2. PROJEKTINĖ DALIS

2.1.Mobilios aplikacijos struktūra

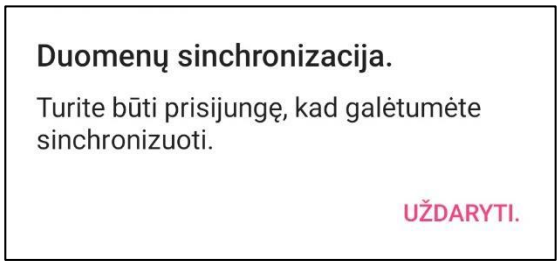
Pagrindiniai failai

Startavus aplikacijai, visuomet yra užtikrinama, kad bus sukurtas SQLite duomenų failas, pavadinimu „Moms_Database.db3“, kuriame vartotojas galės saugoti visus duomenis, susijusius su aplikacijoje galimais veiksmais.

Aplikacijos veikimas/ Vartotojo sąsaja

Paleidus aplikaciją, mobiliojo telefono ekrane atsidaro balto fono su grafiniu dramblių mamos ir vaiko sąryšio vaizdu langas, kuriame ant keturių balionėlių išsidėstę pagrindiniai mygtukai. Aplikacijos stilistika (žr. 7 priedą). siekiama išlaikyti švarų, minimalistinį dizainą, neįpareigojantį vartotoją mėtytis tarp daug funkcijų viename lange, sudėtingos meniu struktūros ar begalės spalvų ar šriftų, varginančių vartotojo sąmoningumą, naudojantis programa. Pradinė vartotojo veiksmų seka kaip naudotis aplikacija, pateikta 6 priede. Viršuje – melsvos spalvos juosta, kurios kairėje pusėje matomas šoninio meniu mygtukas. Paspaudus ant jo iš kairės pusės „išvažiuoja“ papildomo meniu langas su keturiais galimais pasirinkimais:

- Apie. Pateikiama bendra informacija apie mobiliąją aplikaciją.
- Sinchronizuoti (žr. 5 priedą). Norėdamas neprarasti duomenų, vartotojas turi galimybę juos sinchronizuoti su paslaugos tiekėju, kuris suteikia galimybę saugoti duomenis, šiuo atveju – FTPS serveriu. Tam kad pasinaudoti šia funkcija, vartotojas turi prisijungti, kitu atveju ekrane išmetamas įspėjamasis pranešimo langas (žr. pav. 18).



pav. 18 Pranešimo langas

- Prisijungti (žr. 4 priedą). Jeigu vartotojas yra prisiregistravęs, tereikia suvesti elektroninį paštą ir slaptažodį bei paspausti mygtuką „Prisijungti“. Kitu atveju spaudžiamas mygtukas „Registruotis“. Tuomet ekrane atidaromas naujas langas su galimybe įvesti elektroninį paštą, slaptažodį ir pakartotinai tą patį slaptažodį. Užsiregistravus spaudžiamas mygtukas „Grįžti“, prisijungiama prie aplikacijos ir suteikiama galimybė sinchronizuoti duomenis.
- Išjungti. Paspaudus šį mygtuką, aplikacija iškart išjungiama.

Kad vartotojas lengviau galėtų naudotis aplikacijos teikiamomis galimybėmis, pagrindiniame lange visos aplikacijos funkcijos suskirstytos į keturias grupes, kurios atitinkamai veikia kaip pasirinkimo mygtukai:

- Kontaktai;
- Kasdienė rutina;
- Duomenys;
- Žymekliai.

Kiekvienoje iš jų sudėtos pagrindinės reikalingos funkcijos, atvaizduotos grafiniais paveikslėliais, nurodančiais paskirtį.

Paspaudus mygtuką „Kontaktai“, atidaromas naujas langas, kuriame pasirinkus atitinkamą grafą ir paspaudus viršutinėje juostoje „Įvesti“, galima įsirašyti bei matyti:

- Bendrinius kontaktus;
- Gydytojų kontaktus;
- Įstaigų kontaktus.

