POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: INFORMATYKA (INF)

SPECJALNOŚĆ: SYSTEMY INFORMATYKI W MEDYCYNIE (IMT)

PRACA DYPLOMOWA INŻYNIERSKA

Projekt i realizacja aplikacji dla trenera personalnego w klubie fitness.

An application for a personal trainer in a fitness club.

AUTOR:

Robert Gmerski

PROWADZĄCY PRACĘ:

Mgr inż., Karol Puchała, W-4/K-2

OCENA PRACY:

WROCŁAW, 2019

Spis treści

Spis listingów

Spis rysunków

1. Wstęp

1.1. Wprowadzenie

Celem pracy jest zaprojektowanie i wykonanie aplikacji na urządzenia mobilne z systemem operacyjnym Android. Aplikacja realizuje funkcjonalności związane ze wspomaganiem działalności trenera personalnego w klubie fitness. Aplikacja z założenia miała być jak najprostsza, aby każdy był w stanie z niej skorzystać. Została ona napisana w języku C# korzystając ze środowiska Xamarin. Ćwiczenia przechowywane są w bazie danych utworzonej przy pomocy Entity Framework. Dana baza pozwala trenerowi wygenerować automatycznie plan treningowy. Dodatkowo każde ćwiczenie w bazie ma pole z linkiem, do którego trener ma dostęp. W trakcie wykonywania ćwiczenia może on sprawdzić, czy ćwiczenie jest wykonane prawidłowo. Symulacje zostały wykonane na systemie Android w wersji 8.1 (Oreo).

Zawartość pracy została przedstawiona poniżej:

- opis instalacji wymaganych komponentów (np. symulatora, wymagane paczki w NuGET),
- analiza wymaganych w użyciu widoków (np. przyciski),
- hierarchia stron,
- kod źródłowy stron,
- zaimplementowanie widoków oraz funkcji na stronach,
- testy na symulatorze,
- dostosowanie rekordów wybieranych z bazy biorąc pod uwagę utworzony profil,
- losowanie ćwiczeń (sugerowanych dla trenera) na odpowiednie części ciała,
- dołączenie filmów video pokazujących poprawne wykonanie ćwiczenia.

1.2. Środowisko Xamarin

Xamarin polega głównie na podzieleniu całej aplikacji na strony. Strona jest możliwym do osiągnięcia widokiem. Aby lepiej to przedstawić posłużmy się przykładem. Istnieje aplikacja, która po włączeniu wyświetla kilka przycisków. Jeden z nich jest przyciskiem zawierającym tekst „Settings”. Po kliknięciu tego przycisku pokazują nam się różne opcje. W rzeczywistości następuje przejście do kolejnej strony, która posiada swoje metody oraz widoki. Każda strona jest złożona z dwóch plików o rozszerzeniach .xaml oraz .xaml.cs, które służą odpowiednio do umiejscowienia widoków w korzystając z języka XAML oraz do stworzenia tzw. code-behind, czyli funkcjonalności owych widoków oraz samej w sobie stronie. Przechodzenie między stronami odbywa się przez klasę Navigation, która dodaje na stos strony i dokonuje odpowiednich, wymaganych edycji stosu.

2. Funkcjonalności

Aplikacja głównie opiera się na generowaniu planu z ćwiczeń zawartych w bazie. Dodatkowe funkcje są potrzebne do jego spersonalizowania jak i ułatwiające pracę.

2.1. Opis funkcji

2.1.1. Dodawanie, usuwanie, edytowanie profilu

Aby poprawnie korzystać z aplikacji potrzebny jest profil. Ta funkcja jest dostępna ze strony MenuPage, przechodząc na stronę ProfilePage. Po wpisaniu danych lub przyciśnięciu przycisku „Delete” zmieniana jest zawartość pliku tekstowego, do którego są zapisywane dane profilu. Plik ten jest ukryty. Później stwarzają się widoki na stronie profilu.

Listing. 1. Kod odpowiadający za zapisywanie profilu

```
private void BTN_Save_Clicked(object sender, EventArgs e)
{
    if (!File.Exists(_filename))
    {
        try
        {
            using (StreamWriter sw = File.CreateText(_filename))
            {
                sw.WriteLine(edit_name.Text);
                sw.WriteLine(edit_age.Text);
                sw.WriteLine(edit_weight.Text);
                sw.WriteLine(PCK_Train.Title);
            }
        }
        catch (FileNotFoundException ex)
        {
            edit_name.Text = ex.Message;
            edit_age.Text = ex.Message;
            edit_weight.Text = ex.Message;
            PCK_Train.Title = ex.Message;
        }
        BTN_Save.IsEnabled = false;
        BTN_Update.IsEnabled = true;

        edit_name.IsReadOnly = true;
        edit_age.IsReadOnly = true;
        edit_weight.IsReadOnly = true;
        PCK_Train.IsEnabled = false;
    }
}
```

Listing. 2. Usuwanie profilu

```
private void BTN_Del_Clicked(object sender, EventArgs e)
{
    if (File.Exists(_filename))
    {
        File.Delete(_filename);
    }
    edit_name.Text = string.Empty;
    edit_age.Text = string.Empty;
    edit_weight.Text = string.Empty;
    PCK_Train.Title = "Training type";

    BTN_Save.IsEnabled = true;
    BTN_Update.IsEnabled = false;

    edit_name.IsReadOnly = false;
    edit_age.IsReadOnly = false;
    edit_weight.IsReadOnly = false;
    PCK_Train.IsEnabled = true;

}
```

Tutaj można zauważyć, jaka jest różnica w widokach między usuwaniem a zapisywaniem profilu. Przy usuwaniu flaga IsReadOnly lub IsEnabled jest ustawiana na false dla usuwania, zaś na true dla zapisywania. Dzięki temu od razu po kliknięciu przycisku „Delete” można wprowadzić od nowa profil, zaś po wciśnięciu przycisku „Save” nie można zmienić wartości pól tekstowych, chyba że zostanie wciśnięty przycisk „Update”. On sam jest wyłączany, gdyż nie można edytować profilu, który nie istnieje. Analogicznie odbywa się to dla dodawania profilu.

