AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Aplikacja Lekoman

Autor:
Gabriela Zborowska
Nikodem Tyc
Marcin Podobiński

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

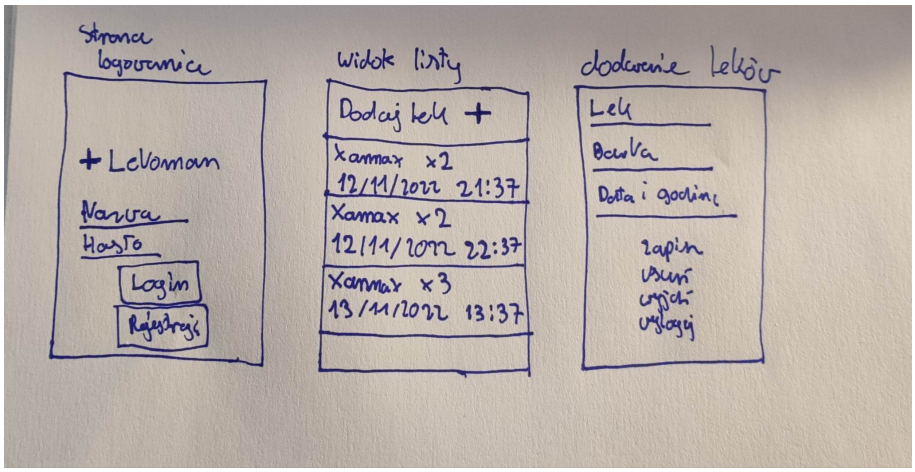
Spis treści

1. Ogólne określenie wymagań

Aplikacja powinna być łatwa w obsłudze dla osób w każdym wieku. Użytkownik będzie mógł zwiększyć rozmiar fontu oraz zmienić tryb na nocny. Dodatkowo użytkownik dopiero po zalogowaniu będzie mógł zobaczyć leki oraz dawkowanie. Będzie mógł na bieżąco wprowadzać zmiany, a w razie czego usunąć dany wpis. Na jego telefon będą przychodzić powiadomienia push zgodnie z ustalaną przez niego częstotliwością. Po zażyciu leku użytkownik będzie mógł zaznaczyć, że dawka została przez niego przyjęta.

Wygląd aplikacji powienien być przyjazny dla oka, czcionka czytelna. Brak zbędnych informacji, wszystko przejrzyste. Okno rejestracji i logowania powinno się otwierać zaraz po załadowaniu aplikacji, bez zbędnego klikania. Po rejestracji konta powinny zostać zapisane dane logowania użytkownika.

Layout aplikacji ma się prezentować mniej więcej w ten sposób:



Rys. 1.1. Poglądowa wizja Klienta jak aplikacja ma wyglądać.

2. Określenie wymagań szczegółowych

Aplikacja będzie pozwalała na wprowadzanie leków oraz dawek.Aplikacja ma niebieski layout, przyjemny dla oka i nawiązujący do branży medycznej. Na ekranie startowym nasza aplikacja umożliwia okno do rejestracji nowego konta użytkownika, podczas której osoba musi podać swój email, ustalić login oraz hasło. Ze względu na bezpieczeństwo danych nasz software wymaga autoryzacji za pomocą loginu i hasła. Gdy podamy nieprawidłowe dane, wyświetli się komunikat: "Błędny login lub hasło!". Jest to gwarancja ochrony utworzonych kont użytowników przed nieuprawnionym dostępem. Poświadczenia użytkownika zostaną zapamiętane w wewnętrznej bazie danych wykorzystującej otwartoźródłowy system zarządzania relacyjną bazą danych SQLite.

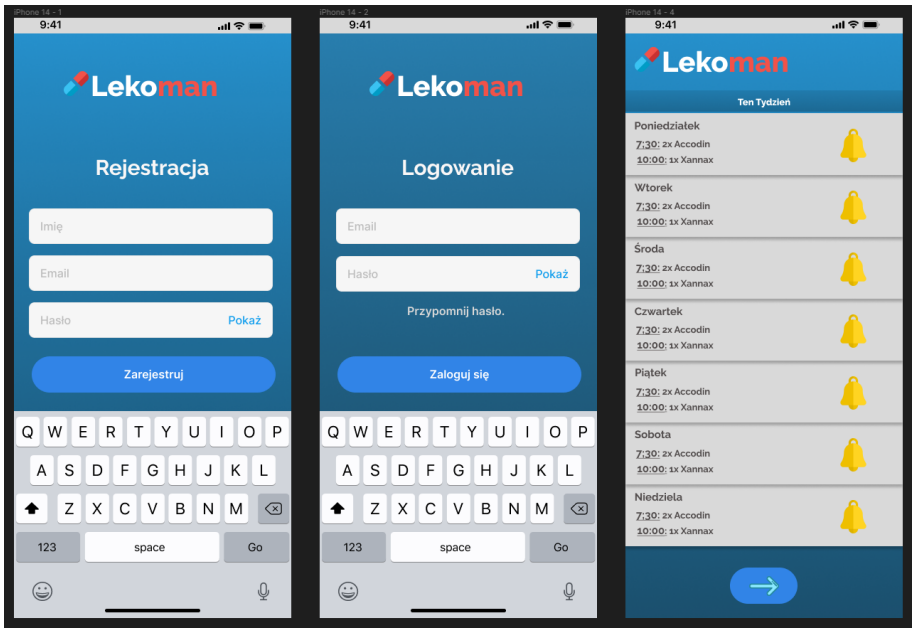
Po zalogowaniu użytkownika będzie mógł wprowadzić rodzaj leku oraz dawkę. Po ustaleniu tych aspektów klient może zapisać swoje ustawienia oraz przeciągnąć przycisk "Zrobione", a następnie "WYJDŹ". Również, gdy dawka będzie mylnie wprowadzona użytkownik ma możliwość usunięcia jej za pomoca przycisku "USUŃ". Po tej czynności pojawi się w oknie wprowadzania lekarstw zielona fiszka przy tej pozycji. Z tego poziomu zezwolone jest także bezpośrednie wylogowanie się z aplikacji, wciskając "WYLOGUJ".

Na telefon komórkowy będzie przychodzić powiadomienie push zgodnie z ustaloną porą przed zażyciem leków. Powiadomienie push jest to jeden z czujników, który wykorzystaliśmy tworząc aplikację. Nie jesteśmy w stanie wdrożyć zmiany na tryb nocny.

Została wprowadzona funkcja kalendarza, która obrazuje użytkownikowi w pełni klarowny podgląd na wszystkie środki lecznicze, które powinien przyjmować. Aczkolwiek Klient nie może wprowadzić wcześniejszej daty niż aktualna, która synchroznizuje się z jego zegarem systemowym w telefonie.