Spustelėjus viršutinės juostos kairėje pusėje esančią rodyklę, grįžtama į pagrindinį aplikacijos langą. Pasirinkus antrajame balionėlyje esantį „Kasdienė rutina“ mygtuką, vartotojas nukreipiamas į naują langą, kuriame pasirinktinai galimos šešios funkcijos:

- Maistas. Galimi trys įvedimo pasirinkimai:
    - Maitinimai krūtine. Pasirenkama krūtinės pusė, maitinimo data bei laikas.
    - Mišinukai. Įvedamas mišinuko pavadinimas, maitinimo data ir laikas bei kiekis.
    - Kitas maistas. Įvedamas pavadinimas, maitinimo data, laikas ir kiekis.
- Gėrimai. Įvedamas pavadinimas, maitinimo data, laikas ir kiekis.
- Maudymas. Įvedamas trumpas aprašymas, data bei laikas.
- Sauskelnės. Įvedama sauskelnių/ vystyklų būsena, data bei laikas.
- Savijauta. Įvedama vaiko savijauta, data bei laikas.
- Miegas. Įvedamas trumpas aprašymas, pradžios bei pabaigos datos ir laikas.

Grįžus į pagrindinį langą ir pasirinkus trečiąjį mygtuką „Duomenys“, galimos aštuonios funkcijos pasirinktinai:

- Svoris. Įvedama data, laikas ir svoris kilogramais.
- Ūgis. Įvedama data, laikas ir ūgis centimetrais.
- Dantys. Pasirenkamas žandikaulis, žandikaulio pusė, dantuko numeris, įvedama data ir laikas.
- Judesiai. Įvedamas trumpas judesio aprašymas, data ir laikas.

- Temperatūra. Įvedama data, laikas ir temperatūros dydis laipsniais.
- Ligos. Įvedamas ligos pavadinimas, pastaba, data ir laikas.
- Informacija. Pateikiamos bendrinės nuorodos ir informacija, susijusi su vaikų priežiūra ir auginimu.
- Užrašai. Šia funkcija vartotojas gali naudotis kaip užrašų knygele reikalingai pagalbinei informacijai įrašyti. Įvedamas aprašymas, data ir laikas.

Pasirinkus pagrindiniame lange ketvirtąjį mygtuką „Žymekliai“, vartotojas nukreipiamas į kalendorių, kuriame suteikiama galimybė žymėtis visus vizitus, įsimintinas datas, vaistų vartojimo laikus ir visa kita, kas vartotojui aktualu.

Atidžiai apgalvojus aplikacijos struktūrą ir parengus jos modelį, pirmiausia VS aplinkoje sukonstruoti pagrindiniai puslapiai. Detaliai programiniu kodu aprašius pagrindines puslapių funkcijas, toliau sumodeliuoti pagalbiniai puslapiai papildomoms funkcijoms realizuoti.

Aplikacijos struktūra

- **Models katalogas**
  - DataBase. Šiame kataloge sudėtos visos klasės, skirtos atvaizduoti duomenų bazių struktūrą ir su ja komunikuoti:
    - Bathing.cs, Contacts.cs, Dipers.cs, Drinks.cs, Drugs.cs, Feeding.cs, Flus.cs, Health.cs, Heights.cs, Info.cs, Itable.cs, Markers.cs, Moves.cs, Notes.cs, Sleep.cs
    - , Temperature.cs, Tooths.cs, Weights.cs;
  - Enums (žr. 8 priedą). Šiame kataloge sudėtos kontrolės (pasirinkimo galimybės duomenų įvedimo metu), padedančios suformuoti objektą, pagal kurį galima atskirti, kokio tipo įrašas daromas ir išsaugomas duomenų bazėje:
    - ContactType.cs, FeedByChestSide.cs, FoodType.cs, Jaw.cs, JawSide.cs, ToothNumber.cs;
- **Services katalogas** (žr. 10 priedą)
  - ExitApp katalogas. Šiame kataloge yra klasė CloseApplication, kurioje aprašoma funkcija ExitApp(), skirta aplikacijos uždarymui bei komunikacijos sąsaja (angl. k. interface) ICloseApplication, per kurią pasiekiamos objekto funkcijos.
    - CloseApplication.cs, funkcijos kodas:

```
public void ExitApp()
{
    System.Diagnostics.Process.GetCurrentProcess().Kill();
```

