İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

Mobil Glukoz Takip Programı

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Tezi

Melike Eymirli

ORCID 0000-0000-0000-0000

Tez Danışmanı: Doç. Dr.Aytuğ Onan

Nisan 2023

Yazarlık Beyanı

Ben, Melike Eymirli, Mobil Glukoz Takip Programı olan bu tezimin ve tezin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 25.04.2023

Mobil Glikoz Takip Programı

Öz

Günümüz teknoloji çağında mobil telefon kullanımı hızla ilerlemektedir. Bu yüzden mobil programlama da her geçen gün yeni uygulamalar gün yüzüne çıkmakta ve kullanılmaktadır. Sağlık alanında da bilişim teknolojileri ile her geçen gün artan internet kullanımı mobil programlamaya da yansımaya başlamıştır. Sağlık sektöründe mobil telefonların kullanımı yaygınlaşmakta ve kullanım miktarı da artmaktadır.

Bu bitirme tezi, Diyabet hastalarının mobil uygulama ile kan glukoz seviyelerini günlük olarak kaydetmesine ve sürekli olarak grafikler ile izlemelerine yardımcı olmak için tasarlanmıştır.

Anahtar Sözcükler: Mühendislik, teknoloji, diyabet, fen bilimleri, araştırma,mobil,glukoz

Teşekkür

Yüksek lisans çalışma sürecimde değerli katkılarını ve desteklerini esirgemeyen, değerli hocam ve sayın danışmanım Sayın Doç. Dr. Aytuğ ONAN'a teşekkürlerimi ve şükranlarımı sunarım.

İçindekiler

Yazarlık Beyanı 2

Öz 3

Teşekkür 4

1 Giriş 5

2 Tezin İçeriği 7

3 Sonuç……28

4 Kaynaklar 29

5 Özgeçmiş 30

Bölüm 1

Giriş

Flutter, Google tarafından geliştirilen açık kaynaklı bir kullanıcı arayüzü (UI) yazılım geliştirme kiti (SDK)'dir. Tek bir kod tabanından mobil, web ve masaüstü platformları için derlenmiş uygulamalar oluşturmayı sağlar. Flutter, geliştiricilerin kodu bir defa yazıp birden çok platformda kullanmasına olanak tanır, bu da geliştirme sürecinde zaman ve çaba tasarrufu sağlar.

Flutter'ın bazı önemli özellikleri ve kavramları şunlardır:

1. Dart Programlama Dili: Flutter, Google tarafından geliştirilen modern ve nesne yönelimli bir programlama dili olan Dart'ı kullanır. Dart, kolay anlaşılabilen ve hızlı bir dil olup, Flutter uygulamalarının temelini oluşturur.
2. Widget'lar: Flutter, kullanıcı arayüzünü oluşturmak için "widget" adı verilen yapıları kullanır. Widget'lar, uygulamanın görünümünü ve davranışını tanımlayan yapı taşlarıdır. Her şey bir widget olarak düşünülür ve bu widget'lar birbirleriyle birleştirilerek karmaşık arayüzler oluşturulur.

3. Hot Reload: Flutter'ın en güçlü özelliklerinden biri olan "Hot Reload", uygulama geliştirme sürecini hızlandırır. Hot Reload, kodun anında değişikliklerinin uygulamaya yansıtılmasını sağlar, böylece geliştirici hızlı bir şekilde değişiklikleri görebilir ve hataları hızlıca düzeltebilir.
4. Platform Bağımsızlık: Flutter, platform bağımsız bir framework olarak çalışır. Aynı kod tabanıyla hem iOS hem de Android için uygulama geliştirmek mümkündür. Ayrıca, web ve masaüstü platformları için de Flutter destek sunmaktadır.
5. Zengin Widget Kütüphanesi: Flutter, zengin bir widget kütüphanesine sahiptir. Bu kütüphane, farklı türdeki kullanıcı arayüz elemanlarını (butonlar, giriş kutuları, listeler, görüntüler vb.) sağlar ve geliştiricilere hızlı bir şekilde uygulama arayüzünü oluşturma imkanı sunar.

