginWithEmailAndPassword(string email, string password)
{
    try
    {
        var user = await FirebaseAuth.Instance.SignInWithEmailAndPasswordAsync(email, password);
        var token = user.User.Uid;
        return token;
    }
    catch (FirebaseAuthInvalidUserException e)
    {
        e.PrintStackTrace();
        return string.Empty;
    }
    catch (FirebaseAuthInvalidCredentialsException e)
    {
        e.PrintStackTrace();
        return string.Empty;
    }
}
```

Rys. 4.14. Funkcja logowania

Email i hasło zostają sprawdzone, jeżeli zgadzają się one z danymi zawartymi w bazie to logowanie jest skuteczne. W przypadku użycia błędnych danych użytkownik dostaje informacje o błędzie.

Kolejne dwie funkcje sprawdzają status użytkownika przy użyciu typu bool.

```
public bool IsSignIn()
{
    var user = FirebaseAuth.Instance.CurrentUser;
    return user != null;
}
```

Rys. 4.15. Funkcja IsSignIn

Widoczna na rysunku 4.15 funkcja **IsSignIn** sprawdza jaki użytkownik jest zalogowany i zwraca informacje o nim.

```
public bool SignOut()
{
    try
    {
        FirebaseAuth.Instance.SignOut();
        return true;
    }
    catch (Exception e)
    {
        return false;
    }
}
```

Rys. 4.16. Funkcja SignOut

Funkcja **SignOut** widoczna na rysunku 4.16 odpowiada za wylogowanie użytkownika z aplikacji. Jeżeli wszystko przebiegnie prawidłowo zwróci ona wartość true a jeżeli operacja się nie powiedzie to zwróci wartość false.

W rozwiążaniu głównym w pliku App.xaml dodana jest instrukcja odpowiedzialna za przekierowanie użytkownika.

Wywołana zostaje funkcja **IsSignIn**, która sprawdza czy użytkownik jest poprawnie zalogowany. Jeżeli logowanie przebiegło prawidłowo to użytkownik zostanie

```
IAuth auth;
public App()
{
    InitializeComponent();

    auth = DependencyService.Get<IAuth>();

    if (auth.IsSignIn())
    {
        MainPage = new Strona_glowna();
    }
    else
    {
        MainPage = new Login();
    }

    MainPage = new MainPage();
}
```

Rys. 4.17. Przekierowanie w pliku App.xaml

przeniesiony na stronę główną aplikacji, natomiast jeżeli logowanie się nie powiedzie to zostaje on przekierowany na stronę logowania co widać na rysunku 4.17.

5. Testowanie

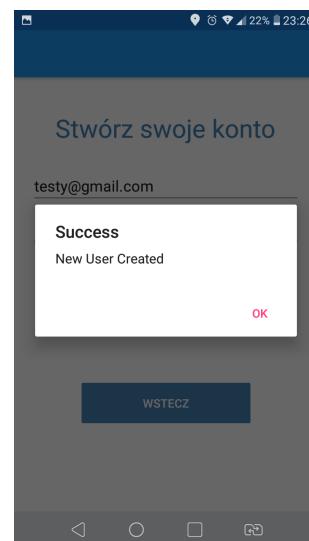
5.1. Tworzenie nowego konta oraz logowanie do aplikacji

Naszym celem jest utworzenie nowego konta użytkownika a następnie zalogowanie się do aplikacji przy pomocy nowo utworzonego konta.



Rys. 5.1. Tworzenie nowego konta

Wpisujemy poprawny adres E-mail oraz hasło składające się z co najmniej sześciu znaków jak to jest pokazane na rysunku 5.1 a następnie klikamy przycisk zarejestruj się.



Rys. 5.2. Informacja o utworzeniu konta

Po kliknięciu przycisku zarejestruj się otrzymujemy informacje, że tworzenie nowego konta przebiegło pomyślnie jak widać na rysunku 5.2.