Listing. 3. Edytowanie profilu

```
private void BTN_Update_Clicked(object sender, EventArgs e)
{
    if (File.Exists(_filename))
    {   // pierwsze klikniecie umozliwia edytowanie pol edytorow, drugie zapisuje
        upd_clicked++;

        if (upd_clicked == 1)
        {
            edit_age.IsReadOnly = false;
            edit_weight.IsReadOnly = false;
            PCK_Train.IsEnabled = true;
        }

        string old_name;
        using (StreamReader sr = new StreamReader(_filename))
        {
            old_name = sr.ReadLine();
        }

        if (upd_clicked == 2)
        {
            try
            {
                using (StreamWriter sw = File.CreateText(_filename))
                {
                    sw.WriteLine(old_name);
                    sw.WriteLine(edit_age.Text);
                    sw.WriteLine(edit_weight.Text);
                    sw.WriteLine(PCK_Train.Title);
                }
            }
            catch (FileNotFoundException ex)
            {
                edit_name.Text = ex.Message;
                edit_age.Text = ex.Message;
                edit_weight.Text = ex.Message;
            }

            upd_clicked = 0;
            BTN_Update.IsEnabled = false;
        }
    }
}
```

W przypadku edycji profilu sprawa jest trochę bardziej skomplikowana. Aby uniknąć niechcianych komplikacji między działaniami to potrzebny jest licznik kliknięć w przycisk. Najpierw jest przyciśnięty raz, aby zmienić wymagane pola tekstowe na edytowalne. Drugie kliknięcie powoduje zapisanie danych zmian. Została usunięta opcja zmiany nazwy, gdyż jest to niepotrzebne. Więc najpierw nazwa jest odczytywana z pliku i zapisana do zmiennej. Później razem z innymi zmienionymi wartościami jest wpisywana do pliku tekstowego.

2.1.2. Dodawanie i usuwanie ćwiczeń (dla trenera) oraz przeglądanie ćwiczeń

Następną funkcją jest dodawanie ćwiczeń do bazy. Jest to dostępne na stronie ExercisePage, do której jest dostęp ze strony MenuPage poprzez wciśnięcie odpowieniego przycisku. Została podjęta decyzja, że będzie to funkcja zamknięta, dostępna tylko dla trenera dla bezpieczeństwa bazy. Z tego powodu również została dodana opcja wypełnienia bazy danymi ćwiczeniami, gdyż raz z powodu błędu symulatora została utracona cała baza dlatego, że wszystkie pliki są zapisywane lokalnie na urządzeniu.

Listing. 4. Dodanie ćwiczenia

```
private async void Save_BTN_Clicked(object sender, EventArgs e)
{
    string name = Name_ed.Text;
    string muscles = Muscles_ed.Text;
    string equipment = Equipment_ed.Text;
    int potok = int.Parse(Potok_ed.Text);
    string type = PCK_Train.Title;
    string link = Uri_ed.Text;

    Rep rep = new Rep
    {
        Name = name,
        Muscles = muscles,
        Equipment = equipment,
        Potok = potok,
        Type = type,
        Link = link
    };

    await App.Database.SaveRepAsync(rep);

    await Navigation.PopAsync();
}
```

Tutaj również widać, jak korzystając z klasy Navigation aplikacja cofa do poprzedniej strony i usuwa aktualną ze stosu.

Listing. 5. Usuwanie ćwiczenia

```
private async void Del_BTN_Clicked(object sender, EventArgs e)
{
    var rep = (Rep)BindingContext;
    await App.Database.DeleteRepAsync(rep);
    await Navigation.PopAsync();
}
```

Korzystając z BindingContext, który jest ćwiczeniem przekazywanym na stronę usuwa się dany rekord z bazy w bardzo prosty sposób. Żeby zrozumieć lepiej ten fragment trzeba dodać w jaki sposób działa BindingContext. Żeby usunąć ćwiczenie potrzeba kliknąć na dane ćwiczenie. Na tej stronie będzie dostępny przycisk „Delete”.

Same dodanie gotowych ćwiczeń działa podobnie jak dodanie własnego, tylko zamiast wczytywania danych ze strony przekazuje się gotowe dane. Same przeglądanie ćwiczeń jest dostępne na stronie z przyciskami do dodawania/zapełnienia bazy danych. Dodatkowo, jeśli któreś ćwiczenie zostanie naciśnięte to można zobaczyć szczegóły oraz możliwość usunięcia jego.

2.1.3. Tworzenie i usuwanie planu

Mając profil użytkownika oraz ćwiczenia pozostało utworzenie planu. Ta funkcja, podobnie jak edycja profilu jest dostępna ze strony ProfilePage. W zależności od typu treningu podanego w profilu będą generowane plany. Kod generujący plany jest w miarę długi, więc będzie on pominięty.

2.1.4. Trening

Po wygenerowaniu planu należy przejść do treningu. Jest on dostępny z ekranu głównego aplikacji. Na stronie treningu dostępne są nowe funkcje.

2.1.4.1. Wybór dnia

Można wybrać jeden z trzech dni, na które podzielony jest plan treningowy. Każdy dzień posiada swoje ćwiczenia. Pobierając ćwiczenia z planu przekazywane są na stronę: nazwa oraz link do pokazowego video z poprawnym wykonaniem. Aby obejrzeć video wymagane jest wciśnięcie odpowiedniego przycisku.

Listing. 6. Przycisk pierwszego dnia

```
private async void BTN_1_Clicked(object sender, EventArgs e)
{
    BTN_1.IsEnabled = false;
    BTN_2.IsEnabled = false;
    BTN_3.IsEnabled = false;

    Next_BTN.IsEnabled = true;

    which = 1;
    number = 0;
    using (StreamReader sr = new StreamReader(_plan1))
    {
        string line = sr.ReadLine();
        int ID = int.Parse(line);
        Rep temp = await App.Database.GetRepAsync(ID);
        Exer_name.Text = temp.Name;
        type = temp.Type;

        currUri = new Uri(temp.Link);
    }
    number++;
}
```

Dla pozostałych dni kod wygląda podobnie, różni się tylko numer dnia (np. do zmiennej which dla dnia drugiego przypisana jest liczba 2, zaś zamiast zmiennej _plan1 do StreamReader przekazuje się zmienną _plan2).