}

- ICloseApplication.cs

- Ftp katalogas
  - FtpSyncService.cs – klasė, kurioje aprašomos žemiau išvardintos funkcijos, pasiekiamos per komunikacijos sąsajas IFtpSyncService ir IDisposable:
    - Task ConnectToFTPSAsync() – funkcijai naudojamas „Fluent FTP Nuget Package“, su kuriuo galima jungtis į FTPS serverį;
    - Dispose() – skirta kontroliuoti objektų gyvavimo laiką sistemoje (objektų išvalymui), kai paspaudžiamas mygtukas išeiti;
    - Dispose(bool disposing) – virtualus metodas, kurį iškviečia aukščiau minėta funkcija Dispose();
    - Task UploadFileAsync(string uuid) – sinchronizuoja vartotojų duomenų bazės failą i FTPS serverį pagal unikalų identifikacinį numerį;
    - Task DownloadFileAsync(string uuid) – parsiunčia iš FTPS serverio failą pagal vartotojo unikalų identifikacinį numerį;
    - Task<bool> FileExistsAsync (string uuid) – patikrina, ar failas FTPS serveryje egzistuoja pagal unikalų identifikacinį numerį;
  - IFtpSyncService.cs – komunikacijos sąsaja, per kurią pasiekiamos aukščiau išvardintos funkcijos;
- Repositories katalogas
  - IMySqlDatabase.cs – komunikacijos sąsaja, per kurią klasė MySqlDatabase pasiekia funkcijas;
  - ISqlLiteDatabase.cs – komunikacijos sąsaja, per kurią klasė SqlLiteDatabase pasiekia funkcijas;
  - MySqlDatabase.cs – klasė, kurioje aprašomos žemiau išvardintos funkcijos:
    - GetMySqlConnection() – skirta prisijungimui prie MariaDB duomenų bazės;

- Dispose() – skirta kontroliuoti objektų gyvavimo laiką sistemoje (objektų išvalymui), kai paspaudžiamas mygtukas išeiti;
- Dispose(bool disposing) – virtualus metodas, kurį iškviečia aukščiau minėta funkcija Dispose();
- CreateLogin(string email, string password) – skirta sukurti naują vartotoją ;
- CheckEmailExistsInDatabase(string email) – registruojant naują vartotoją skirta patikrinti, ar įvestas elektroninis paštas egzistuoja duomenų bazėje, kreipiamasi į funkciją CheckEmailExists(string email);
- CheckLoginSuccess(string email, string password) – čia iškviečiama GetSha256Hash(string value) funkcija, patikrinamas įvestas slaptažodis su esamu sistemoje;
- CheckLoginExists(string email) – prisijungimo metu patikrinama, ar toks vartotojas egzistuoja sistemoje, kreipiamasi į funkciją CheckEmailExists(string email);
- IList<Info> GetDataFromInfoTable() – skirta gauti duomenims iš duomenų bazės lentelių;
- MySqlDataReader CheckEmailExists(string email) – patikrinama, ar toks elektroninis paštas egzistuoja sistemoje;
- GetSalt() – skirta grąžinti papildomą pagalbinį raktą slaptažodžio šifravimui;
- GetSha256Hash(string value) – aprašomas slaptažodžio šifravimo algoritmas;

o SqlLiteDatabase.cs – klasė, atsakinga už komunikavimą su duomenų baze, kurioje aprašomos žemiau išvardintos funkcijos:

- CreateAllTables() – skirta duomenų lentelių sukūrimui;
- Task Insert(object @object) – bendrinė SQLite funkcija, skirta įrašo kūrimui, atpažįstant kokio tipo duomenis norima įrašyti;
- Task Update(object @object) – bendrinė SQLite funkcija, skirta įrašo atnaujinimui, atpažįstant kokio tipo duomenis norima atnaujinti;

- Task Delete(object @object) – bendrinė SqLite funkcija, skirta įrašo ištrynimui, atpažįstant kokio tipo duomenis norima ištrinti;
- Pasikartojančios funkcijos, skirtos gauti ir trinti duomenims, priklausomai nuo vartotojo pasirinkto veiksmo tipo (pvz. maudymas, judesiai ir kita). Kodas:

```
#region BabyHeights
    public async Task<List<Heights>> ReadAllBabyHeights()
    {
        return await Connection
            .Table<Heights>()
            .OrderByDescending(x => x.ActualTime)
            .ToListAsync();
    }
#endregion

#region BabyWeights
public async Task<List<Weights>> ReadAllBabyWeights()
{
    return await Connection
        .Table<Weights>()
        .OrderByDescending(x => x.ActualTime)
        .ToListAsync();
}
#endregion
```