3. Projektowanie

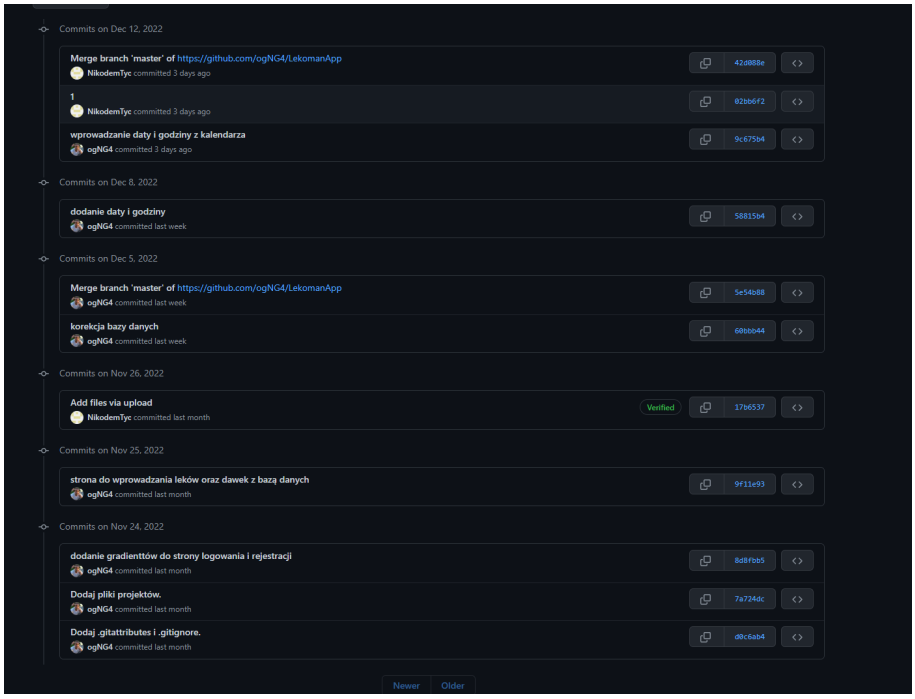
Projektowanie rozpoczynamy od zrobienia Layout'u naszej aplikacji. Został on przygotowany w programie *Figma*.



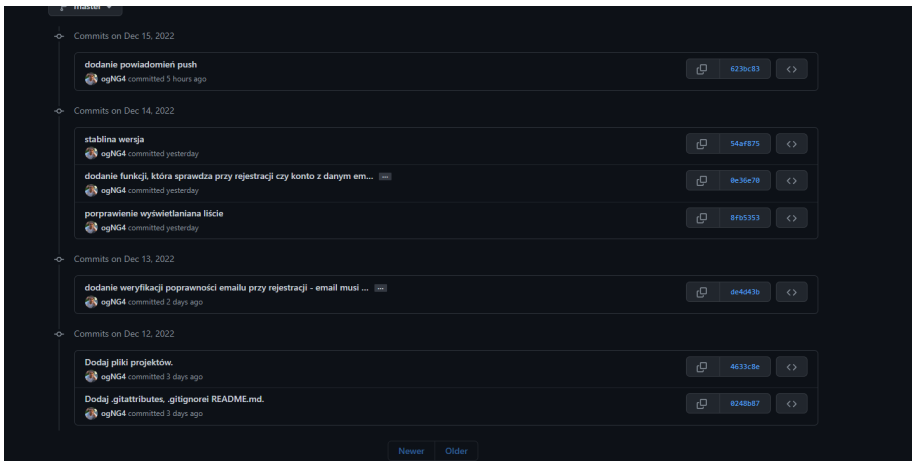
Rys. 3.1. Layout aplikacji - wykonany w programie Figma z perspektywy programisty

Kolejnym etapem tworzenia aplikacji było stworzenie możliwości rejestracji i logowania dla klienta. Udostępniliśmy patent na zapis informacji logowania użytkownika przez implementację *SQLite* w tym projekcie. Klientowi zezwala się na wprowadzenie dawek oraz rodzajów leków, również usuwianie tych wartości.

Zaimplementowaliśmy narzędzie *GitHub*, czyli hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujących system kontroli wersji *Git*. *GitHub* udostępnia darmowy hosting programów open source i prywatnych repozytoriów. Stworzyliśmy tam repozytorium, służące do commitowania i udostępniania całego projektu.

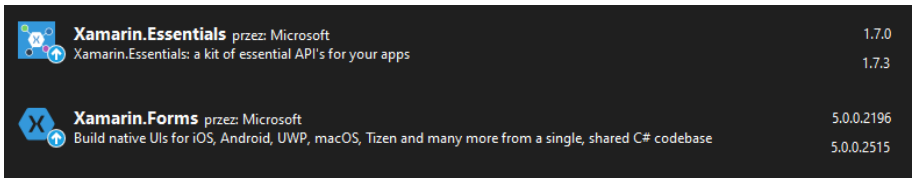


Rys. 3.2. Repozytorium - początki commitowania



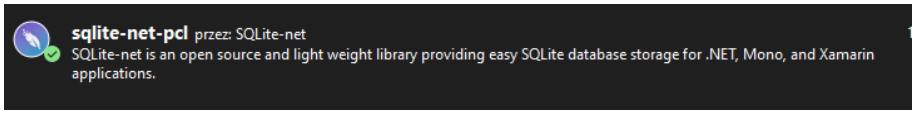
Rys. 3.3. Repozytorium - commitowania dalsza część

Aplikacja tworzona jest w języku C#. Do samego kodowania użyliśmy oprogramowanie *Microsoft Visual Studio*; instalując w nim emulator *Xamarin.Forms* i *Xamarin.Essentials*, wykorzystując system Android.



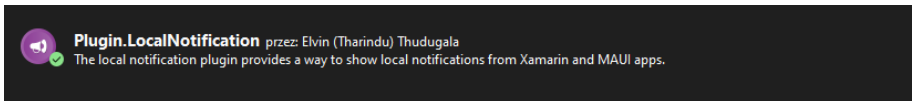
Rys. 3.4. Emulator Xamarin

Do zapisu rekordów danych wybraliśmy *SQLite* otwartoźródłowy system zarządzania relacyjną bazą danych.



Rys. 3.5. Pakiet SQLite

Zainstalowaliśmy także *Pakiet NuGet*, który udostępnia centralne repozytorium *nuget.org* z obsługą hostingu prywatnego i narzędzia do tworzenia, publikowania i używania pakietów.W naszym przypadku został on wykorzystany do wdrożenia powiadomień PUSH.



Rys. 3.6. Pakiet NuGet

4. Implementacja

```
using System;
using System.Runtime.CompilerServices;
using System.Threading.Tasks;

namespace LekomanApp
{
    public class AsyncLazy<T>
    {
        readonly Lazy<Task<T>>instance;

        public AsyncLazy(Func<T> factory)
        {
            instance = new Lazy<Task<T>>(() => Task.Run(factory));
        }

        public AsyncLazy(Func<Task<T>> factory)
        {
            instance = new Lazy<Task<T>>(() => Task.Run(factory));
        }

        public TaskAwaiter<T>GetAwaiter()
        {
            return instance.Value.GetAwaiter();
        }

        public void Start()
        {
            var unused = instance.Value;
        }
    }
}
```