2.1.4.2. Video z przykładowym wykonaniem ćwiczenia

Listing. 7. Otwieranie linku

```
private void Link_BTN_Clicked(object sender, EventArgs e)
{
    Device.OpenUri(currUri);
}
```

2.1.4.3. Zmiana ćwiczenia

Po odbytym ćwiczeniu po przyciśnięciu przycisku „Next” program przechodzi do następnego ćwiczenia z danego dnia.

Listing. 8. Przejście do następnego ćwiczenia

```
private async void Next_BTN_Clicked(object sender, EventArgs e)
{
    if (type == "Str")
    {
        if (number > 6)
        {
            Exer_name.Text = "Congratulations! Your strenght training has ended.";
            Next_BTN.IsEnabled = false;

            BTN_1.IsEnabled = true;
            BTN_2.IsEnabled = true;
            BTN_3.IsEnabled = true;

            goto A;
        }
    }
    else
    {
        if (number > 4)
        {
            Exer_name.Text = "Congratulations! Your cardio training has ended.";
            Next_BTN.IsEnabled = false;

            BTN_1.IsEnabled = true;
            BTN_2.IsEnabled = true;
            BTN_3.IsEnabled = true;

            goto A;
        }
    }
    string line;
    switch (which)
    {
        case 1:
            using (StreamReader sr = new StreamReader(_plan1))
            {

                for (int i = 0; i < number; i++)
                {
                    line = sr.ReadLine();
                }
                line = sr.ReadLine();
                int ID = int.Parse(line);
                Rep temp = await App.Database.GetRepAsync(ID);
                Exer_name.Text = temp.Name;
                currUri = new Uri(temp.Link);
                number++;
            }
            break;
        case 2:
```

```
            using (StreamReader sr = new StreamReader(_plan2))
            {
                for (int i = 0; i < number; i++)
                {
                    line = sr.ReadLine();
                }
                line = sr.ReadLine();
                int ID = int.Parse(line);
                Rep temp = await App.Database.GetRepAsync(ID);
                Exer_name.Text = temp.Name;
                currUri = new Uri(temp.Link);
                number++;
            }
            break;
        case 3:
            using (StreamReader sr = new StreamReader(_plan3))
            {
                for (int i = 0; i < number; i++)
                {
                    line = sr.ReadLine();
                }
                line = sr.ReadLine();
                int ID = int.Parse(line);
                Rep temp = await App.Database.GetRepAsync(ID);
                Exer_name.Text = temp.Name;
                currUri = new Uri(temp.Link);
                number++;
            }
            break;
        default:
            break;
    }
A:;
}
```

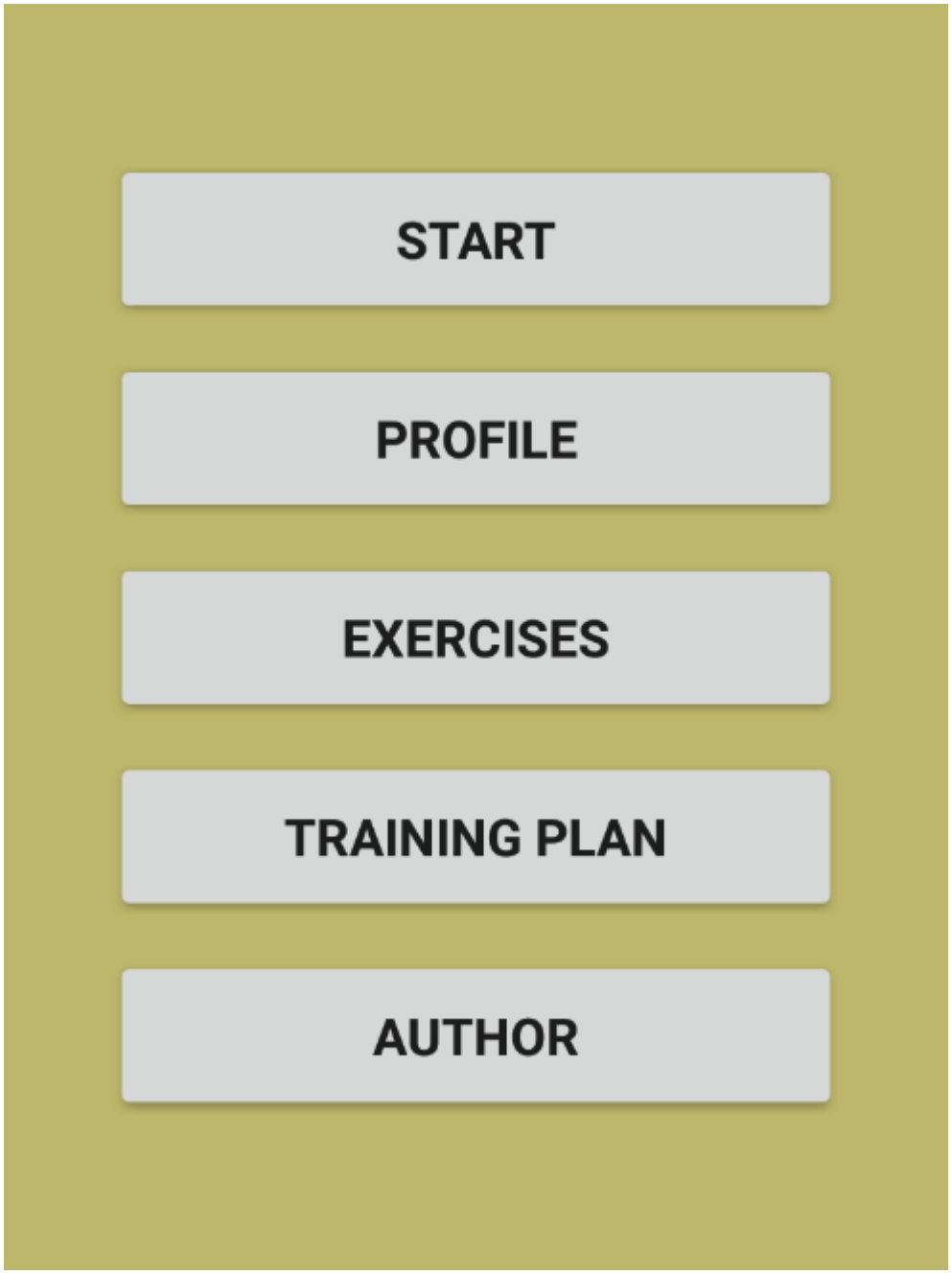
2.1.4.4. Timer

Ostatnią funkcją w aplikacji jest timer. Odlicza on dany czas, domyślnie jest to 120 sekund. Służy to do odliczania przerw między seriami.