- ISynchronizationService.cs
- SynchronizationService.cs – klasė, skirta sinchronizavimo veiksmas atlikti, kurioje aprašomos šios funkcijos:
    - Task Synchronize() – funkcija, kurioje aprašymas duomenų sinchronizavimo metodas;
    - Task UpdateInfoTable() – skirta atnaujinti duomenų bazės lentelės įrašus;

- CompareData<T> : IEqualityComparer<T> where T : Itable – Generic tipo objektas, kuris leidžia lyginti du objektus pagal tam tikras reikšmes. Tikrinama pagal unikalų raktą;

- **Utils katalogas** (žr. 9 priedą). Jame sudėti pagalbiniai metodai, kurie padeda atlikti tam tikras operacijas.
  - DateTimeUtils.cs – klasė, skirta datos formavimui, kurioje aprašomos funkcijos:
    - DateTime FormatDateTime(this DateTime date, TimeSpan time) – statinis metodas, skirtas konvertuoti datos formatui. Kodas:

```
public static DateTime FormatDateTime(this DateTime date, TimeSpan time)
{
    return new DateTime(date.Year, date.Month, date.Day, time.Hours, time.Minutes, time.Seconds);
}
```

    - FormatMySqlDateTime(this DateTime dateTime) – statinis metodas, skirtas konvertuoti datą į MariaDB duomenų bazės duomenų tipą. Kodas:

```
public static string FormatMySqlDateTime(this DateTime dateTime)
{
    return dateTime.ToString("yyyy-MM-dd hh:mm:ss");
}
```

  - GlobalProperties.cs – per visa aplikacijos gyvavimo laiką kol ji paleista, šioje klasėje aprašyti gyvuojantys kintamieji, kurių pagalba galima kontroliuoti aplikacijos būseną. Taip pat aprašyta funkcija CheckEmailWithRegex(), kuri leidžia tikrinti ar elektroninis paštas atitinka reikalavimus;
  - MapUtils.cs – pagalbinė klasė objektams transformuoti, naudojantis funkcijomis MapContacts(this Contacts contacts, Contacts contactsFromForm) ir MapFeedings(this Feeding feeding, Feeding feedingFromForm), prieš išsaugojant į duomenų bazę. Kodas:

```
public static void MapContacts(this Contacts contacts, Contacts contactsFromForm)
{
    contacts.Key = contactsFromForm.Key;
    contacts.Address = contactsFromForm.Address;
```

```
    contacts.FlatNumber = contactsFromForm.FlatNumber;
    contacts.HouseNumber = contactsFromForm.HouseNumber;
    contacts.Mailbox = contactsFromForm.Mailbox;
    contacts.ContactName = contactsFromForm.ContactName;
    contacts.Telephone = contactsFromForm.Telephone;
}

public static void MapFeedings(this Feeding, Feeding feedingFromForm)
{
    feeding.Key = feedingFromForm.Key;
    feeding.ActualTime = feedingFromForm.ActualTime;
    feeding.FoodDescription = feedingFromForm.FoodDescription;
    feeding.FoodType = feedingFromForm.FoodType;
    feeding.Quantity = feedingFromForm.Quantity;
}
```

- **ViewModels katalogas** (žr. 12 priedą). Šiame kataloge sudėtos klasės, kur kiekvienoje iš jų (išskyrus MarkersViewModel.cs) aprašomos trys funkcijos Initialize() ir Task InitializeContactTitlesList() (inicializuojama duomenų kolekcija (masyvas), kuri atvaizduoja duomenis (ką vartotojas įvedė) iš duomenų bazės) ir InitializeSubscribes() (funkcija, atsakinga už priregistruotų objektų apdorojimą ir įrašymą į duomenų bazę), priklausomai nuo klasės veiksmų tipo (pvz. maudymas, sauskelnės, svoris) ir pateikiamų duomenų:
    - BabyHeightViewModel.cs, kodas:

```
async void Initialize()
{
    BabyHeightCollection.Clear();
    var babyHeightData = await Database.ReadAllBabyHeights();

    foreach (var babyHeight in babyHeightData)
    {
        BabyHeightCollection.Add(babyHeight);
    }
}
```

```
void InitializeSubscribes()
{
    MessagingCenter.Subscribe<HeightViewPage,Heights>(this,      "AddOrUpdateBabyHeight",
async (sender, height) =>
    {
        if (height.Key == null)
        {
            height.Key = Guid.NewGuid().ToString();
            await Database.Insert(height);
        }
        else
        {
            await Database.Update(height);
        }
    });

    MessagingCenter.Subscribe<HeightViewPage,  Heights>(this,  "DeleteBabyHeight",  async
(sender, height) =>
    {
        await Database.Delete(height);
    });
}
```