Flutter, hızlı, esnek ve verimli bir şekilde platformlar arası uygulamalar geliştirmek için güçlü bir araçtır.

Tezin İçeriği

Flutter projemde kullanmak için databese olarak sqflite kütüphanesi kullanıldı.

Flutter Sqflite, Flutter uygulamalarında yerel veritabanı işlemleri için kullanılan bir pakettir. Sqflite, SQLite veritabanı üzerinde çalışır ve Flutter'ın SQLite desteğini kolaylaştırır.

Sqflite paketi, veritabanı oluşturma, tablo oluşturma, veri ekleme, veri güncelleme, veri sorgulama ve silme gibi temel veritabanı işlemlerini gerçekleştirmek için kullanılır. Flutter uygulamalarında yerel veritabanına ihtiyaç duyulduğunda, Sqflite paketi kullanılarak bu işlemler kolaylıkla gerçekleştirilebilir.

Sqflite paketi, Flutter uygulamalarında yerel veritabanı kullanmanın yanı sıra verilerin etkili bir şekilde yönetilmesini ve depolanmasını sağlar. Veritabanına erişim, Dart programlama diliyle yapılan basit çağrılarla gerçekleştirilir ve SQL sorguları kullanılarak veritabanı işlemleri yapılır.

Flutter Sqflite, özellikle veri yoğun uygulamalar ve çevrimdışı kullanılabilirlik gerektiren uygulamalar için yararlı bir pakettir. Örneğin, not uygulamaları, görev yöneticileri ve çevrimdışı veri senkronizasyonu gerektiren uygulamalar Sqflite paketinden faydalanabilir.

Sqflite paketi, Flutter geliştiricilerine güçlü bir veritabanı çözümü sunar ve veritabanı işlemlerini kolaylaştırarak uygulama performansını artırır.

Uygulama içerisinde dbhelper.dart dosyası içerisinde sqflite database için gerekli tabloların oluşturulması,bu tablolardan okuma yazma ve silme gibi işlemlerin yapılması sağlandı. İlgili doküman içerisindeki kodlar ise;

```
import 'dart:async';

import 'package:melike_deneme/hastalar.dart';
import 'package:melike_deneme/kullanicilar.dart';
import 'package:sqflite/sqflite.dart';
import 'package:path/path.dart';


class DatabaseHelper {
 static final _databeseName = "hastalar3.db";
 static final _databeseVersion = 1;

 static final table = 'hastalar_tabl';
 static final table_kullanici = 'hastalar_kullanici';
 static final columnTc = 'TC';
 static final columnsifre = 'SIFRE';
 static final columnpasif = 'PASIF';
 static final columnid = 'id';
 static final columnadi = 'ADI';
 static final columnsoyadi = 'SOYADI';
 static final columnSonuc = 'Sonuc';
 static final columnSonuctarihi = 'SONUCTARIHI';
 static final columnSonucsaati = 'SONUCSAATI';

 DatabaseHelper._privateConstructor();
 static final DatabaseHelper instance = DatabaseHelper._privateConstructor();
 static Database? _database;

 Future <Database> get database async{
  return _database ??= await _initDatabase();
 }

 _initDatabase() async {
  String path = join(await getDatabasesPath(), _databeseName);
  return await openDatabase(path,
     version: _databeseVersion, onCreate: _onCreate);
 }

 Future _onCreate(Database db, int version) async {
  await db.execute('''
  CREATE TABLE $table(
  $columnid INTEGER PRIMARY KEY AUTOINCREMENT,
  $columnTc INTEGER,
  $columnSonuc TEXT,
  $columnSonuctarihi TEXT,
  $columnSonucsaati TEXT)
  ''');
  await db.execute('''
  CREATE TABLE $table_kullanici(
  $columnTc INTEGER PRIMARY KEY AUTOINCREMENT,
  $columnadi TEXT,
```

```
  $columnsoyadi TEXT,
  $columnsifre TEXT,
  $columnpasif INTEGER)
  ''');
}

Future<int> insert(Hastalar hastalar) async {
  Database db = await instance.database;
  return await db.insert(table, {
    'id': hastalar.id,
    'Tc': hastalar.Tc,
    'Sonuc': hastalar.Sonuc,
    'Sonuctarihi': hastalar.Sonuctarihi,
    'Sonucsaati': hastalar.Sonucsaati
  });
}
Future<int> insert_kullanici(KULLANICILAR kullanicilar) async {
  Database db = await instance.database;
  return await db.insert(table_kullanici, {
    'tc': kullanicilar.tc,
    'adi': kullanicilar.adi,
    'soyadi': kullanicilar.soyadi,
    'sifre': kullanicilar.sifre,
    'pasif': kullanicilar.pasif
  });
}

Future<int> update(Hastalar hs) async {
  Database db = await instance.database;
  int id = hs.toMap()['id'];
  return await db.update(table, hs.toMap(), where: "$columnid = $id");
}

Future<int> delete(int id) async {
  Database db = await instance.database;
  return await db.delete(table, where: '$columnid = ?', whereArgs: [id]);
}

Future<List<Map<String, dynamic>>> queryRows(id) async {
  Database db = await instance.database;
  return await db.query(table, where: "$columnid LIKE '%$id%'");
}
Future<List<Map<String, dynamic>>> querykullanici(tc) async {
  Database db = await instance.database;
  return await db.query(table_kullanici, where: "$columnTc = '$tc'");
}

Future<List<Map<String, dynamic>>> queryALLRows() async {
  Database db = await instance.database;
  return await db.query(table);
}

Future<int?> queryRowCount() async {
  Database db = await instance.database;
  return Sqflite.firstIntValue(
      await db.rawQuery('SELECT COUNT(*) FROM $table'));
}
```

```
}
```