Listing 1. Klasa *AsyncLazy.cs*

Klasa *AsyncLazy.cs* służy do asynchronicznego tworzenia obiektów typu T. Klasa ta opiera się na mechanizmie $Lazy < T >$, który oznacza, że obiekt typu T jest tworzony dopiero w momencie pierwszego jego użycia (tzw. lazy initialization). W przypadku $AsyncLazy < T >$ obiekt jest tworzony w sposób asynchroniczny, czyli w osobnym wątku. Klasa posiada dwa konstruktory. Pierwszy przyjmuje jako argument delegat $Func < T >$, czyli funkcję, która zwraca obiekt typu T. Drugi konstruktor przyjmuje jako argument delegat $Func < Task < T >>$, czyli funkcję zwracającą obiekt typu $Task < T >$. Klasa ta posiada też metodę *GetAwaiter()*, która zwraca obiekt typu $TaskAwaiter < T >$, umożliwiający asynchroniczne oczekiwanie na zakończenie działania obiektu typu $Task < T >$ oraz metodę *Start*(), która uruchamia tworzenie obiektu typu T w osobnym wątku.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LekomanApp
{
    public static class Constants
    {
        public const string DatabaseFilename = "Lekoman1SQLite.db3";

        public const SQLite.SQLiteOpenFlags Flags =
            SQLite.SQLiteOpenFlags.ReadWrite |
            SQLite.SQLiteOpenFlags.Create |
            SQLite.SQLiteOpenFlags.SharedCache;

        public static string DatabasePath
        {
            get
            {
                var basePath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
                return Path.Combine(basePath, DatabaseFilename);
            }
        }
    }
}
```

Listing 2. Klasa *Constants.cs*

Powyższy kod jest fragmentem klasy statycznej o nazwie *Constants.cs*, zdefiniowanej w przestrzeni nazw LekomanApp. Klasa ta zawiera kilka stałych, takich jak *DatabaseFilename* i *Flags*, które są używane w aplikacji do określenia nazwy i parametrów otwarcia pliku bazy danych. Klasa *Constants* zawiera również właściwość *DatabasePath*, która zwraca ścieżkę do pliku bazy danych na podstawie środowiskowego katalogu aplikacji oraz nazwy pliku bazy danych.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading.Tasks;
using LekomanApp.Models;
using LekomanApp;
using SQLite;

namespace LekomanApp.Data
{
    public class LekomanItemDatabase
    {
        static SQLiteAsyncConnection Database;

        public static readonly AsyncLazy<LekomanItemDatabase>
    Instance =
            new AsyncLazy<LekomanItemDatabase>(async () =>
            {
                var instance = new LekomanItemDatabase();
                try
                {
                    CreateTableResult result = await Database.
    CreateTableAsync<LekomanItem>();
                }
                catch (Exception ex)
                {
                    throw;
                }
                return instance;
            });

        public LekomanItemDatabase()
        {
            Database = new SQLiteAsyncConnection(Constants.
    DatabasePath, Constants.Flags);
        }

        public Task<List<LekomanItem>> GetItemsAsync()
        {
            return Database.Table<LekomanItem>().ToListAsync();
        }

        public Task<List<LekomanItem>> GetItemsNotDoneAsync()
        {
            return Database.QueryAsync<LekomanItem>("SELECT * FROM
```

```
    [LekomanItem] WHERE [Zrobione] = 0");
43          }
44
45          public Task<LekomanItem> GetItemAsync(int id)
46          {
47              return Database.Table<LekomanItem>().Where(i => i.ID ==
     id).FirstOrDefaultAsync();
48          }
49
50          public Task<int> SaveItemAsync(LekomanItem item)
51          {
52              if (item.ID != 0)
53              {
54                  return Database.UpdateAsync(item);
55              }
56              else
57              {
58                  return Database.InsertAsync(item);
59              }
60          }
61
62          public Task<int> DeleteItemAsync(LekomanItem item)
63          {
64              return Database.DeleteAsync(item);
65          }
66      }
67  }
```

Listing 3. Klasa *LekomanItemDatabase.cs*

Ten kod tworzy klasę *LekomanItemDatabase*, która służy do obsługi bazy danych *SQLite* zawierającej elementy typu *LekomanItem*. Klasa ta udostępnia asynchroniczne metody do pobierania, zapisywania i usuwania elementów z bazy danych, a także metody do pobierania wszystkich elementów lub tylko tych, które nie zostały oznaczone jako zrobione. Klasa ta używa *SQLiteAsyncConnection* do asynchronicznej komunikacji z bazą danych.

```
using SQLite;
using System.Linq;
using System;
using System.Collections.Generic;
using System.Text;

namespace LekomanApp.Models
{
    public class LekomanItem
    {
        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }

        public string Lek { get; set; }

        public string Dawka { get; set; }

        public DateTime Data { get; set; }

        public TimeSpan Godzina { get; set; }

        public bool Zrobione { get; set; }

        public Guid UserId { get; set; }
        public string UserName { get; set; }
        public string Password { get; set; }
        public string Email { get; set;}
    }
}
```

Listing 4. Klasa *LekomanItem.cs*

Ten kod tworzy klasę o nazwie *LekomanItem*, która jest używana w aplikacji LekomanApp do przechowywania informacji o lekach. Klasa posiada kilka właściwości, takich jak Lek, Dawka, Data i Godzina, które służą do gromadzeniu danych o leku, oraz Zrobione, które określa, czy dana dawka leku została już przyjęta. Klasa zawiera również właściwości dotyczące użytkowników aplikacji, takie jak UserId, UserName, Password i Email. Należy zauważyć, że klasa *LekomanItem* używa atrybutu *PrimaryKey* i *AutoIncrement* dla właściwości ID. To oznacza, że ta właściwość jest używana jako główny klucz dla tej klasy i jej wartość jest automatycznie zwiększana przy każdym dodaniu nowego elementu. Jest to powszechnie stosowane w systemach baz danych, w tym w bazie danych *SQLite*, która jest otwartoźródłowym systemem zarządzania relacyjną bazą danych oraz biblioteką *C* implementującą taki system, obsługująca *SQL*.

```
using LekomanApp.Models;
using SQLite;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace LekomanApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RegistrationPage : ContentPage
    {
        public RegistrationPage()
        {
            InitializeComponent();
        }
        void Button_Clicked(object sender, EventArgs e)
        {
            if (!EntryUserEmail.Text.Contains("@"))
            {
                Device.BeginInvokeOnMainThread(async () =>
                {
                    await this.DisplayAlert("Blad", "Wprowadz
   poprawny adres email.(Brakuje znaku '@')", "OK");
                });
                return;
            }

            if (string.IsNullOrEmpty(EntryUserPassword.Text))
            {

                Device.BeginInvokeOnMainThread(async () =>
                {
                    await this.DisplayAlert("Blad", "Wprowadz haslo
   .", "OK");
                })
                return;
            }

            if (string.IsNullOrEmpty(EntryUserName.Text))
            {
```

```
44
45              Device.BeginInvokeOnMainThread(async () =>
46              {
47                  await this.DisplayAlert("Blad", "Wprowadz nick
    .", "OK");
48              });
49              return;
50          }
51
52          var dbpath = Path.Combine(Environment.GetFolderPath(
    Environment.SpecialFolder.LocalApplicationData), "UserDatabase.
    db");
53          var db = new SQLiteConnection(dbpath);
54          db.CreateTable<LekomanItem>();
55
56          var item = new LekomanItem()
57          {
58              UserName = EntryUserName.Text,
59              Password = EntryUserPassword.Text,
60              Email = EntryUserEmail.Text
61          };
62          // Sprawdz, czy istnieje juz uzytkownik o podanym
    adresie email lub nazwie uzytkownika
63          var existingUser = db.Table<LekomanItem>().Where(u => u
    .Email == EntryUserEmail.Text || u.UserName == EntryUserName.
    Text).FirstOrDefault();
64          if (existingUser != null)
65          {
66              Device.BeginInvokeOnMainThread(async () =>
67              {
68                  await this.DisplayAlert("Blad", "Uzytkownik o
    podanym adresie email lub nazwie uzytkownika juz istnieje.", "OK
    ");
69              });
70              return;
71          }
72
73          db.Insert(item);
74          Device.BeginInvokeOnMainThread(async () => {
75
76              var resultPositive = await this.DisplayAlert("
    Gratulacje", "Rejestracja przebiegla pomyslnie", "OK", "Anuluj")
    ;
77
78              if (resultPositive)
```

```
79                          await Navigation.PushAsync(new LoginPage());
80                  });
81              }
82          }
83      }
```