- BabyWeightViewModel.cs
- BaseViewModel.cs – bazinė klasė, automatiškai sukuriama projekto pradžioje
- BathingViewModel.cs
- ContactsPageViewModel.cs, kodas:

```
async Task InitializeContactTitlesList()
{
    Contacts.Clear();
    var contacts = await Database.ReadAllContacts();
    contacts = contacts.Where(x => x.ContactType == Contact.ContactType).ToList();

    foreach (var contact in contacts)
```

```
    {
        Contacts.Add(contact);
    }
}
```

- DiperingViewModel.cs
- DrinksViewModel.cs
- DrugsPageViewModel.cs
- FeedingPageViewModel.cs – šioje klasėje papildomai aprašyta LoadFoodTypes() funkcija, kuri skirta sąrašo suformavimui, kad vartotojas galėtų pasirinkti iš kurios krūtinės pusės maitina
- FlusViewModel.cs
- HealthViewModel.cs
- InfoViewModel.cs
- MarkersViewModel.cs, kodas:

```
private async void AddCalendarEvents()
{
    Events.Clear();
    var markersList = await Database.ReadAllMarkers();
    var markersListTimeEvents = markersList.Select(x =>  x.ActualTime.Date).Distinct();
    foreach (var time in markersListTimeEvents)
    {
        if(Events.ContainsKey(time))
        {
            var existingCollection = Events.Where(x => x.Key == time.Date).Select(x => x.Value as
List<EventData>).First();
            Events.Remove(time);
            var arrayOfData = markersList.Where(x => x.ActualTime.Date == time.Date).ToList();
            var eventListData = new List<EventData>();

            foreach (var data in arrayOfData)
            {
                eventListData.Add(new EventData
                {
                    Name = $"Įvykis - {data.ActualTime.ToString("HH:mm")}",
```

```
                Description = data.Text,
                DateTime = data.ActualTime
            });
        }

        Events.Add(time, existingCollection.Union(eventListData).ToList());
    }
    else
    {
        var eventListData = new List<EventData>();
        var arrayOfData = markersList.Where(x => x.ActualTime.Date == time.Date).ToList();

        foreach(var data in arrayOfData)
        {
            eventListData.Add(new EventData {
                Name = $"Įvykis - {data.ActualTime.ToString("HH:mm")}",
                Description = data.Text,
                DateTime = data.ActualTime
            });
        }

        Events.Add(time, eventListData);
    }
  }
}

void InitializeSubscribes()
{
   MessagingCenter.Subscribe<MarkerPage, Markers>(this, "AddMarker", async (sender, marker) =>
   {
      if (marker.Key == null)
      {
         marker.Key = Guid.NewGuid().ToString();
         await Database.Insert(marker);
```

```
        }
        else
        {
            await Database.Update(marker);
        }
    });
}

public class EventData
{
    public string Name { get; set; }
    public string Description { get; set; }
    public DateTime { get; set; }
}
```

- MovesViewModel.cs
- NotesViewModel.cs
- SleepViewModel.cs
- TemperatureViewModel.cs
- ToothsViewModel.cs – šioje klasėje papildomai aprašytos funkcijos:
    - List<KeyValuePair<int, string>> GenerateJawList() – žandikaulio parinkimui;
    - List<KeyValuePair<int, string>> GenerateJawSideList() – žandikaulio pusės parinkimui;
    - List<KeyValuePair<int, string>> GenerateToothsList() skirta pasirinkti, kuris dantis išdygo.