The screenshot shows the 'Authentication' section of the Firebase console for a project named 'Spacer'. The 'Users' tab is selected. A search bar at the top allows searching by email address, phone number, or user UID. Below it is a table with columns: Identifier, Providers, Created, Signed In, and User UID. A single row is visible for a user with the identifier 'testy@gmail.com', created on 'Jan 11, 2...', signed in on 'Jan 11, 2...', and a long User UID. There are buttons for 'Add user' and 'Copy'.

Rys. 5.3. Nowe konto w bazie danych

Na rysunku 5.3 widzimy, że nowy użytkownik został dodany w bazie danych Google Firebase co oznacza, że powinna być możliwość zalogowania się na jego konto.



Rys. 5.4. Logowanie na utworzone konto

Na stronie logowania jak widać na rysunku 5.4 wpisujemy E-mail i hasło takie samo jak podczas zakładania konta. Następnie klikamy przycisk zaloguj się.

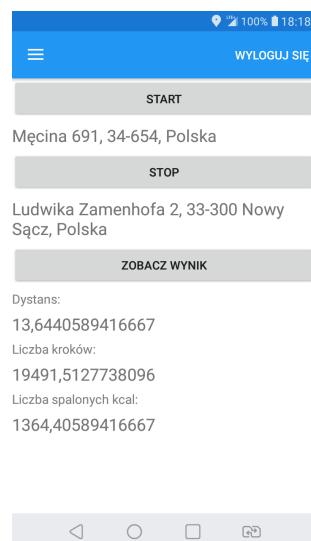
Jak widać na rysunku 5.5 logowanie przebiegło pomyślnie i zostaliśmy przeniesieni na stronę główną aplikacji.



Rys. 5.5. Widok po zalogowaniu

5.2. Test działania geolokalizacji i geokodowania na stronie głównej

Celem naszego testu jest sprawdzenie czy po kliknięciu przycisków aplikacja wykonuje oczekiwane przez nas działania. Przyciski start i stop powinny zwracać nam informacje naszym obecnym adresie pobieranym w momencie klikania danego przycisku. Przycisk zobacz wynik powinien wyświetlić nam wyniki czyli przebyty dystans, liczbę kroków i liczbę spalonych kalorii.



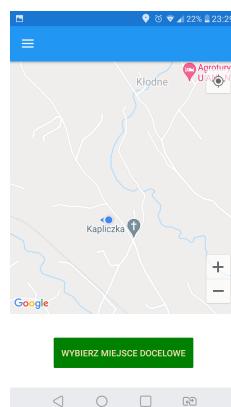
Rys. 5.6. Wyniki

Na rysunku 5.6 pokazany jest wynik końcowy po użyciu wszystkich trzech przy-

cisków. W momencie, gdy zaczynamy spacer naciskamy przycisk start a aplikacja wypisuje nam obecną lokalizację. Po zakończeniu spaceru naciskamy przycisk stop, aplikacja podaje nam adres końcowy. Naciskamy przycisk zobacz wyniki i otrzymujemy przebyty dystans, liczbę wykonanych kroków i liczbę spalonych kalorii. Pokazany dystans zgadza się z rzeczywistą odlegością jaką przebyliśmy.

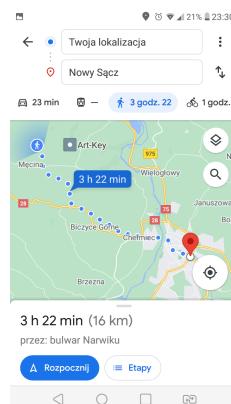
5.3. Test działania mapy

Nasza mapa powinna pokazywać miejsce w którym znajduje się użytkownik za pomocą wskaźnika. Po kliknięciu przycisku wybierz miejsce docelowe powinniśmy zostać przeniesieni do Google Maps, gdzie będzie możliwość wyznaczenia trasy poprzez wpisanie naszego celu.



Rys. 5.7. Mapa z lokalizacją

Jak widać na rysunku 5.7 mapa pokazuje naszą obecną lokalizację za pomocą wskaźnika. Wyznaczona lokalizacja jest zgodna z naszym realnym położeniem.