flutter uygulamaları da diğer uygulamalar gibi main dosyası üzerinden başlamaktadır. Main.dart dosyası başlangıçta yine anasayfa.dart dosyası içindeki giriş_sayfası class ına yönlendirerek giriş ekranının açılmasını sağlamaktadır.

Main.dart dosyası içerisindeki kod bilgisi ise;

```
import 'package:flutter/material.dart';
import 'package:melike_deneme/MyHomePage.dart';
import 'Anasayfa.dart';
import 'package:flutter_localizations/flutter_localizations.dart';

void main() {
 runApp(const MyApp());
}

class MyApp extends StatelessWidget {
 const MyApp({super.key});

 @override
 Widget build(BuildContext context) {
  return MaterialApp(
   title: 'Glukoz Ölçüm',
   localizationsDelegates: [
    GlobalMaterialLocalizations.delegate,
    GlobalWidgetsLocalizations.delegate,
   ],
   supportedLocales: [
    const Locale('en', 'US'),
    const Locale('ru', 'RU'),
   ],
   theme: ThemeData(
    primarySwatch: Colors.blue,
   ),
   home: giris_sayfasi(),
   debugShowCheckedModeBanner: false,
   routes:{
  'myhomepage' : (context)=>Anasayfa(),
  }
  );
 }
}
```

şeklinledir.

Anasayfa.dart dosyasına geçtiğimizde giriş sayfasının içerisinde bulunan “Tekrar Hoşgeldiniz.” Yazısının FadeTransition widget i kullanılarak yapıldı.

FadeTransition, Flutter'da kullanılan bir animasyonlu geçiş widget'idir. Bu widget, bir çocuk widget'inin görünürlüğünü değiştirmek ve kaybolma/ortaya çıkma efektleri oluşturmak için kullanılır.

FadeTransition, bir animasyon kontrolcüsü ve bir çocuk widget'i alır. Animasyon kontrolcüsü, çocuk widget'inin görünürlük durumunu kontrol eder ve animasyonun nasıl oynatılacağını belirler. Çocuk widget, görünürlük durumuna göre animasyonlu geçişin etkileneceği widget'i temsil eder.

FadeTransition, animasyonlu geçişin birkaç özelliğini kontrol etmek için kullanılabilen bazı parametrelere sahiptir. Bunlar aşağıdaki bilgileri içermektedir;

Opacity: Geçişin görünürlüğünü belirleyen animasyonun opaklık değeri. 0.0 (tamamen kaybolmuş) ile 1.0 (tamamen görünür) arasında bir değer alır.

alwaysIncludeSemantics: Erişilebilirlik için önemli olan bir özellik. Eğer true olarak ayarlanırsa, geçiş sırasında çocuk widget'in erişilebilirlik bilgileri dahil edilir.