- **Views katalogas** (žr. 11 priedą). Jame sudėtos pagrindinių veiksmų grupių (kontaktai, kasdienė rutina, duomenys) funkciniai klasių ir .xaml tipo failai, kur .xaml failuose aprašoma visas UI, Vaizdo Puslapių (angl. k. ViewPage) klasėse aprašomos funkcijos:
    - Cancel_Clicked(object sender, EventArgs eventArgs) – skirta atšaukti duomenų įrašymą;
    - Save_Clicked(object sender, EventArgs eventArgs) – skirta išsaugoti duomenų įrašymą;

- Delete_Clicked(object sender, EventArgs eventArgs) – skirta ištrinti įrašytus duomenis;

Puslapių (angl.k. Page) klasėse aprašomos funkcijos:

- OnAppearing() – ekrane užkrovus UI puslapį, kur vaizduojami duomenys, juos automatiškai atnaujina;
- OnItemSelected(object sender, SelectedItemChangedEventArgs args) – pažymi iš sąrašo įrašą, kurį norima redaguoti arba ištrinti;
- Add_Clicked(object sender, EventArgs eventArgs) – nukreipia į langą, kuriame galima įvesti duomenis užsaugojimui;

Kataloge esančios klasės:

- Drugs katalogas
  - DrugsPage.xaml, DrugsPage.xaml.cs
- Helpers katalogas
  - Bathing katalogas
    - BathingPage.xaml, BathingPage.xaml.cs, kodas:

```
async void Add_Clicked(object sender, EventArgs eventArgs)
{
    await Navigation.PushAsync(new BathingsViewPage());
}

protected override void OnAppearing()
{
    base.OnAppearing();
    BathingViewModel.BathingViewCommand.Execute(null);
    BathingListView.ItemsSource = BathingViewModel.BathingsCollection;
}

async void OnItemSelected(object sender, SelectedItemChangedEventArgs args)
{
    var bathing = BathingListView.SelectedItem as Bathings;
    if (bathing == null)
    {
        return;
    }
```

```
    BathingViewModel.Bathing = bathing;
    await Navigation.PushAsync(new BathingsViewPage(BathingViewModel));
    BathingListView.SelectedItem = null;
}
```

– BathingsViewPage.xaml, BathingsViewPage.xaml.cs, kodas:

```
async void Save_Clicked(object sender, EventArgs eventArgs)
{
    if (string.IsNullOrEmpty(BathingText.Text))
    {
        await DisplayAlert("Maudymas", "Neįrašytas maudymo tekstas.", "Pildyti");
        return;
    }

    BathingViewModel.Bathing.ActualTime =
StartDatePicker.Date.FormatDateTime(CurrentTimePicker.Time);
    BathingViewModel.Bathing.Text = BathingText.Text;

    MessagingCenter.Send(this, "AddOrUpdateBathing", BathingViewModel.Bathing);

    await Navigation.PopAsync();
}

async void Delete_Clicked(object sender, EventArgs eventArgs)
{
    var deleteBathing = await DisplayAlert(
        "Ištrinti įrašą apie maudymą.",
        "Ar norite ištrinti šį įrašą ?",
        "Taip",
        "Ne");

    if (deleteBathing)
    {
        MessagingCenter.Send(this, "DeleteBathing", BathingViewModel.Bathing);
```

```
    }

    await Navigation.PopAsync();
}
```

- Contacts katalogas
  - CommonContacts.xaml, CommonContacts.xaml.cs
  - CompaniesContacts.xaml, CompaniesContacts.xaml.cs
  - ContactViewPage.xaml, ContactViewPage.xaml.cs
  - DoctorsContacts.xaml, DoctorsContacts.xaml.cs
- Dipering katalogas
  - DiperingPage.xaml, DiperingPage.xaml.cs
  - DiperingViewPage.xaml, DiperingViewPage.xaml.cs
- Drink katalogas
  - DrinksPage.xaml, DrinksPage.xaml.cs
  - DrinksViewPage.xaml, DrinksViewPage.xaml.cs
- Flu katalogas
  - FlusPage.xaml, FlusPage.xaml.cs
  - FlusViewPage.xaml, FlusViewPage.xaml.cs
- Food katalogas
  - ChestFeedPage.xaml, ChestFeedPage.xaml.cs
  - ChestFeedViewPage.xaml, ChestFeedViewPage.xaml.cs
  - FoodFeedPade.xaml, FoodFeedPade.xaml.cs
  - FoodViewPage.xaml, FoodViewPage.xaml.cs
- Health katalogas
  - HealthPage.xaml, HealthPage.xaml.cs
  - HealthViewPage.xaml, HealthViewPage.xaml.cs
- Height katalogas
  - HeightPage.xaml, HeightPage.xaml.cs
  - HeightViewPage.xaml, HeightViewPage.xaml.cs
- Info katalogas
  - InfoPage.xaml, InfoPage.xaml.cs
- Move katalogas
  - MovesPage.xaml, MovesPage.xaml.cs
  - MovesViewPage.xaml, MovesViewPage.xaml.cs
- Note katalogas

