Zespołowy Projekt Informatyczny
Dokumentacja aplikacji Meeting Planner

Skład zespołu: Karol Haczyński, Patryk Jagielski, Artem Kuznetsov, Jakub Szulc, Paweł Szypryt

Opiekun zespołu: Jakub Tomczewski

Bydgoszcz, 2023

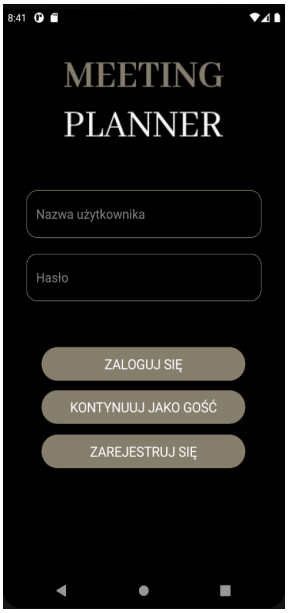
Spis treści

1. Instrukcja użytkownika

1.1. Gość

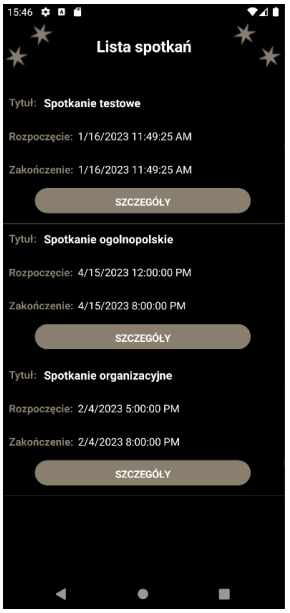
1.1.1. Ekran logowania

Na stronie głównej aplikacji znajduje się ekran logowania, na którym użytkownik może wybrać 3 opcje: “Zaloguj się", jeśli posiada już konto, “Zarejestruj się”, gdzie użytkownik może utworzyć nowe konto oraz opcja “Kontynuuj jako gość” aby nie podawać żadnych danych.



1.1.2. Kontynuuj jako gość

Po kliknięciu przycisku „Kontynuuj jako gosc” użytkownik nie musi wpisywać swojego loginu i hasła i przechodzi do formatki, na której widoczna jest lista spotkań ogólnopolskich.



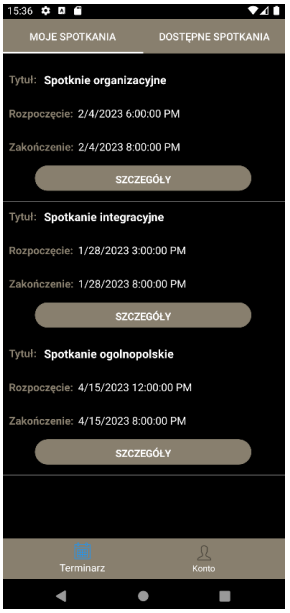
1.1.3. Rejestracja

Na widoku rejestracji, podając takie dane jak login, hasło, imię, nazwisko, e-mail, numer telefonu oraz wybierając swój region, użytkownik może utworzyć konto osobiste, do którego może się zalogować na ekranie logowania.

1.2. Użytkownik

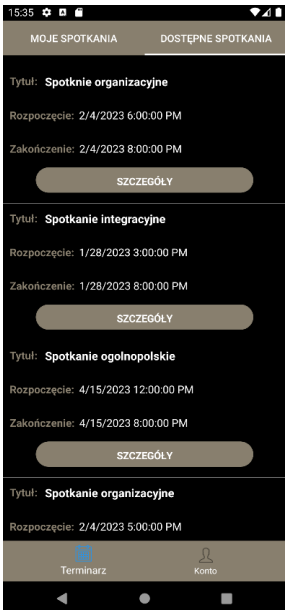
1.2.1. Widok ‘Moje spotkania’

Po wybraniu opcji “Moje spotkania” z górnego paska użytkownik zostaje przeniesiony do widoku wszystkich spotkań, na które został zapisany. Po kliknięciu przycisku „Szczegóły” użytkownik zobaczy szczegóły spotkania, na które się zapisał.