Listing. 9. Timer

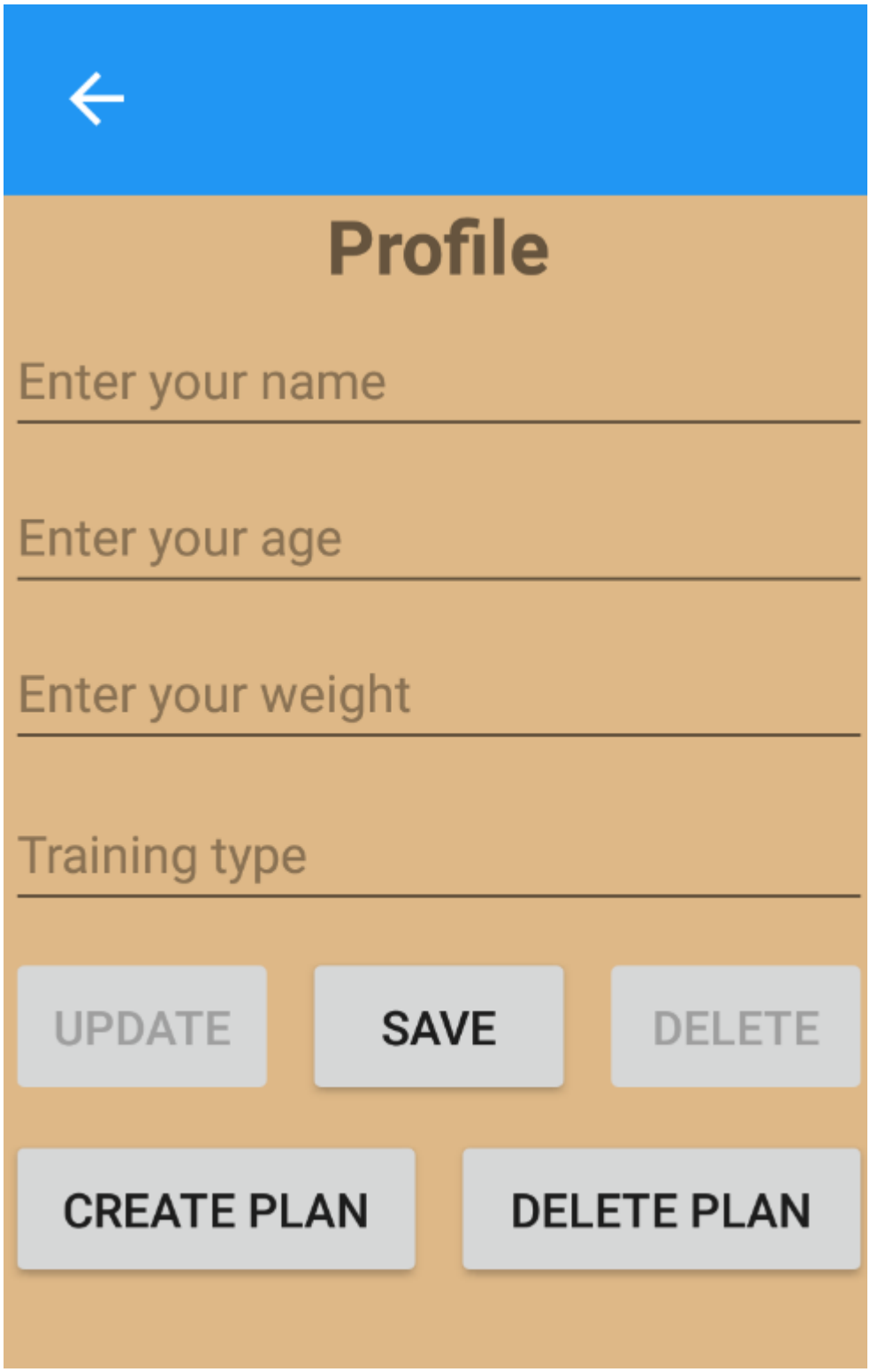
```
private void BTN_Timer_Clicked(object sender, EventArgs e)
{


    if (int.TryParse(Timer_ed.Text, out _))
    {
        lenght = int.Parse(Timer_ed.Text);

    }
    else
    {
        Timer_ed.Text = "NaN";
        lenght = 120;
    }
    time = lenght;
    Device.StartTimer(TimeSpan.FromSeconds(1), () =>
    {
        BTN_Timer.IsEnabled = false;
        time--;
        Timer.Text = time.ToString();
        if (time < 1)
        {
            BTN_Timer.IsEnabled = true;
            return false;
        }
        else return true;
    });
    BTN_Timer.IsEnabled = true;
}
```