Listing 5. Klasa *RegistrationPage.Xaml.cs*

Ten kod tworzy stronę rejestracji w aplikacji mobilnej, która pozwala użytkownikowi wprowadzić swoje dane i zarejestrować się w aplikacji. Kod sprawdza poprawność wprowadzonych danych i jeśli są one prawidłowe, zapisuje je w lokalnej bazie danych *SQLite*. Jeśli w bazie danych już istnieje użytkownik o podanym adresie email lub nazwie użytkownika, kod wyświetla komunikat o błędzie i zaprzestaje rejestracji. Po pomyślnej rejestracji użytkownik jest przekierowywany do strony logowania.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="LekomanApp.Views.RegistrationPage"
             BackgroundColor="#2981B2">
    <ContentPage.Content>
        <Frame
       HasShadow="True"

       HeightRequest="120"
       WidthRequest="120">
            <Frame.Background>
                <!-- StartPoint defaults to (0,0) -->
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="#30B6EF"
                          Offset="0.1" />
                    <GradientStop Color="#437EC3"
                          Offset="1.0" />
                </LinearGradientBrush>
            </Frame.Background>

            <StackLayout>
                <ScrollView>
                    <StackLayout>
                        <Grid Padding="40,60,40,0" VerticalOptions
   ="CenterAndExpand">
                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                                <RowDefinition Height="Auto"/>
                            </Grid.RowDefinitions>

                            <StackLayout Grid.Row="0">
                                <Image Source="logo.png" Margin
   ="0,0,30,30" WidthRequest="150" HeightRequest="120"></Image>
                            </StackLayout>

                            <StackLayout Grid.Row="1">
                                <Entry Placeholder="Email"
   FontSize="16" x:Name="EntryUserEmail" />
                            </StackLayout>

```

```
                            <StackLayout Grid.Row="2">
                                <Entry Placeholder="Nazwa" FontSize
    ="16" x:Name="EntryUserName"/>
                            </StackLayout>

                            <StackLayout Grid.Row="4">
                                <Entry Placeholder="Haslo" FontSize
    ="16" IsPassword="true" x:Name="EntryUserPassword"/>
                            </StackLayout>

                            <StackLayout Grid.Row="5">
                                <Button
                            VerticalOptions="Center"
                             Text="Rejestracja"
                              TextColor="white"
                             BackgroundColor="#0e387d"
                             BorderWidth="1.5"
                             CornerRadius="50"
                             Clicked="Button_Clicked"
                             Margin="0,50,0,0"
                            ></Button>
                            </StackLayout>
                        </Grid>
                    </StackLayout>
                </ScrollView>

            </StackLayout>

        </Frame>
    </ContentPage.Content>
</ContentPage>
```

Listing 6. Klasa *RegistrationPage.Xaml*

Ten kod wyświetla stronę rejestracji za pomocą Xamarin.Forms. W kodzie znajduje się element interfejsu użytkownika, taki jak *Frame*, *StackLayout*, *Grid*, *Entry*, *Button* itp., które są używane do wyświetlania elementów interfejsu użytkownika na stronie, takich jak logo, pola tekstowe do wprowadzenia danych użytkownika (np. adresu e-mail, nazwy użytkownika i hasła), oraz przycisk do zatwierdzenia rejestracji. Kod zawiera również właściwości i atrybuty, które określają wygląd i zachowanie elementów interfejsu użytkownika na stronie, takie jak kolor tła, wielkość czcionki, czy pole tekstowe jest ukrywane (np. hasło), a także metodę obsługi zdarzenia *Button_Clicked*, która jest wywoływana po kliknięciu przycisku rejestracji.

```
1
2  using SQLite;
3  using System;
4  using System.Collections.Generic;
5  using System.Text;
6  using System.Threading.Tasks;
7  using System.IO;
8
9  using Xamarin.Forms;
10 using Xamarin.Forms.Xaml;
11 using LekomanApp.Models;
12 using LekomanApp.Data;
13
14 namespace LekomanApp.Views
15 {
16     [XamlCompilation(XamlCompilationOptions.Compile)]
17     public partial class LoginPage : ContentPage
18     {
19         public LoginPage()
20         {
21             InitializeComponent();
22         }
23
24         private async void Button_Clicked(object sender, EventArgs
       e)
25         {
26             await Navigation.PushAsync(new RegistrationPage());
27         }
28
29         private void Button_Clicked_1(object sender, EventArgs e)
30         {
31             var dbpath = Path.Combine(Environment.GetFolderPath(
       Environment.SpecialFolder.LocalApplicationData), "UserDatabase.
       db");
32             var db = new SQLiteConnection(dbpath);
33             var myquery = db.Table<LekomanItem>().Where(u => u.
       UserName.Equals(EntryUser.Text) && u.Password.Equals(
       EntryPassword.Text)).FirstOrDefault();
34
35             if (myquery != null)
36             {
37                 App.Current.MainPage = new NavigationPage(new
       LekomanListPage());
38             }
39
```

```
40              else
41              {
42                  Device.BeginInvokeOnMainThread(async () => {
43
44                      var result = await this.DisplayAlert("Blad!", "
    Bledny login lub haslo!", "OK", "Anuluj");
45
46                      if (result)
47                          await Navigation.PushAsync(new LoginPage())
    ;
48                      else
49                      {
50                          await Navigation.PushAsync(new LoginPage())
    ;
51                      }
52                  });
53
54              }
55          }
56
57      }
58  }
```

Listing 7. Klasa *LoginPage.Xaml.cs*

Ten kod tworzy stronę logowania. W kodzie znajduje się element interfejsu użytkownika, taki jak Entry oraz przyciski *Button*, które są używane do wyświetlania pól tekstowych do wprowadzenia nazwy użytkownika i hasła oraz przycisków do zatwierdzenia lub anulowania logowania. Kod zawiera również metody obsługi zdarzeń, takie jak *Button_Clicked* i *Button_Clicked_1*, które są wywoływane po kliknięciu odpowiednich przycisków. Metoda *Button_Clicked* przechodzi do strony rejestracji, natomiast *Button_Clicked_1* sprawdza, czy wprowadzone dane logowania są poprawne i jeśli tak, to przechodzi do strony głównej aplikacji. W przeciwnym razie wyświetla komunikat o błędzie logowania. Kod używa również biblioteki *SQLite* do połączenia z bazą danych aplikacji i sprawdzenia, czy wprowadzone dane logowania znajdują się w bazie danych.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="LekomanApp.Views.LoginPage"
             BackgroundColor="#2981B2">
    <ContentPage.Content>

        <Frame
       HasShadow="True"

       HeightRequest="120"
       WidthRequest="120">
            <Frame.Background>
                <!-- StartPoint defaults to (0,0) -->
                <LinearGradientBrush EndPoint="0,1">
                    <GradientStop Color="#30B6EF"
                          Offset="0.1" />
                    <GradientStop Color="#437EC3"
                          Offset="1.0" />
                </LinearGradientBrush>
            </Frame.Background>
            <ScrollView>
                <StackLayout>
                    <Grid Padding="40,0,40,0" VerticalOptions="
   CenterAndExpand">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="Auto"/>
                            <RowDefinition Height="Auto"/>
                            <RowDefinition Height="Auto"/>
                            <RowDefinition Height="Auto" />
                            <RowDefinition Height="Auto"/>
                        </Grid.RowDefinitions>

                        <Image Margin="0,0,30,30" WidthRequest
   ="150" HeightRequest="120"  Source="logo.png" Grid.Row="0" ></
   Image>
                        <Entry Placeholder="Nazwa" x:Name="
   EntryUser" Grid.Row="1"/>
                        <Entry Placeholder="Haslo" x:Name="
   EntryPassword" IsPassword="true" Grid.Row="2"/>
                        <Button Text="Zaloguj sie." Grid.Row="3"
   Margin="0,20,0,0" BackgroundColor="#0e387d" TextColor="White"
   CornerRadius="50" Clicked="Button_Clicked_1"/>
                        <Button Text="Zarejestruj sie." Grid.Row
   ="4" Margin="0,20,0,0" CornerRadius="50" Clicked="Button_Clicked
```

```
    "/>
38                      </Grid>
39                  </StackLayout>
40              </ScrollView>
41          </Frame>
42      </ContentPage.Content>
43  </ContentPage>
```