Rys. 5.8. Google Maps

Po kliknięciu przycisku wybierz miejsce docelowe zostajemy przeniesieni do Google Maps, tam wpisujemy nasze miejsce docelowe i jak to jest pokazane na rysunku 5.8 otrzymujemy wyznaczoną trasę od naszego miejsca pobytu do miejsca docelowego.

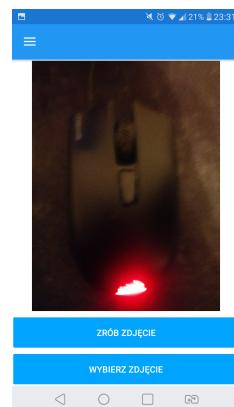
5.4. Obsługa aparatu i dodawanie zdjęcia z galerii

Celem testu jest otwarcie aparatu z poziomu aplikacji i wykonanie zdjęcia, które następnie powinno zostać wyświetcone na stronie. Potem sprawdzimy możliwość zamiany tego zdjęcia na inne wybrane z galerii.



Rys. 5.9. Wykonanie zdjęcia

Po naciśnięciu przycisku zrób zdjęcie otwiera się nasz aparat z ustawioną domyślnie kamerą tylną aparatu. Po zrobieniu zdjęcia możemy je zatwierdzić klikając ok co widać na rysunku 5.9.



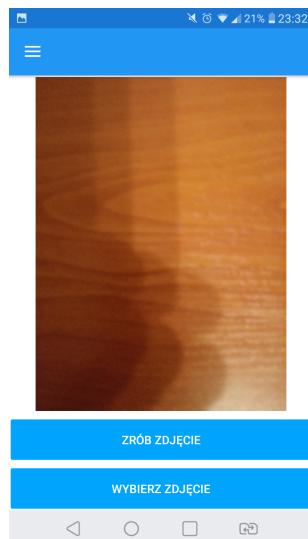
Rys. 5.10. Wyświetlenie zdjęcia

Na rysunku 5.10 widać, że zdjęcie zostało poprawnie dodane na stronie w aplikacji.



Rys. 5.11. Wybieranie zdjęcia z galerii

Po kliknięciu przycisku wybierz zdjęcie otwiera się nasza galeria jak to jest pokazane na rysunku 5.11 w której możemy obejrzeć wykonane przez nas wcześniej zdjęcia. Wybieramy jedno z wcześniejszych zdjęć aby widzieć je w naszej aplikacji.

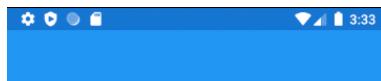


Rys. 5.12. Nowe zdjęcie

Jak widać na rysunku 5.12 zdjęcie zostało zmienione na to, które wybraliśmy z naszej galerii.

6. Podręcznik użytkownika

Ekran logowania



Zdrowy Spacer

E-mail

Password

ZALOGUJ SIĘ

STWÓRZ NOWE KONTO

BEZ LOGOWANIA

Rys. 6.1. Strona startowa

Po włączeniu aplikacji widoczna jest strona logowania, którą widać na rysunku 6.1. Jeżeli użytkownik ma już utworzone konto może się zalogować poprzez wprowadzenie swojego adresu email i hasła a następnie naciśnięcie przycisku **zaloguj się**. Jeśli użytkownik chce utworzyć nowe konto to należy wybrać przycisk **stwórz nowe konto**.