2.2. Korzystanie z aplikacji przez użytkownika



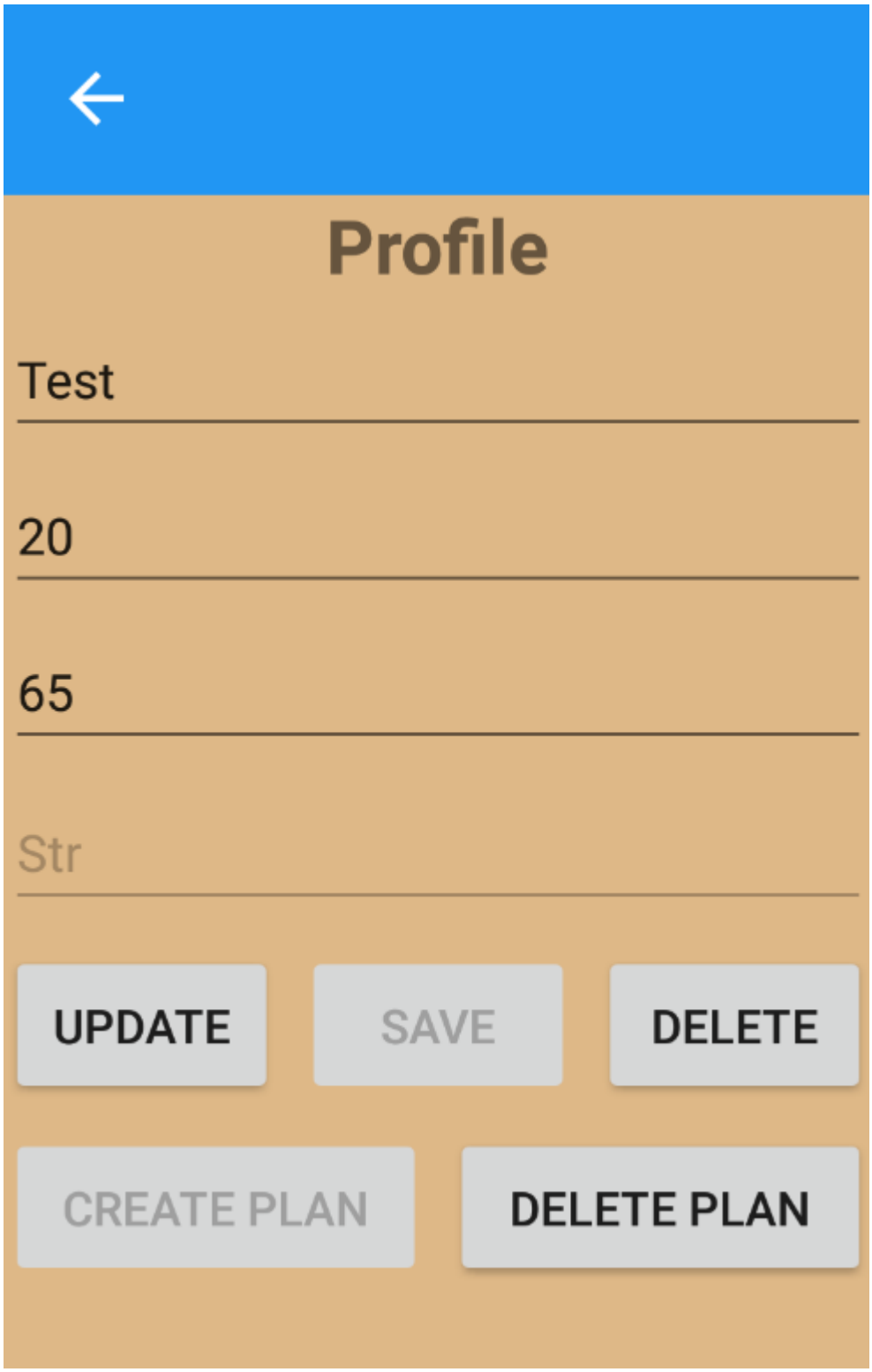
Rys. 1. Menu główne aplikacji

Użytkownik po włączeniu aplikacji musi stworzyć profil, jeśli jeszcze nie jest on utworzony. Aby to zrobić potrzebne jest wciśnięcie przycisku „Profile”. Później pokaże się strona z polami tekstowymi oraz jednym polem wyboru. Trzeba je uzupełnić.



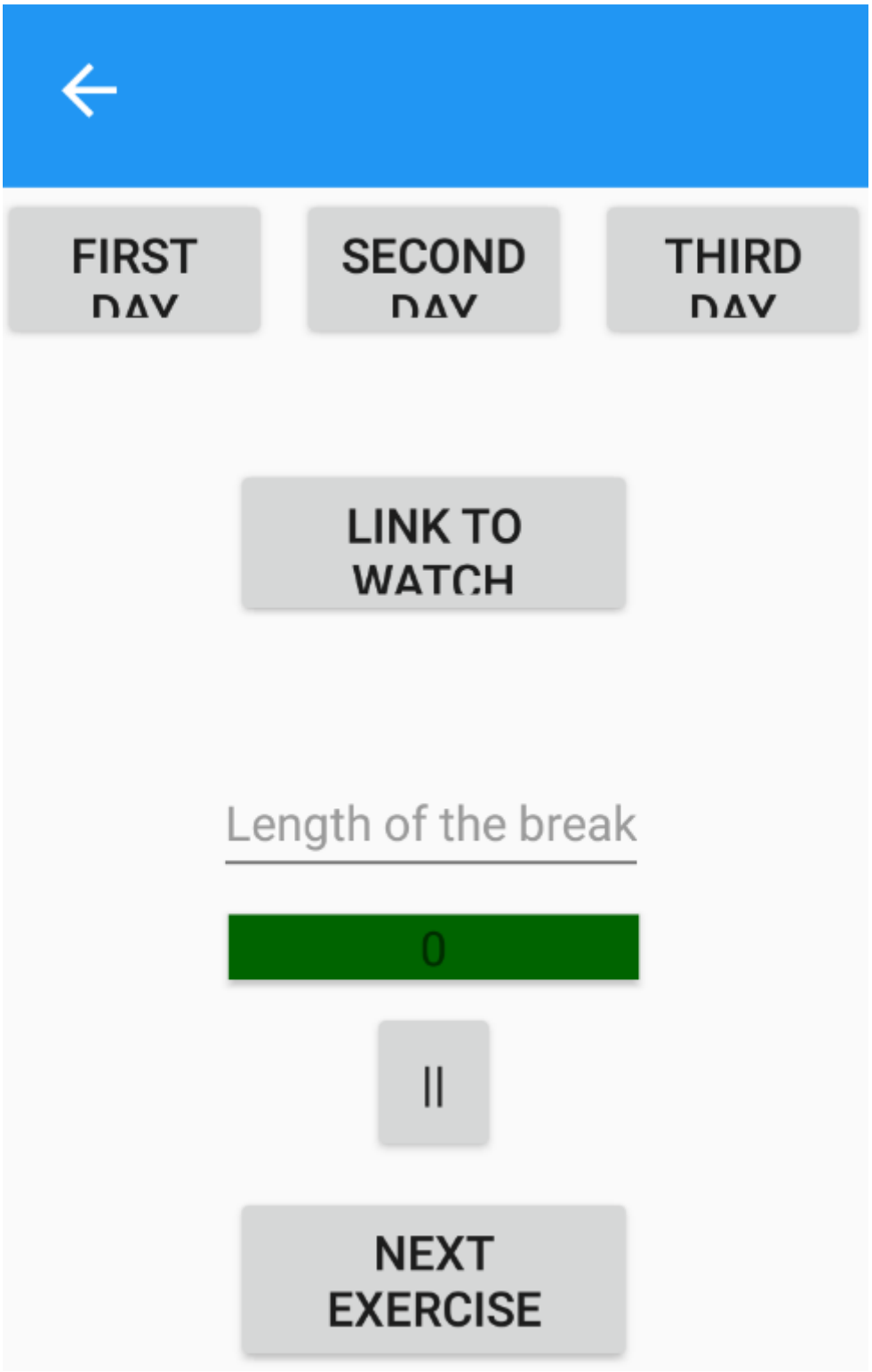
Rys. 2. ProfilePage (puste)