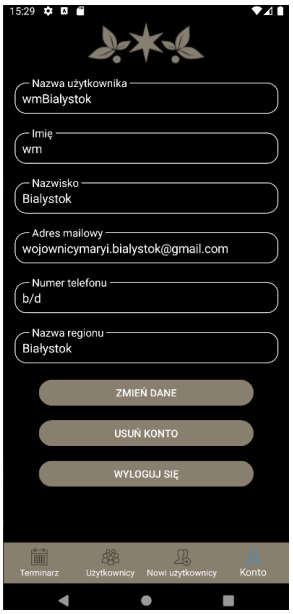
1.2.2. Widok ‘Dostępne spotkania’

Wybierając opcję „Moje spotkania” użytkownik zobaczy wszystkie spotkania powiązane z regionem podanym podczas rejestracji oraz spotkania ogólnopolskie.



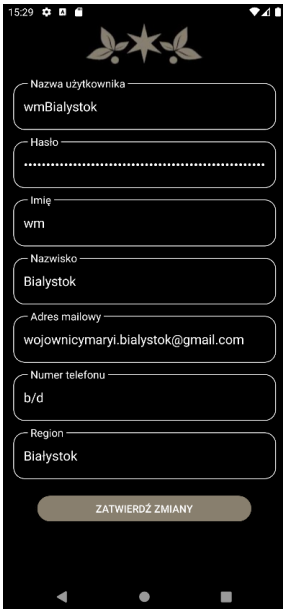
1.2.3. Szczegóły konta użytkownika

Po wybraniu opcji „Konto” z dolnego paska ekranu, użytkownik zostanie przekierowany do widoku „Szczegóły konta użytkownika”, gdzie może zobaczyć wszystkie dane swojego konta, a także zmienić dane wprowadzone podczas rejestracji, usunąć konto lub po prostu wylogować się z konta.



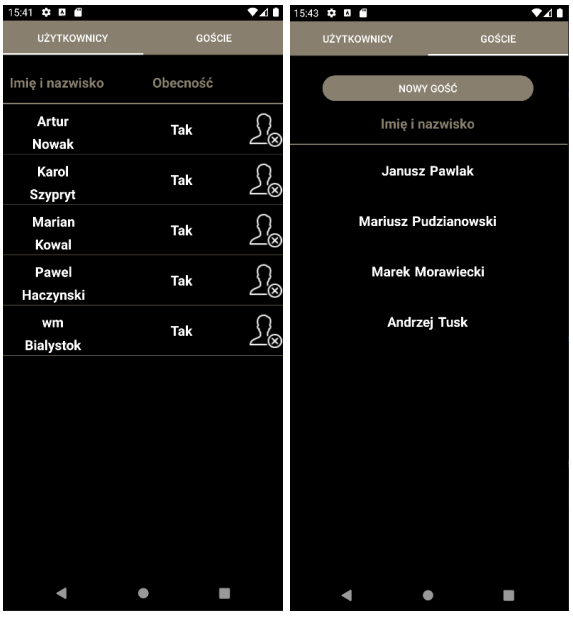
1.2.4. Edycja konta użytkownika

Po kliknięciu przycisku „Zmień dane” użytkownik zostanie przeniesiony do widoku, w którym może zmienić dane wprowadzone podczas rejestracji. Po naciśnięciu „Zatwierdź zmiany” zmiany wprowadzone przez użytkownika będą widoczne w zakładce „Konto”.



1.2.5. Widok ‘Lista obecności’

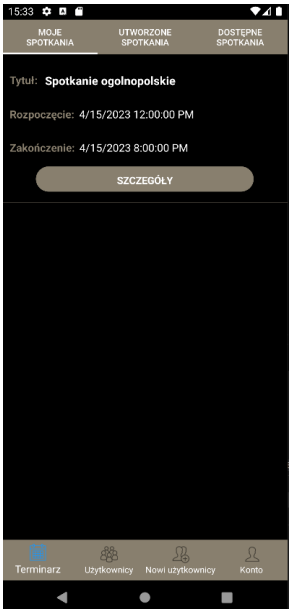
W tym widoku użytkownik widzi wszystkie osoby, które zapisały się na wybrane spotkanie, a także osoba sprawdzająca obecność na spotkaniu może zaznaczyć czy dana osoba na liście była obecna na spotkaniu czy nie. Można również dodać gościa, który nie posiada własnego konta, ale był obecny na spotkaniu.