Stwórz swoje konto

E-mail

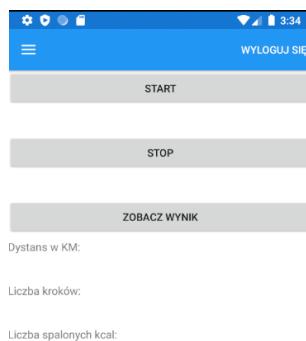
Hasło

ZAREJESTRUJ SIĘ

Rys. 6.2. Strona rejestracji

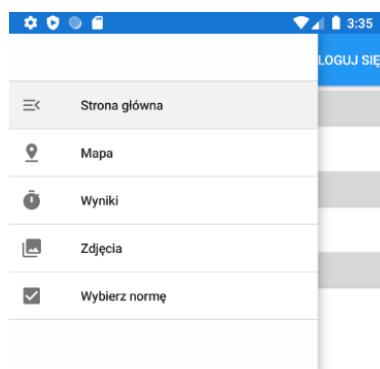
Po wybraniu opcji **stwórz nowe konto** użytkownik zostanie przeniesiony na

stronę rejestracji, która jest przedstawiona na rysunku 6.2. Aby utworzyć nowe konto użytkownik musi wprowadzić prawidłowy adres email oraz hasło o długości minimum 6 znaków a następnie nacisnąć przycisk **zarejestruj się**. W przypadku poprawnego utworzenia konta użytkownik zostanie przeniesiony na stronę logowania. Jeżeli tworzenie konta się nie powiedzie pojawi się komunikat o błędzie.



Rys. 6.3. Strona główna

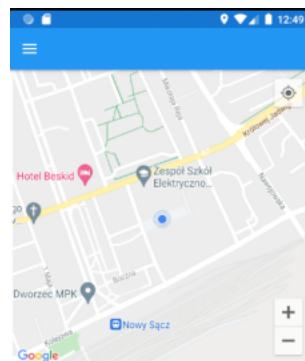
Na rysunku 6.3 widzimy stronę główną na którą po zalogowaniu zostanie przeniesiony użytkownik na stronę główną. Wybranie przycisku **wyloguj się** który znajduje się w prawym górnym rogu spowoduje wylogowanie z aplikacji i powrót na stronę logowania. Wybranie ikony menu, która znajduje się w prawym górnym rogu spowoduje wyświetlenie się menu bocznego. Przycisk **start** należy kliknąć w momencie rozpoczęcia spaceru, spowoduje to wyświetlenie adresu lokalizacji startowej. Przycisk **stop** należy nacisnąć w momencie zakończenia spaceru, zostanie wtedy wyświetlony adres lokalizacji końcowej. Kliknięcie przycisku **zobacz wynik** będzie skutkowało wyświetleniem przebytego dystansu podanego w kilometrach, liczby wykonanych kroków oraz liczby spalonych kalorii.



Rys. 6.4. Widok po wyłączeniu menu bocznego

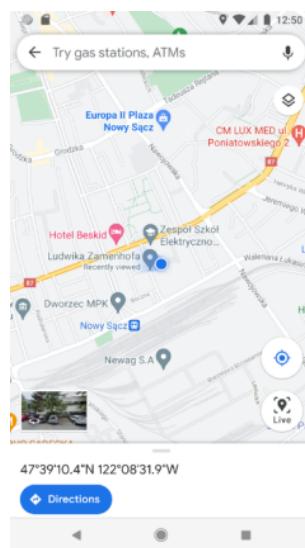
Na rysunku 6.4 widzimy menu z którego użytkownik ma do wyboru pięć opcji:

strona główna, mapa, wyniki, zdjęcia, wybierz normę. W zależności od tego, którą opcję wybierze użytkownik zostanie on przekierowany na odpowiadającą swojemu wyborowi stronę.



Rys. 6.5. Mapa

Po wybraniu mapy widoczna będzie mapa wraz ze znacznikiem określającym obecną lokalizację użytkownika oraz przycisk **Wybierz miejsce docelowe** tak jak to jest pokazane na rysunku 6.5.



Rys. 6.6. Google Maps i wybór miejsca docelowego

Na rysunku 6.6 widzimy mapę z Google Maps, która otwiera się po kliknięciu przycisku **Wybierz miejsce docelowe**. Korzystając z niej użytkownik może wpisać adres miejsca docelowego co będzie skutkowało wyznaczeniem przez Google Maps trasy z obecnej lokalizacji do wyznaczonego celu.