Po uzupełnieniu danych można utworzyć plan naciskając na „Create plan”.



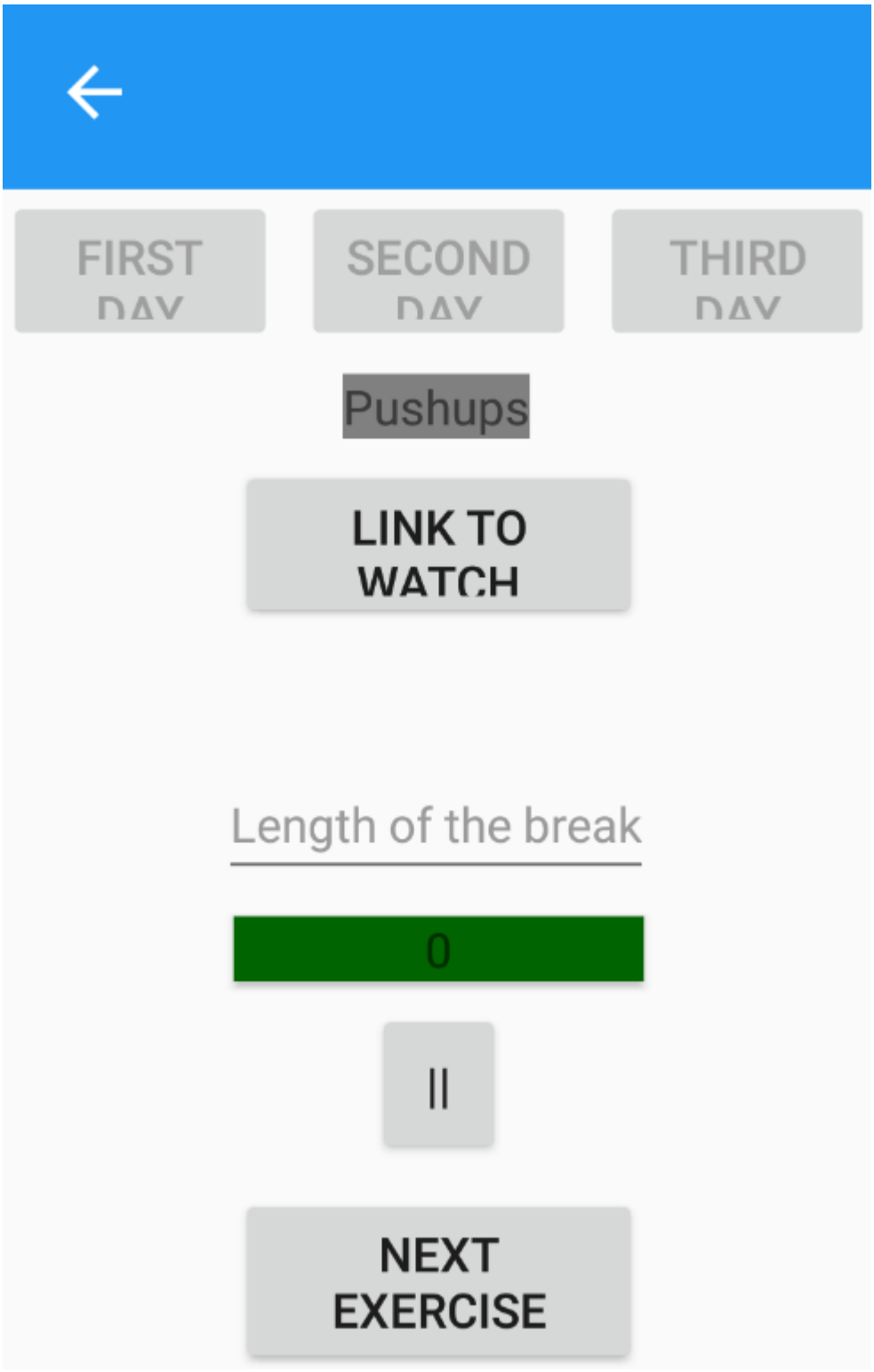
Rys. 3. ProfilePage (uzupełnione)

Następnie można już przejść do treningu, lub sprawdzić jak wygląda nasz plan. Można również z ciekawości sprawdzić, jakie ćwiczenia są dostępne w bazie.