1.3. Lider Grupy

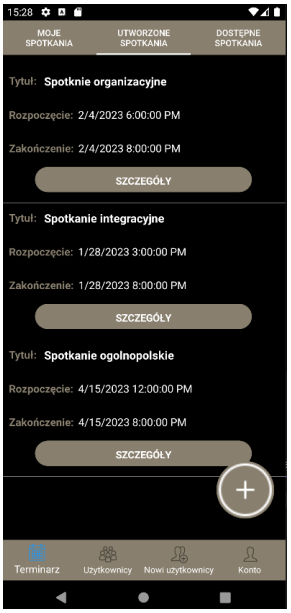
1.3.1. Widok ‘Moje spotkania’

Lider regionu po poprawnym zalogowaniu się do aplikacji zostaje przeniesiony do zakładki ,,Moje spotkania” - to spotkania, na które się zapisał, a nie jest ich twórcą. Po naciśnięciu przycisku ,,Szczegóły” zostaną wyświetlone szczegóły danego spotkania, możliwość rezygnacji z obecności oraz wygenerowania listy obecności dla spotkania.



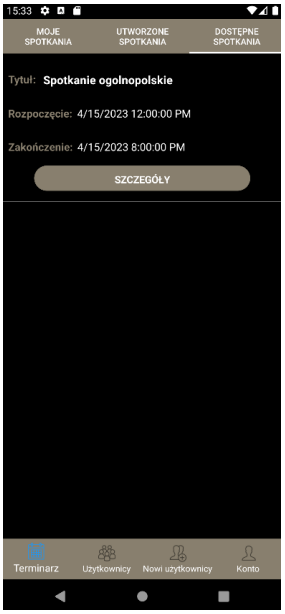
1.3.2. Widok ‘Utworzone spotkania’

Użytkownik po wybraniu zakładki ‘’Utworzone spotkania’’ zostaje przeniesiony do listy spotkań, które sam utworzył. Utworzone spotkanie jest równoznaczne z obecnością na tym spotkaniu. Po naciśnięciu ,,Szczegóły” użytkownik zostaje przeniesiony do szczegółów spotkania, gdzie ma możliwość edycji, usunięcia spotkania oraz wygenerowania listy obecności. Po naciśnięciu przycisku w dolnym prawym rogu ‘’+’’ użytkownik zostaje przeniesiony do formularza tworzenia nowego spotkania.



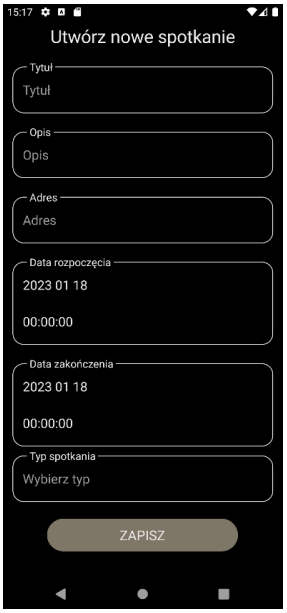
1.3.3. Widok ‘Dostępne spotkania’

Użytkownika po wybraniu z górnego paska zostaje przeniesiony do widoku spotkań, do których nie jest zapisany i których nie utworzył.



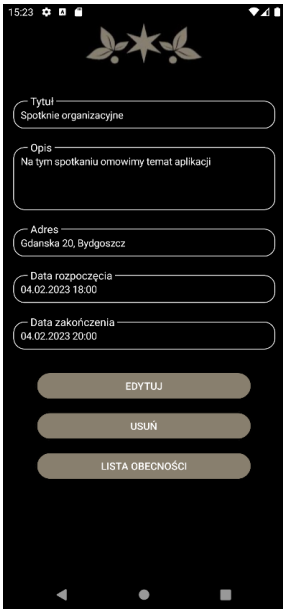
1.3.4. Widok ‘Dodania nowego spotkania’

Użytkownik po wybraniu przycisku z “+” zostaje przeniesiony do formatki, której wypełnienie i zapisanie skutkuje wygenerowaniem nowego spotkania.