Listing 8. Klasa *LoginPage.Xaml*

Ten kod wyświetla stronę logowania. W kodzie znajduje się element interfejsu użytkownika, taki jak *Frame*, *StackLayout*, *Grid*, *Image*, *Entry*, *Button* itp., które są używane do wyświetlania elementów interfejsu użytkownika na stronie, takich jak logo, pola tekstowe do wprowadzenia nazwy użytkownika i hasła, oraz przyciski do zatwierdzenia lub anulowania logowania.

Kod zawiera również właściwości i atrybuty, które określają wygląd i zachowanie elementów interfejsu użytkownika na stronie, takie jak kolor tła, wielkość czcionki, czy pole tekstowe jest ukrywane (np. hasło), a także metody obsługi zdarzeń *Button_Clicked* i *Button_Clicked_1*, które są wywoływane po kliknięciu odpowiednich przycisków. Metoda *Button_Clicked* przechodzi do strony rejestracji, natomiast *Button_Clicked_1* sprawdza, czy wprowadzone dane logowania są poprawne i jeśli tak, to przechodzi do strony głównej aplikacji. W przeciwnym razie wyświetla komunikat o błędzie logowania.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LekomanApp.Models;
using LekomanApp.Data;
using Xamarin.Forms.Xaml;
using Xamarin.Forms;
using Plugin.LocalNotifications;

namespace LekomanApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LekomanItemPage : ContentPage
    {
        public LekomanItemPage()
        {
            InitializeComponent();
        }

        async void OnSaveClicked(object sender, EventArgs e)
        {
            var lekomanItem = (LekomanItem)BindingContext;

            CrossLocalNotifications.Current.Show(
            "Przypomnienie!",
             "Nie zapomnij o zazyciu: " + lekomanItem.Lek + " w dawce: " + lekomanItem.Dawka + "  za 10 minut", // tekst powiadomienia
                lekomanItem.ID, // identyfikator powiadomienia
             lekomanItem.Data.Add(lekomanItem.Godzina.Add(TimeSpan.FromMinutes(-10)))  // data i godzina
            );
            LekomanItemDatabase database = await LekomanItemDatabase.Instance;
            await database.SaveItemAsync(lekomanItem);
            await Navigation.PopAsync();
        }

        async void OnDeleteClicked(object sender, EventArgs e)
        {
            var lekomanItem = (LekomanItem)BindingContext;
            LekomanItemDatabase database = await LekomanItemDatabase.Instance;
```

```
41              await database.DeleteItemAsync(lekomanItem);
42              await Navigation.PopAsync();
43          }
44
45          async void OnCancelClicked(object sender, EventArgs e)
46          {
47              await Navigation.PopAsync();
48          }
49
50          async void Exit_Clicked(object sender, EventArgs e)
51          {
52              await Navigation.PushAsync(new LoginPage());
53          }
54      }
55  }
```

Listing 9. Klasa *LekomanItemPage.Xaml.cs*

Ten kod jest odpowiedzialny za implementację logiki dla widoku *LekomanItemPage* w aplikacji. Widok ten służy do tworzenia i edycji elementów w bazie danych aplikacji. Umożliwia on dodawanie nowych pozycji do listy oraz modyfikację już istniejących. Jest to implementacja klasy która dziedziczy po klasie *ContentPage*. Klasa ta posiada kilka metod obsługujących różne akcje użytkownika w widoku: *OnSaveClicked* wywoływana jest gdy użytkownik kliknie przycisk zapisz – zapisuje on zmodyfikowany element, *OnDeleteClicked* gdy użytkownik kliknie przycisk usuń – usuwa wybrany element, a *OnCancelClicked* gdy użytkownik kliknie przycisk anuluj. Ponadto, metoda *OnSaveClicked* wywołuje również powiadomienie za pomocą biblioteki *Plugin.LocalNotifications* – dzięki temu użytkownik dostanie powiadomienie 10 minut przed planowaną datą i godziną zażycia leku.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:sys="clr-namespace:System;assembly=mscorlib"
             x:Class="LekomanApp.Views.LekomanItemPage"
             Title="{Binding Lek}"
             BackgroundColor="#9ac3e6">
    <ContentPage.Content>
        <StackLayout
            Margin="20"
            Spacing="10"
            VerticalOptions="StartAndExpand">

            <Label Text="Lek" />

            <Entry Text="{Binding Lek}" />

            <Label Text="Dawka" />
            <Entry Text="{Binding Dawka}" />

            <Label Text="Data zazycia:" />
            <DatePicker Date="{Binding Data}"
                        MinimumDate="2022/12/15"/>
            <Label Text="Godzina zazycia:" />
            <TimePicker Time="{Binding Godzina}" />

            <StackLayout Orientation="Horizontal">
                <Label
                    Margin="0,10"
                    HorizontalOptions="StartAndExpand"
                    Text="Zrobione" />
                <Switch HorizontalOptions="EndAndExpand" IsToggled
   ="{Binding Zrobione}" />
            </StackLayout>

            <Button Clicked="OnSaveClicked" CornerRadius="50"
   TextColor="white" BackgroundColor="#093861" Text="Zapisz" />

            <Button Clicked="OnDeleteClicked" CornerRadius="50"
   TextColor="white" BackgroundColor="#093861"  Text="Usun" />

            <Button Clicked="OnCancelClicked" CornerRadius="50"
   TextColor="white" BackgroundColor="#093861" Text="Wyjdz" />

            <Button  Clicked="Exit_Clicked" CornerRadius="50"
```

```
    TextColor="white" BackgroundColor="#cf4b42" Text="Wyloguj"/>
42
43          </StackLayout>
44      </ContentPage.Content>
45  </ContentPage>
```