Rys. 4. MainPage

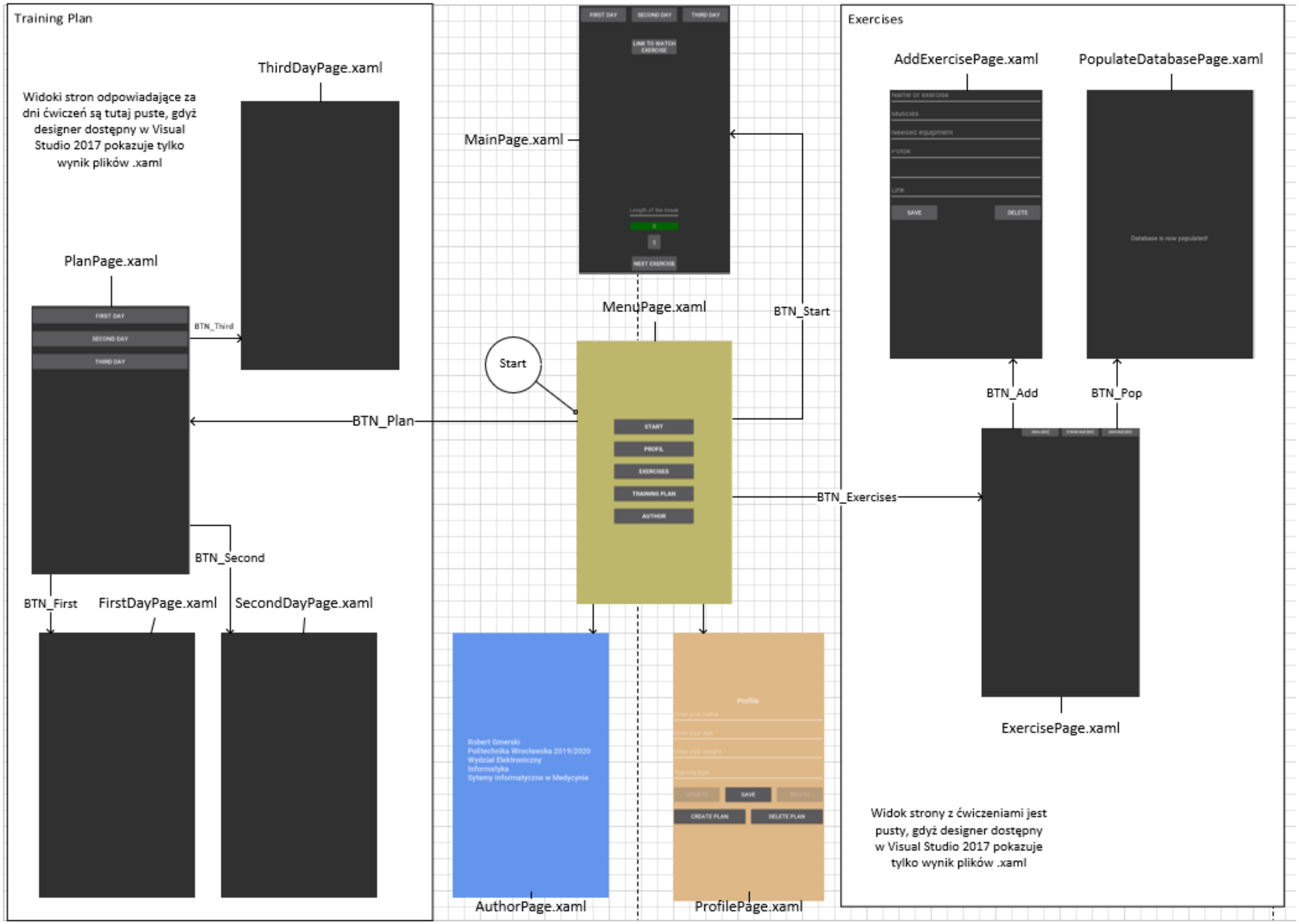
Tak wygląda strona główna aplikacji. Po wyborze dnia treningowego zmienia się zawartość strony. W polu, na którym widnieje napis „Length of the break” można wpisać, ile sekund ma trwać przerwa. Po wciśnięciu przycisku „||” aplikacja odlicza dany okres.