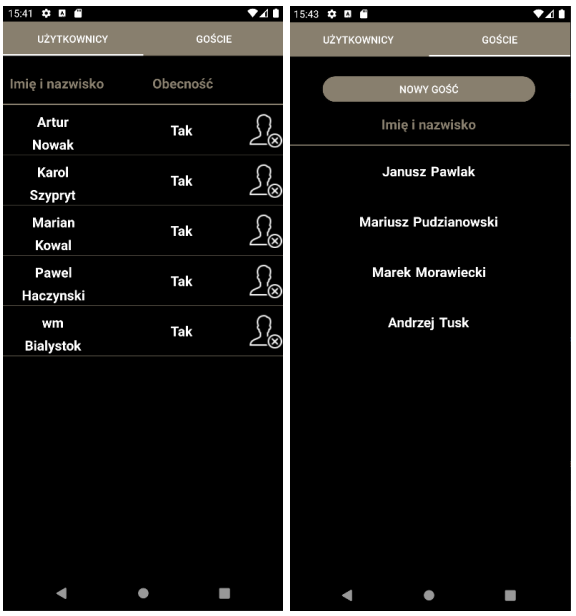
1.3.5. Widok ‘Edycji spotkania’

Użytkownik po wybraniu tej opcji zostaje przeniesiony do widoku, w którym widzi szczegóły spotkania. Z tego widoku ma możliwość edycji, usunięcia spotkania oraz przejście do listy obecności, która zostaje wygenerowana.



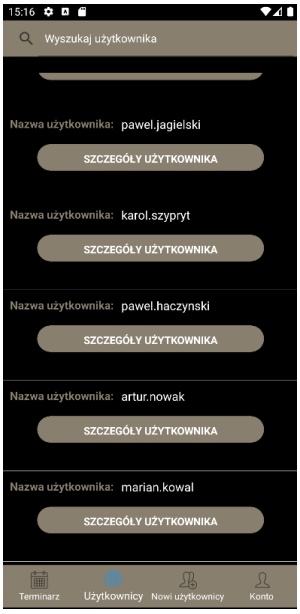
1.3.6. Widok ‘Listy obecności’

Użytkownikowi zostaje wyświetlona lista obecności osób, które zaznaczły obecność na tym spotkaniu. Taka możliwość jest do 30 minut po rozpoczęciu spotkania. Z górnego paska użytkownik może wybrać zarówno użytkowników, jak i gości. Gośca dodaje się naciskając na przycisk ,,Nowy gość”. Aby anulować obecność na danym spotkaniu wystarczy nacisnąć ikonę przy wierszu z imieniem i nazwiskiem danego użytkownika.



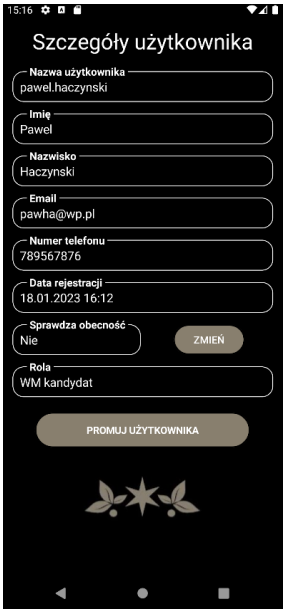
1.3.7. Widok ‘Lista użytkowników’

Użytkownik po wybraniu tej opcji zostaje przeniesiony do widoku użytkowników, którzy są przypisani do regionu, w którym tenże użytkownik jest liderem. Po naciśnięciu przycisku ,,Szczegóły użytkownika” zostaje przeniesiony do widoku szczegółów użytkownika. Jest również możliwość wyszukania użytkownika po naciśnięciu pola ,,Wyszukaj użytkownika” i wpisaniu jego nazwy użytkownika.



1.3.8. Widok ‘Szczegółów danego użytkownika’

Użytkownik wybierając tę opcję zostaje przeniesiony do widoku szczegółów użytkownika. Z tej formatki lider może promować użytkownika oraz nadać mu uprawnienie do sprawdzania listy obecności na spotkaniach.