Listing 10. Klasa *LekomanItemPage.Xaml*

Ten kod tworzy interfejs wprowadzania i edytowana leku w aplikacji. Aplikacja wyświetla na ekranie elementy interfejsu, takie jak etykiety tekstowe, pola tekstowe – do wprowadzania leku oraz dawki, przyciski i wybór daty i godziny. Elementy te są powiązane z danymi za pomocą zbindowania (binding). Przyciski wywołują określone procedury obsługi zdarzeń, takie jak Zapisz, Usuń i Wyjdź oraz wyloguj.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LekomanApp.Data;
using Xamarin.Forms;
using LekomanApp.Models;
using Xamarin.Forms.Xaml;

namespace LekomanApp.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class LekomanListPage : ContentPage
    {
        public LekomanListPage()
        {
            InitializeComponent();
        }

        protected override async void OnAppearing()
        {
            base.OnAppearing();
            LekomanItemDatabase database = await
   LekomanItemDatabase.Instance;
            listView.ItemsSource = await database.GetItemsAsync();
        }
        async void OnItemAdded(object sender, EventArgs e)
        {
            await Navigation.PushAsync(new LekomanItemPage
            {
                BindingContext = new LekomanItem()
            });
        }
        async void OnListItemSelected(object sender,
   SelectedItemChangedEventArgs e)
        {
            if (e.SelectedItem != null)
            {
                await Navigation.PushAsync(new LekomanItemPage
                {
                    BindingContext = e.SelectedItem as LekomanItem
                });
            }
        }
```

```
44          async void Exit_Clicked(object sender, EventArgs e)
45          {
46              await Navigation.PushAsync(new LoginPage());
47          }
48      }
49  }
```

Listing 11. Klasa *LekomanListPage.Xaml.cs*

Ten kod odpowiada za zarządzania listą leków. Konkretnie, kod ten definiuje klasę *LekomanListPage*, która jest stroną z listą lekarstw w aplikacji. Klasa ta dziedziczy po klasie *ContentPage* i zawiera kilka metod, takie jak *OnAppearing*, *OnItemAdded*, *OnListItemSelected* i *Exit_Clicked*, które są wywoływane w odpowiednich momentach życia strony.

Metoda *OnAppearing* jest wywoływana, gdy strona jest wyświetlana na ekranie jej głównym zadaniem jest pobranie listy lekarstw z bazy danych i ustawienie jej jako źródła dla *listView* - czyli widoku na liśćie. Metoda *OnItemAdded* jest wywoływana po kliknięciu przycisku dodawania nowego lekarstwa, a jej zadaniem jest przejście do nowej strony, na której można dodać nowe lekarstwo do bazy danych. Metoda *OnListItemSelected* jest wywoływana po wybraniu elementu z listy lekarstw i jej zadaniem jest przejście do strony, na której można edytować wybrane lekarstwo. Metoda *Exit_Clicked* jest wywoływana po kliknięciu przycisku wyjścia i jej zadaniem jest wyjście do strony logowania aplikacji.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="LekomanApp.Views.LekomanListPage"
             Title="Dodaj swoje leki!"
             BackgroundColor="#dae0eb">
    <ContentPage.ToolbarItems>
        <ToolbarItem Clicked="OnItemAdded" Text="+">

            <ToolbarItem.IconImageSource>

                <OnPlatform x:TypeArguments="ImageSource">
                    <On Platform="Android,UWP" Value="plus.png" />
                </OnPlatform>
            </ToolbarItem.IconImageSource>
        </ToolbarItem>
    </ContentPage.ToolbarItems>

    <ListView
        x:Name="listView"
        ItemSelected="OnListItemSelected"
        HasUnevenRows="True"
         SeparatorColor="black"
         SeparatorVisibility="Default">

        <ListView.ItemTemplate>
            <DataTemplate>
                <ViewCell>

                    <Frame>
                        <Frame.Background>
                            <LinearGradientBrush EndPoint="0.5,1"
   StartPoint="0.5,0">
                                <GradientStop Color="#dae0eb"
   Offset="0" />
                                <GradientStop Color="#a8adb5"
   Offset="0" />
                            </LinearGradientBrush>
                        </Frame.Background>

                        <StackLayout>
                            <StackLayout
                                Padding="0,10,0,0"
                                Margin="0,0,0,10"

```

```
43                                  HorizontalOptions="FillAndExpand"
44                                    Orientation="Horizontal"
45                                  VerticalOptions="CenterAndExpand">
46
47                                 <Label HorizontalOptions="Start"
    FontSize="20" TextColor="#3d83fc" Text="Lek: " />
48                                 <Label HorizontalOptions="
    StartAndExpand" FontSize="20" TextColor="#2e5585" Text="{Binding
     Lek}" />
49
50                                 <Label HorizontalOptions="Start"
    FontSize="20" TextColor="#3d83fc" Text="Dawka: " />
51                                 <Label HorizontalOptions="
    StartAndExpand" FontSize="20" TextColor="#292828" Text="{Binding
     Dawka}" />
52                             </StackLayout>
53
54                             <StackLayout
55                         HorizontalOptions="FillAndExpand"
56                         Orientation="Horizontal"
57                         VerticalOptions="StartAndExpand"
58                         Padding="0,0,0,10">
59                                 <Label HorizontalOptions="Start"
    FontSize="20"  TextColor="#3d83fc" Text="Data: " />
60                                 <Label HorizontalOptions="
    StartAndExpand" FontSize="20" TextColor="#292828" Text="{Binding
     Data, StringFormat='{0:dddd, dd MMMM }'}" />
61
62                                 <Label HorizontalOptions="Start"
    FontSize="20" TextColor="#3d83fc" Text="Godzina: " />
63                                 <Label HorizontalOptions="
    StartAndExpand" FontSize="20" TextColor="#292828" Text="{Binding
     Godzina}" />
64
65                                 <Image
66                             HorizontalOptions="End"
67
68                             IsVisible="{Binding Zrobione}"
69                             Source="check1.png"
70                             WidthRequest="20"
71                             />
72
73                             </StackLayout>
74                         </StackLayout>
75                     </Frame>
```

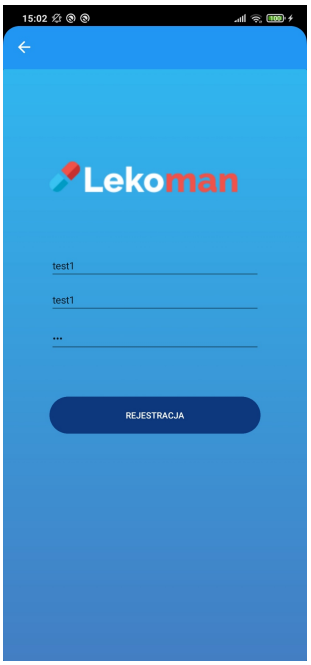
```
76                  </ViewCell>
77
78              </DataTemplate>
79
80          </ListView.ItemTemplate>
81      </ListView>
82
83  </ContentPage>
```

Listing 12. Klasa *LekomanListPage.Xaml*

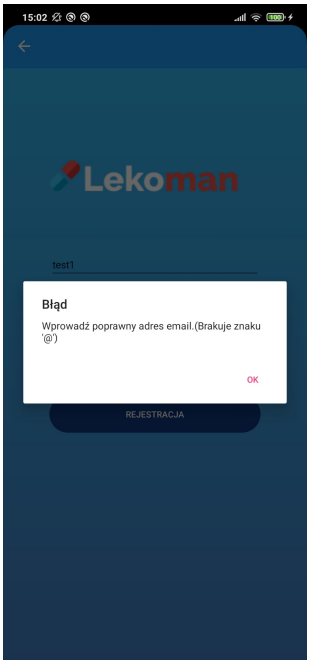
Ten kod definiuje wyświelanie listy leków zawierających nazwę leku, dawkę, datę i godzinę oraz wskazuje, czy lek został przyjęty. Strona zawiera także narzędzie z ikoną „+", które wywołuje metodę *OnItemAdded* po kliknięciu.