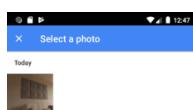
Rys. 6.7. Zakładka Zdjęcia

Po wybraniu z menu bocznego opcji **zdjęcia** użytkownik zostanie przeniesony na stronę, która jest pokazana na rysunku 6.7. Znajdują się tu dwa przyciski **zrób zdjęcie** oraz **wybierz zdjęcie**.



Rys. 6.8. Aparat

Po kliknięciu przycisku **zrób zdjęcie** otwarty zostanie aparat w telefonie i możliwe będzie wykonanie zdjęcia co zostało pokazane na rysunku 6.8. Wykonane zdjęcie zostanie wyświetcone na stronie w miejscu nad przyciskami.



Rys. 6.9. Wybór zdjęcia z galerii

Na rysunku 6.9 pokazany jest widok po kliknięciu przycisku **wybierz zdjęcie**. Otwarta zostaje galeria telefonu i możliwe jest wybranie dowolnego wcześniej wykonanego zdjęcia które następnie zostanie wyświetcone na stronie.

Bibliografia

- [1] Strona internetowa firmy Microsoft opisująca Xamarin Forms. URL: <https://docs.microsoft.com/pl-pl/xamarin/get-started/what-is-xamarin-forms> (term. wiz. 06.11.2021).
- [2] Strona internetowa firmy Microsoft opisująca Xamarin Essentials. URL: <https://docs.microsoft.com/pl-pl/xamarin/essentials/?context=xamarin/xamarin-forms> (term. wiz. 06.11.2021).
- [3] Strona internetowa firebase Google. URL: <https://firebase.google.com/docs/auth> (term. wiz. 12.01.2021).

Spis rysunków

2.1. podstawowy layout aplikacji	4
2.2. layout aplikacji po otwarciu menu	5
3.1. Layout - Mapa	7
3.2. Layout - Wyniki	8
3.3. Layout - Zdjęcia	8
3.4. Layout - Wybierz normę	9
3.5. Odznaczenie właściwości w Opcjach systemu Android	9
3.6. Archiwizacja rozwiązania dla systemu Android	10
3.7. Dystrybucja pliku	10
3.8. Wybranie kanału dystrybucji	11
3.9. Podpis za pomocą aliasu	11
4.1. Dodanie strony Menu do menu bocznego	12
4.2. Widok menu bocznego	13
4.3. Widok strony otwartej po wybraniu danej opcji z menu	13
4.4. Dodanie opcji z polami do wyboru	14
4.5. Dodanie nowych deklaracji w AndroidManifest.xaml	14
4.6. Dodanie funkcji w mapa.xaml.cs	15
4.7. Ustawienie opcji w mapa.xaml	15
4.8. Plik Zdjecia.xml	15
4.9. Plik Zdjecia.xml.cs	16
4.10. Plik AndroidManifest.xml	16
4.11. Inicjalizacja w pliku MainActivity.cs	17
4.12. Interfejs Iauth w pliku Iauth.cs	17
4.13. Funkcja rejestracji	18
4.14. Funkcja logowania	18
4.15. Funkcja IsSignIn	19
4.16. Funkcja SignOut	19
4.17. Przekierowanie w pliku App.xaml	20
5.1. Tworzenie nowego konta	21
5.2. Informacja o utworzeniu konta	21
5.3. Nowe konto w bazie danych	22

5.4.	Logowanie na utworzone konto	22
5.5.	Widok po zalogowaniu	23
5.6.	Wyniki	23
5.7.	Mapa z lokalizacją	24
5.8.	Google Maps	24
5.9.	Wykonanie zdjęcia	25
5.10.	Wyświetlenie zdjęcia	25
5.11.	Wybieranie zdjęcia z galerii	26
5.12.	Nowe zdjęcie	26
6.1.	Strona startowa	27
6.2.	Strona rejestracji	27
6.3.	Strona główna	28
6.4.	Widok po włączeniu menu bocznego	28
6.5.	Mapa	29
6.6.	Google Maps i wybór miejsca docelowego	29
6.7.	Zakładka Zdjęcia	30
6.8.	Aparat	30
6.9.	Wybór zdjęcia z galerii	30

Spis tabel