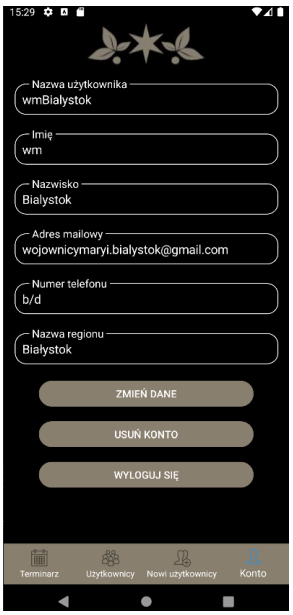


1.3.9. Widok ‘Lista nowych użytkowników’

Lider po wybraniu tej opcji zostaje przeniesiony do widoku kont użytkowników, którzy stworzyli konto w aplikacji, ale jeszcze nie zostali zaakceptowani. Użytkownik ma możliwość zaakceptowania albo usunięcia konta (jeśli ma na przykład podejrzenie co do wiarygodności danych)

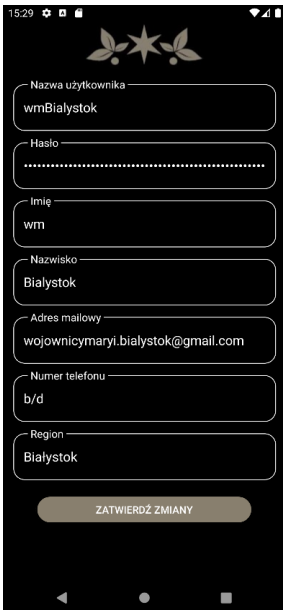
1.3.10. Szczegóły konta użytkownika

Po wybraniu tej opcji użytkownik zostaje przeniesiony do widoku szczegółów swojego konta. Ma moźliwość wylogowania się, usunięcia konta oraz zmiany własnych danych.



1.3.11. Edycja konta użytkownika

Po wybraniu tej opcji użytkownik zostaje przeniesiony do widoku edycji swoich danych. Aby zapisać wprowadzone zmiany należy nacisnąć przycisk ,,Zatwierdź zmiany”



2. Projekt aplikacji

2.1. Założenia funkcjonalne

Aplikacja umożliwia korzystanie z aplikacji osobom, które nie założyły konta w aplikacji (**gość**), w związku z czym ich możliwości są bardzo ograniczone i prezentują się następująco:

- możliwość dostępu tylko do listy spotkań ogólnopolskich

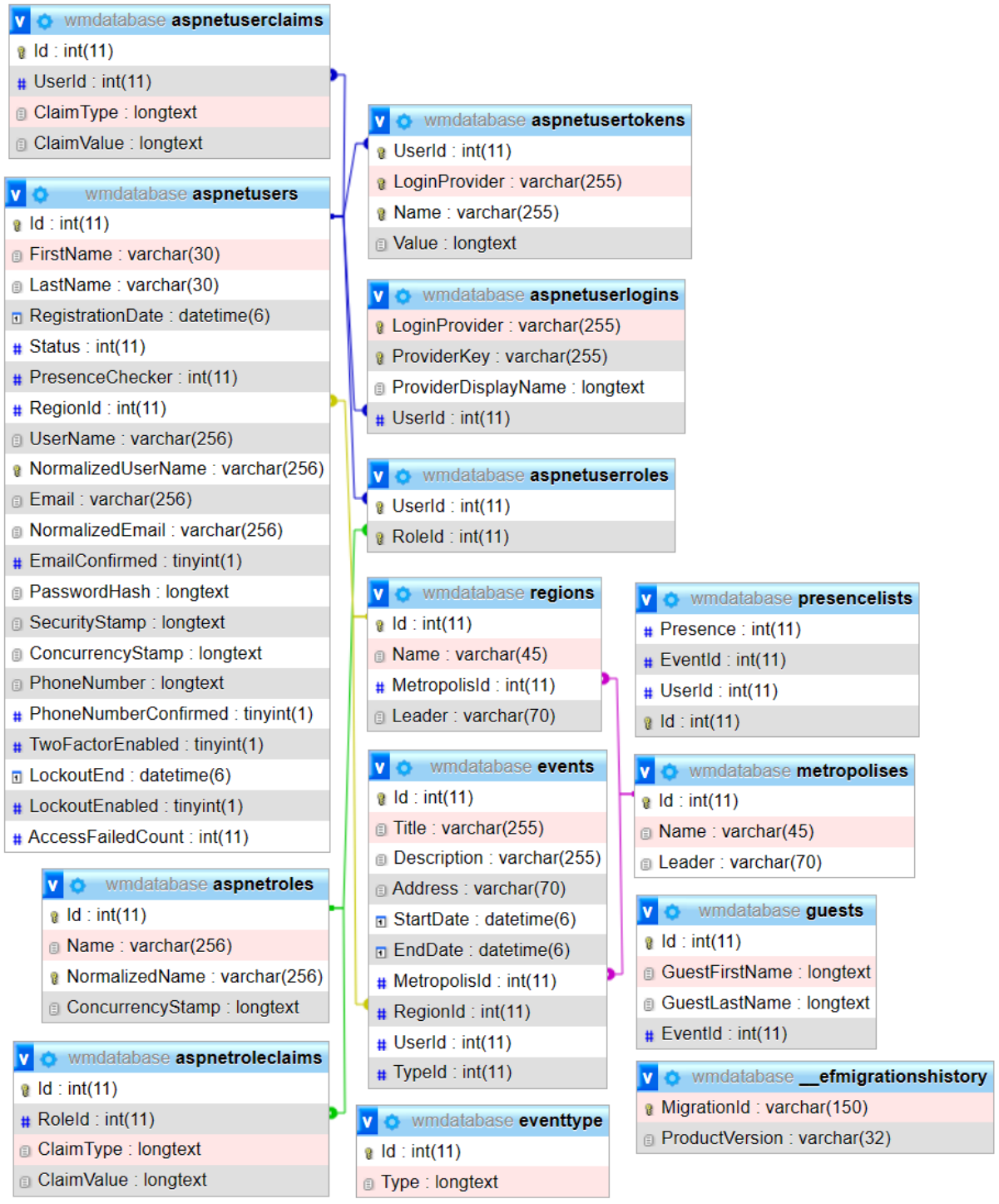
Kolejnym typem użytkownika jest **WM Kandydat/WM Pasowany**. Jest to użytkownik, którego konto zostało zatwierdzone przez lidera grupy: Jego możliwości są następujące:

- możliwość wglądu w listę dostępnych spotkań ogólnopolskich oraz regionalnych (dla regionu w którym się zarejestrował)
- możliwość wglądu w listę spotkań na które się zapisał
- możliwość wglądu w szczegóły swojego konta
- możliwość edycji danych swojego konta
- możliwość usunięcia swojego konta
- możliwość zapisu na spotkanie oraz ewentualnej rezygnacji z obecności
- możliwość wglądu w listę obecności wybranego spotkania

Następnym typem użytkownika jest **lider grupy.** Jest to użytkownik, który ma bardzo rozbudowany zestaw możliwości. Prezentują się one następująco:

- możliwość wglądu w listę dostępnych spotkań ogólnopolskich
- możliwość wglądu w listę spotkań na które się zapisał
- możliwość wglądu w listę spotkań które utworzył
- możliwość wglądu w szczegóły swojego konta
- możliwość edycji danych swojego konta
- możliwość usunięcia swojego konta
- możliwość zapisu na spotkanie oraz ewentualnej rezygnacji z obecności
- możliwość tworzenia nowych spotkań (w swoim regionie oraz ogólnopolskich)
- możliwość anulowania utworzonego spotkania
- możliwość sprawdzania obecności na spotkaniach
- możliwość akceptacji kont użytkowników, którzy przy rejestracji przypisali się do jego grupy
- możliwość wglądu w listę zarejestrowanych użytkowników
- możliwość zmiany roli użytkowników (awans lub zdegradowanie)

2.2. Struktura bazy danych



Rysunek 2.2.1 Struktura bazy danych

3. Implementacja aplikacji

3.1. Serwis internetowy

Poniższy przykład (Rys. 3.1.1) przedstawia fragment stworzonej klasy kontekstu, na podstawie której została wygenerowana baza danych z punktu 2.2.

```
public class WebAppContext : IdentityDbContext<User, Role, int>
{
    public DbSet<Metropolis> metropolises { get; set; }
    public DbSet<Region> regions { get; set; }
    public DbSet<Event> events { get; set; }
    public DbSet<PresenceList> presencelists { get; set; }
    public DbSet<EventType> eventType { get; set; }
    public DbSet<Guest> guests { get; set; }
```