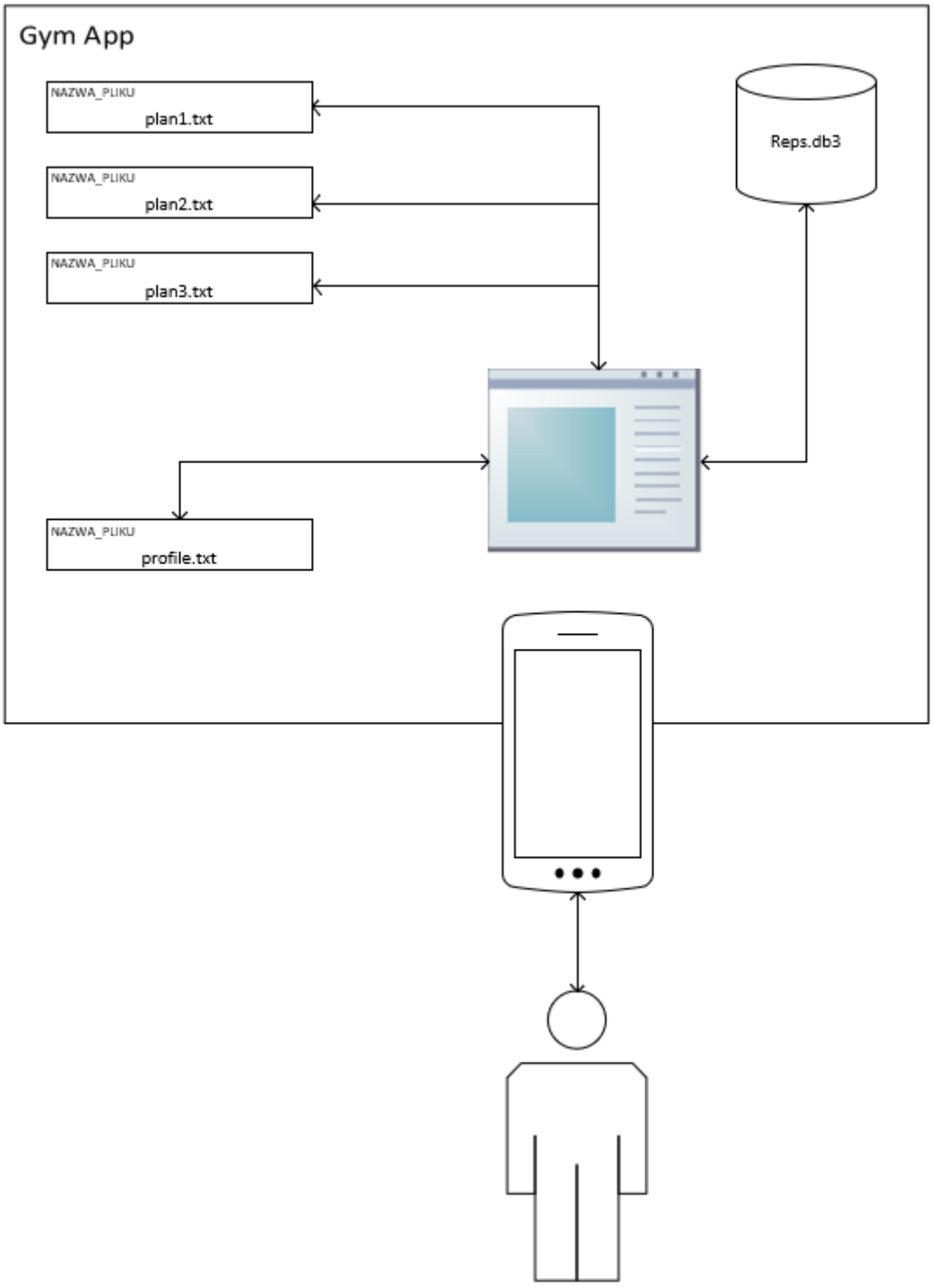
Rys. 5. MainPage po wyborze dnia

Po wyborze dnia pokaże się pierwsze ćwiczenie danego dnia. Po wciśnięciu „Next exercise” ćwiczenie zostanie zmienione na następne, a gdy skończą się ćwiczenia danego dnia to wyświetla się na miejsce ćwiczenia komunikat.

3. Architektura aplikacji



Rys. 6. Spis stron z przejściami



Rys. 7. Architektura aplikacji

Aplikacja łączy się z bazą danych i może ją edytować, tak jak i z plikami tekstowymi plan1.txt, plan2.txt, plan3.txt (przechowujące ćwiczenia z konkretnych dni) oraz z profile.txt.

4. Podsumowanie

Przy projektowaniu aplikacji głównym problemem było dodawanie ćwiczeń w trakcie pracy programu. Zdecydowano się na dodawanie elementów w kodzie po utraceniu bazy. Kolejnym problemem była kwestia odtwarzania filmów z portalu YouTube. Potrzebne było specjalne konto developerskie, aby móc używać Google API. To spowodowało zmianę sposobu wyświetlania. Zamiast wbudowanego odtwarzacza został zaprogramowany przycisk umożliwiający funkcjonalność przekierowywania do serwisu YouTube.

Udało się zaimplementować wszystkie założone funkcjonalności poszerzając je o minutnik. Środowisko aplikacji pozwala na łatwą rozbudowę jej, gdyż wystarczy stworzyć nowe strony oraz połączyć je z aktualnymi.

Literatura

[1] Microsoft. C# documentation. https://docs.microsoft.com/pl-pl/dotnet/csharp/ [dostęp dnia 27 listopada 2019].

[2] Microsoft. Xamarin.Forms documentation. https://docs.microsoft.com/en-gb/xamarin/xamarin-forms/ [dostęp dnia 27 listopada 2019].

[3] George Taskos „Xamarin Cross Platform Development Cookbook". Tłum. Andrzej Watrak, wyd. Helion, 2017.

[4] Steven F. Daniel „Mastering Xamarin UI Development". Tłum. Łukasz Piwko, wyd. Helion, 2017.

[5] Johan Karlsson „Xamarin.Forms Projects". Wyd. Packt Pub, 2018.

Źródła

https://www.bodybuilding.com/
https://www.youtube.com/channel/UCDkn7Mk9el92pDJy2HaFcpw
https://www.youtube.com/channel/UC97k3hlbE-1rVN8y56zyEEA
https://www.youtube.com/channel/UCxOUVqxPjk0PJ7usJhD6quA
https://www.youtube.com/channel/UCRnQ0oe-NLWHPt0UUqoDcNQ
https://www.youtube.com/user/Howcast
https://www.youtube.com/channel/UCKf0UqBiCQI4Ol0To9V0pKQ
https://www.youtube.com/channel/UCQ_P2_JOH4uHeUZlHDL-jCA
https://www.youtube.com/channel/UCEtMRF1ywKMc4sf3EXYyDzw
https://www.youtube.com/channel/UCBbB-PR9CsGcMnswP04TbhQ
https://www.youtube.com/channel/UCzBG5GOe0b07dEWd6W_7l6w
https://www.youtube.com/channel/UCmSEdfW3LpEKyLiCDWBDdVQ
https://www.youtube.com/channel/UCtt_c8PMrt0ZEE-LF8omc2A
https://www.youtube.com/channel/UCQfaOHrgP_QGU-E4Anx_4jA
https://www.youtube.com/channel/UCo0du-IzWuYaVf9QTg10nAQ
https://www.youtube.com/channel/UCVePF6PImEVTHJybNLdDNYA
https://www.youtube.com/channel/UC3bSdAP92Myv1hsxrclt2Tg
https://www.youtube.com/channel/UC3-hVFObb2ZEcdmfP9-Us_Q
https://www.youtube.com/channel/UCHJuQZuzapBh-CuhRYxIZrg
https://www.youtube.com/channel/UCBINFWq52ShSgUFEoynfSwg
https://www.youtube.com/watch?v=uymTGsML0HA

ScottHermanFitness
Musqle
Howcast
MyTraining App
Bodybuilding.com
AMOFitnessTraining
KAGED MUSCLE
Buff Dudes
PureGym
Workout Generator
Mountain Dog
John Garey TV
My PT Hub
Muscle & Motion
Aztec Recreation
GHF Training
30 Day Fitness Challenges
Insider
POPSUGAR Fitness
WatchItNow TV