5. Testowanie

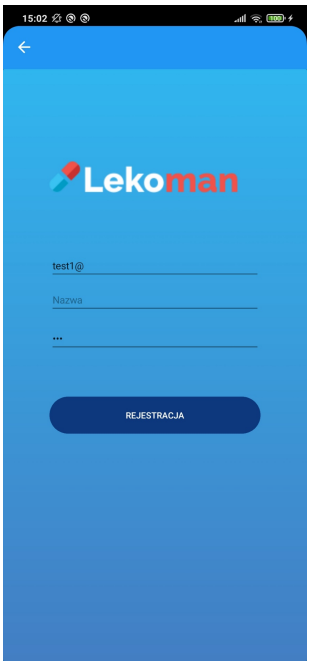
W tym rozdziale przedstawiamy działanie aplikacji Lekoman. Poniżej znajdują się zrzuty ekranu prób rejestracji oraz logowania użytkownika. Pole e-mail posiada walidację znaku @. W przypadku, gdy podana wartość nie zawiera tego znaku pojawia się komunikat błędu. Proces rejestracji wymaga wypełnienia wszystkich z trzech pól, czyli po kolei: e-mail, nazwa i hasło. Po zalogowaniu użytkownik wprowadza dane, które magazynują w bazie danych aplikacji. Przed ustaloną datą z 10-minutowym wyprzedzeniem pojawia się powiadomienie PUSH na telefonie. Jest ono udogodnieniem dla użytkownika i pełni funkcję czujnika w tej aplikacji.



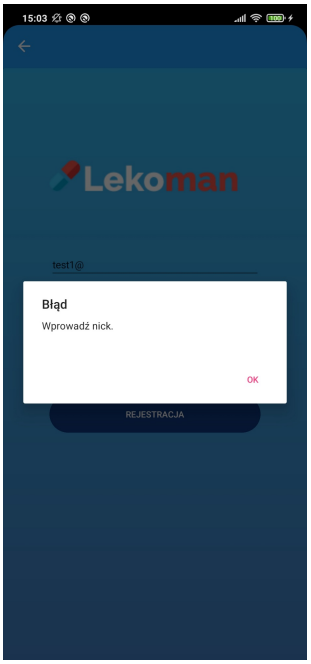
Rys. 5.1. Widok panelu rejestracji oraz pierwsza próba rejestracji użytkownika.



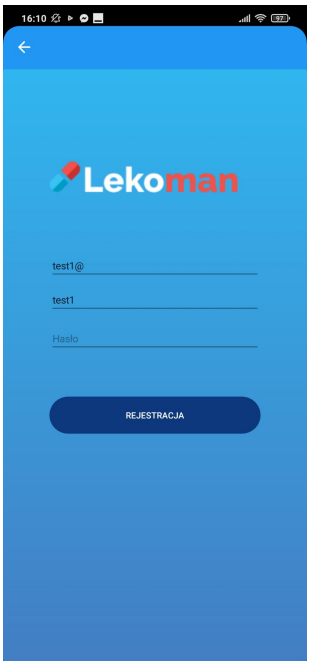
Rys. 5.2. Podanie złego formatu adresu e-mail powoduje pojawienie się komunikatu.



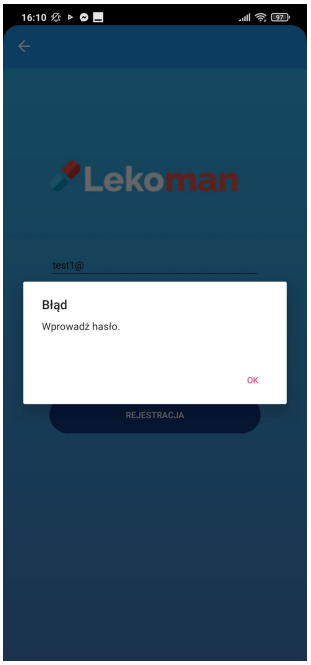
Rys. 5.3. Niepodanie nazwy użytkowika podczas rejestracji.



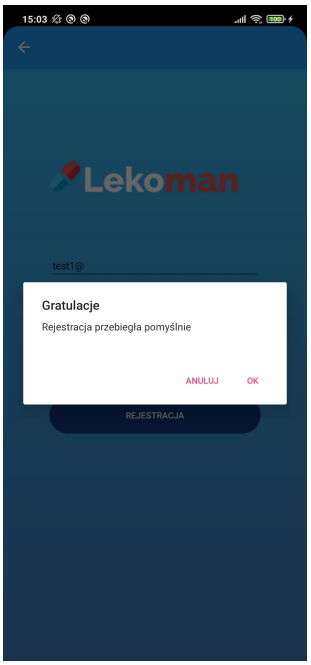
Rys. 5.4. Brak podania nicku powoduje pojawienie się powyższego komunikatu.



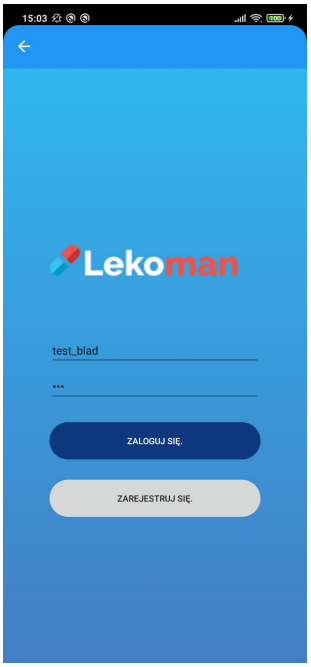
Rys. 5.5. Niepodanie hasła podczas rejestracji.



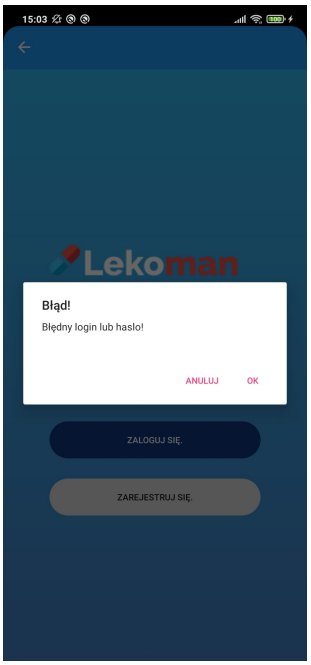
Rys. 5.6. Brak podania hasła powoduje pojawienie się powyższego komunikatu.



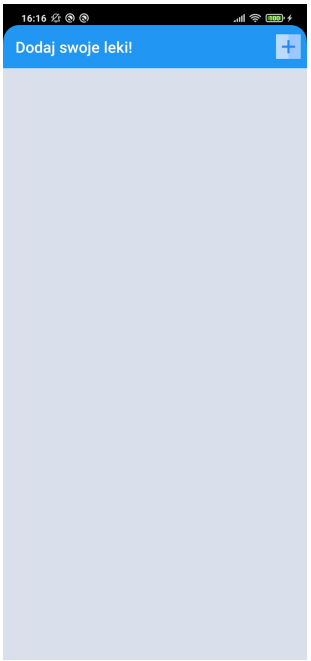
Rys. 5.7. Po podaniu prawidłowych danych w odpowiednim formacie, użytkownik zostaje zarejestrowany, na co wskazuje powyższy komunikat.