        - NotesPage.xaml, NotesPage.xaml.cs
        - NotesViewPage.xaml, NotesViewPage.xaml.cs
    - Sleep katalogas
        - SleepPage.xaml, SleepPage.xaml.cs
        - SleepViewPage.xaml, SleepViewPage.xaml.cs
    - Temperature katalogas
        - TemperaturePage.xaml, TemperaturePage.xaml.cs
        - TemperatureViewPage.xaml, TemperatureViewPage.xaml.cs
    - Tooth katalogas
        - ToothsPage.xaml, ToothsPage.xaml.cs
    - Weight katalogas
        - WeightViewPage.xaml, WeightViewPage.xaml.cs
- About.xaml, About.xaml.cs
- Contacts.xaml, Contacts.xaml.cs
- DailyRoutine.xaml, DailyRoutine.xaml.cs
- Duomenys.xaml, Duomenys.xaml.cs
- Exit.xaml, Exit.xaml.cs
- FoodPage.xaml, FoodPage.xaml.cs
- Info.xaml, Info.xaml.cs
- Login.xaml, Login.xaml.cs
- MainPage.xaml, MainPage.xaml.cs
- Markers.xaml, Markers.xaml.cs
- Pagrindinis.xaml, Pagrindinis.xaml.cs
- Register.xaml, Register.xaml.cs
- RootPage.xaml, RootPage.xaml.cs
- RootPageDetail.xaml, RootPageDetail.xaml.cs
- RootPageMaster.xaml, RootPageMaster.xaml.cs
- RootPageMasterMenuItem.cs
- Sinchronizuoti.xaml, Sinchronizuoti.xaml.cs
- Tooths.xaml, Tooths.xaml.cs
- WeightPage.xaml, WeightPage.xaml.cs

- **App.xaml**
    - App.xaml.cs – pagrindinė klasė, kuri startavimo metu užregistruoja objektus, kurie bus naudojami viso aplikacijos gyvavimo metu. Kodas:

```
public partial class App : Application
{
    public App()
    {
        InitializeComponent();
        DependencyService.Register<ISqlLiteDatabase, SqlLiteDatabase>();
        DependencyService.Register<IMySqlDatabase, MySqlDatabase>();
        DependencyService.Register<IFtpSyncService, FtpSyncService>();
        DependencyService.Register<ICloseApplication, CloseApplication>();
        DependencyService.Register<ISynchronizationService, SynchronizationService>();

        MainPage = new RootPage();
    }
    protected override void OnStart()
    {
    }
    protected override void OnSleep()
    {
    }
    protected override void OnResume()
    {
    }
}
```

2.2.Duomenų sinchronizacija

Duomenų sinchronizacijai naudojama žemiau išvardinta programinė įranga.

- FileZilla [3]Server – ją galima parsisiųsti iš internetinės svetainės.
- FileZilla Client – ją taip pat galima parsisiųsti iš paminėtos aukščiau svetainės.

FileZilla Server programinė įranga naudojama FTPS serveriui paleisti ir keisti jos nustatymus. Atlikus nustatymus, FTPS serveris turėtų būti pasiekiamas internetu. Tam atlikti keičiami naminio maršrutizatoriaus prieigos prievadai (angl. k. port) į FTPS serverį. Taip užtikrinama kaip pavyzdinė debesų saugojimo paslauga duomenims saugoti. Kiekvienas vartotojas prisiregistravęs turėtų savo unikalų identifikacinį numerį sistemoje, pagal kurį galima

[3] https://filezilla-project.org/