Rysunek 3.1.1 Fragment klasy kontekstu

W celu używania Entity Framework Core do komunikowania się z bazą danych należało zdefiniować ‘Connection String’ o nazwie ‘CSWMDB’ (Rys 3.1.2), w którym zostały określone informacje o źródle danych oraz sposobie łączenia z nim, a następnie trzeba było dodać klasę DBContext jako usługę (Rys 3.1.3).

```
"ConnectionStrings": {
  "CSWMDB": "Server=127.0.0.1;Port=3306;Database=wmdatabase;User ID = root"
}
```

Rysunek 3.1.2 Connection String

```
services.AddDbContext<WebAppContext>(options =>
{
    var connetionString = Configuration.GetConnectionString("CSWMDB");
    options.UseMySql(connetionString, ServerVersion.AutoDetect(connetionString));
});
```

Rysunek 3.1.3 Dodanie klasy DBContext jako usługi

Fragment kodu znajdujący się na Rys 3.1.4, prezentuję metodę która odpowiada za pobranie listy obecności osób, które posiadają konta w systemie, z wybranego przez użytkownika wydarzenia, według przekazanego parametru ‘EventId’.

```
[Authorize(Roles = "WM kandydat, WM pasowany, Lider grupy")]
[Route("GetEventPresenceList")]
public async Task<ActionResult> GetEventPresenceList(int EventId)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }
    var presenceEventList = await (from user in appContext.Users
                                   join pl in appContext.presencelists on user.Id equals pl.UserId
                                   where pl.EventId == EventId
                                   select new
                                   {
                                       user.Id,
                                       user.FirstName,
                                       user.LastName,
                                       pl.Presence
                                   }).ToListAsync();

    return Ok(presenceEventList.OrderBy(x => x.FirstName));
}
```

Rysunek 3.1.4 Metoda odpowiedzialna za pobranie listy obecności z wybranego przez użytkownika wydarzenia

3.2. Aplikacja mobilna

Fragment kodu znajdujący się na Rys 3.2.1, prezentuję metodę która jest wywoływana, w celu wysłania żądania do serwisu internetowego, aby pobrać listę obecności osób, które posiadają konta w systemie, z wybranego przez użytkownika wydarzenia.

```
public async Task<IList<UserPresence>> GetEventPresenceList(int EventId)
{
    using (HttpClient httpClient = new HttpClient())
    {
        httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer",
          App.Token);
        var response = await httpClient.GetAsync("http://10.0.2.2:34290/api/Event/GetEventPresenceList?
          EventId=" + EventId);
        var content = await response.Content.ReadAsStringAsync();
        var userModel = JsonConvert.DeserializeObject<IList<UserPresence>>(content);

        return userModel;
    }
}
```

Rysunek 3.2.1 Funkcja odpowiedzialna za wysłanie żądania z parametrem w celu pobrania listy obecności z wybranego przez użytkownika wydarzenia

Fragment kodu znajdujący się na Rys 3.2.2, prezentuję przykładową metodę, która wywołuje funkcję z Rys 3.2.1, a następnie przypisuję zwróconą listę obecności do odpowiednio zdefiniowanej listy, która jest następnie wiązana z obiektem ListView, w celu ukazania żądanej listy użytkownikowi.

```
public ICommand RefreshData
{
    get
    {
        return new Command(async () =>
        {
            EventServices eventServices = new EventServices();
            openUserList = (List<UserPresence>)await eventServices.GetEventPresenceList(eventId);
            guestList = (List<Guest>)await eventServices.GetEventGuestsAsync(eventId);

            if (openUserList != null)
            {
                foreach (UserPresence item in openUserList)
                {
                    if (item.Presence == 1)
                    {
                        item.SourceName = "presenceCancel.png";
                        item.PresenceName = "Tak";
                    }
                    else
                    {
                        item.SourceName = "presenceOk.png";
                        item.PresenceName = "Nie";
                    }
                }
                usersPresenceListLV.IsVisible = true;
                usersPresenceListLV.ItemsSource = openUserList;
            }
            if (guestList != null)
            {
                guestsPresenceListLV.IsVisible = true;
                guestsPresenceListLV.ItemsSource = guestList;
            }
        });
    }
}
```

Rysunek 3.2.2 Przykład metody, która wywołuje funkcję ‘GetEventPresenceList’