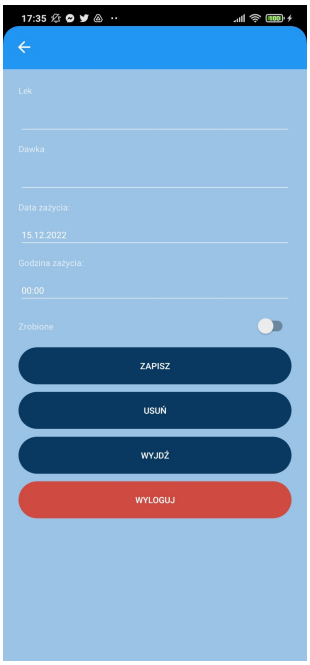
Rys. 5.8. Wprowadzenie błędnych danych do logowania.



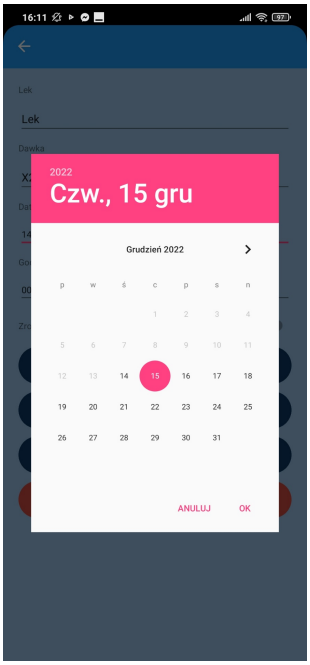
Rys. 5.9. Komunikat błędnego logowania użytkownika.



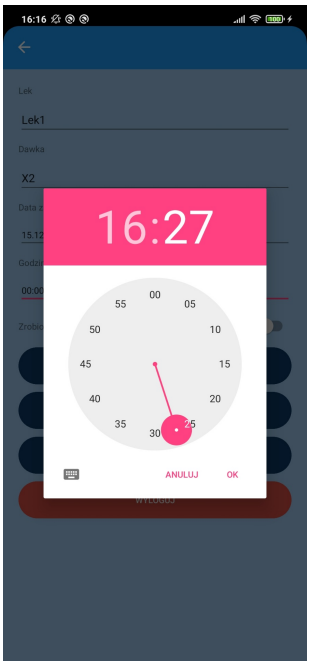
Rys. 5.10. Widok aplikacji po prawidłowym logowaniu użytkownika.



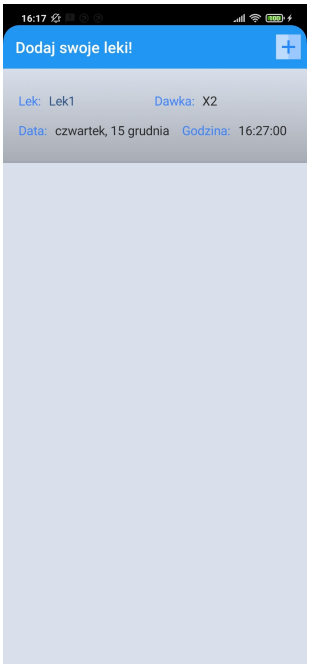
Rys. 5.11. Widok aplikacji podczas wprowadzania nazwy, data zażycia oraz dawki leku.



Rys. 5.12. Widok wyboru daty z zaimplementowanego kalendarza.



Rys. 5.13. Widok wyboru godziny z zaimplementowanego zegara.



Rys. 5.14. Widok w panelu aplikacji po wprowadzeniu dawki oraz daty zażycia leku.

Rys. 5.15. Widok powiadomienia push na telefonie przypominający o zażyciu leku na 10 minut przed wyznaczoną datą.

Rys. 5.16. Widok rozszerzonej bazy leków w panelu aplikacji.

6. Podręcznik użytkownika

Kliknij w ikonę aplikacji „Lekoman" znajdującej sie w panelu startowym na Twoim telefonie. Po uruchomieniu aplikacji ukaże Ci się widok do rejestracji. W tym kroku wprowadź e-mail, nazwę i hasło. Pamiętaj o prawidłowym formacie wpisywanych rekordów. Twój e-mail zawsze musi zawierać znak @. Gdy Twoje personalia będą odpowiednie dla parametrów aplikacji, rejestracja zakończy się sukcesem. W celu weryfikacji powinieneś przejść do procesu logowania, a więc należy uzupełnić pola: login oraz hasło. Uwierzytelnianie zakończyło się pomyślnie i w tym momencie nastąpi przekierowanie do głównego widoku wprowadzania dawek leków.

W prawym górnym rogu znajduję się przycisk do dodawania sporządzonych przez Ciebie partii medykamentów. Pojawią się zdefiniowane pola. Twoim zadaniem będzie podanie: nazwy leku i dawki, czyli częstotliwości z jaką musisz przyjmować dany lek. Następnie z zaimplementowanego kalendarza wyszukujesz datę, która będzie kluczowym aspektem przy zażywaniu konkretnego leku, by zatwierdzić datę kliknij przycisk „OK". Ponadto posiadasz możliwość ustawienia godziny z bardzo przejrzystego i funkcjonalnego zegara, również w tym przypadku, aby zatwierdzić godzinę wciśnij przycisk „OK". Kolejną zaletą aplikacji jest odznaczenie zrealizowanych dawek medykamentów przez przesunięcie pola "Zrobione". To bardzo ważny atut, bo dzięki niemu widzisz, którą pozycję już wprowadziłeś do systemu. Gdy okaże się,że wprowadzona przez Ciebie dawka jest błędna wykorzystaj przycisk „USUŃ" i wtedy całkowicie pozbędziesz się mylnie zaimportowanych danych.

Teraz, aby zapisać wszystkie dane naciśnij „ZAPISZ". Kliknij przycisk „WYJDŹ", a Twoim oczom w panelu aplikacji ukaże się przejrzysty spis wcześniej podanych informacji.

Nie martw się, że będziesz musiał za każdym razem sprawdzać i bezsensownie tracić swój cenny czas, aby pamiętać o godzinie przyjmowania leku. Nasza aplikacja zrobi to za Ciebie. Dzięki wbudowanym, niesamowicie dostosowanym do upodobań użytkownika powiadomieniom PUSH, które będą wyświetlać się na ekranie Twojego telefonu z 10-minutowym wyprzedzeniem, abyś nie był zobligowany do kontrolowania i ciągłego, żmudnego sprawdzania godziny. Na zawsze zapomnisz o zaprzątaniu sobie głowy sprawami przyziemnymi. Gdy chcesz usłyszeć ten niezwykły dźwięk jakże ważnego powiadomienia nie wyciszaj swojego urządzenia mobilnego.

Literatura

https://learn.microsoft.com/en-us/dotnet/csharp/
https://learn.microsoft.com/en-us/xamarin/
https://github.com/praeclarum/sqlite-net
https://github.com/thudugala/Plugin.LocalNotification

Spis rysunków

Spis listingów