Örnek Kullanım:

dart

Copy code

```
FadeTransition(
  opacity: _animation, // Animasyon kontrolcüsü
  child: Container(
    width: 200,
    height: 200,
```

```
    color: Colors.blue,
  ),
)
```

Bu örnekte _animation bir animasyon kontrolcüsüdür ve FadeTransition widget'ı, çocuk widget olan bir mavi bir konteynerin görünürlüğünü kontrol eder. _animation değeri, opaklık değerini belirler ve çocuk widget'inin görünürlüğünü değiştirmek için kullanılır.

Bu şekilde, FadeTransition ile animasyonlu geçişler oluşturabilir ve çocuk widget'inin kaybolma ve ortaya çıkma efektlerini kontrol edebilirsiniz.

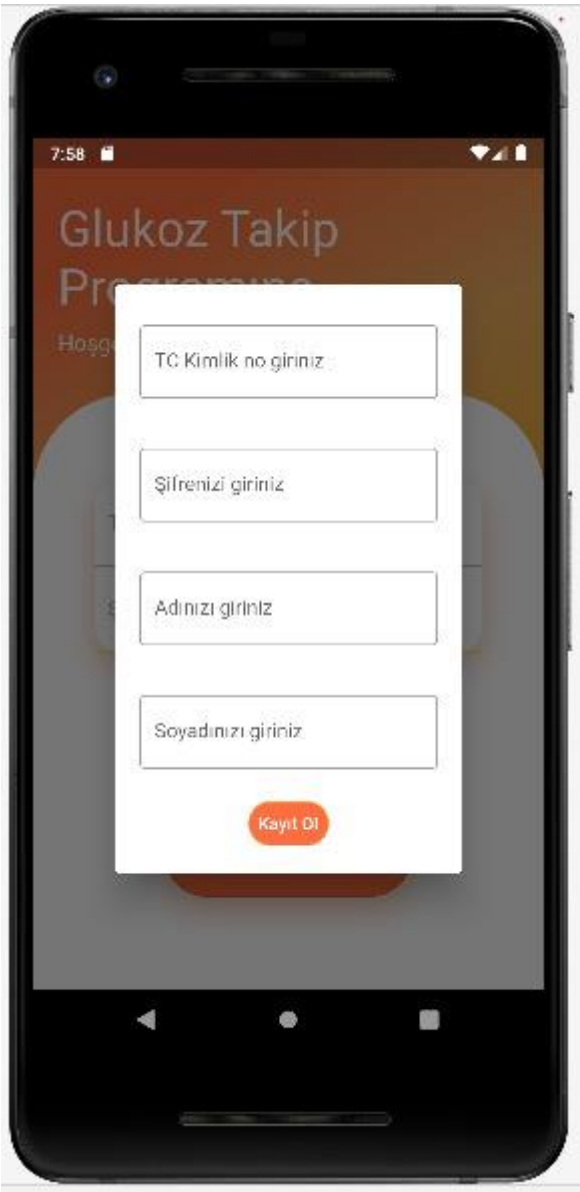
Bir sonraki alanda ise TC kimlik numarası için yine flutter ‘in TextField widget ‘ında inputFormatters özelliğinde sadece rakam girilmesi için formatlama kullandım.

```
TextField(controller: tcController,
  keyboardType: TextInputType.number,
  inputFormatters: [FilteringTextInputFormatter.digitsOnly],
  decoration: InputDecoration(
     hintText: "Tc Kimlik No:",
     hintStyle: TextStyle (color: Colors.grey),
     border: InputBorder.none
  ),
),
```

Şifre kısmında ki bulunan TextField için ise kişiler karakterde girilebileceği için digital giriş engeli koyulmamıştır. Onun yerine şifre kısmının gözükmemesi için *** gizlemesi kullanılmıştır. Bunun içinde TextField widget ‘inda özellik olarak obscureText:true özelliği kullanılmıştır.

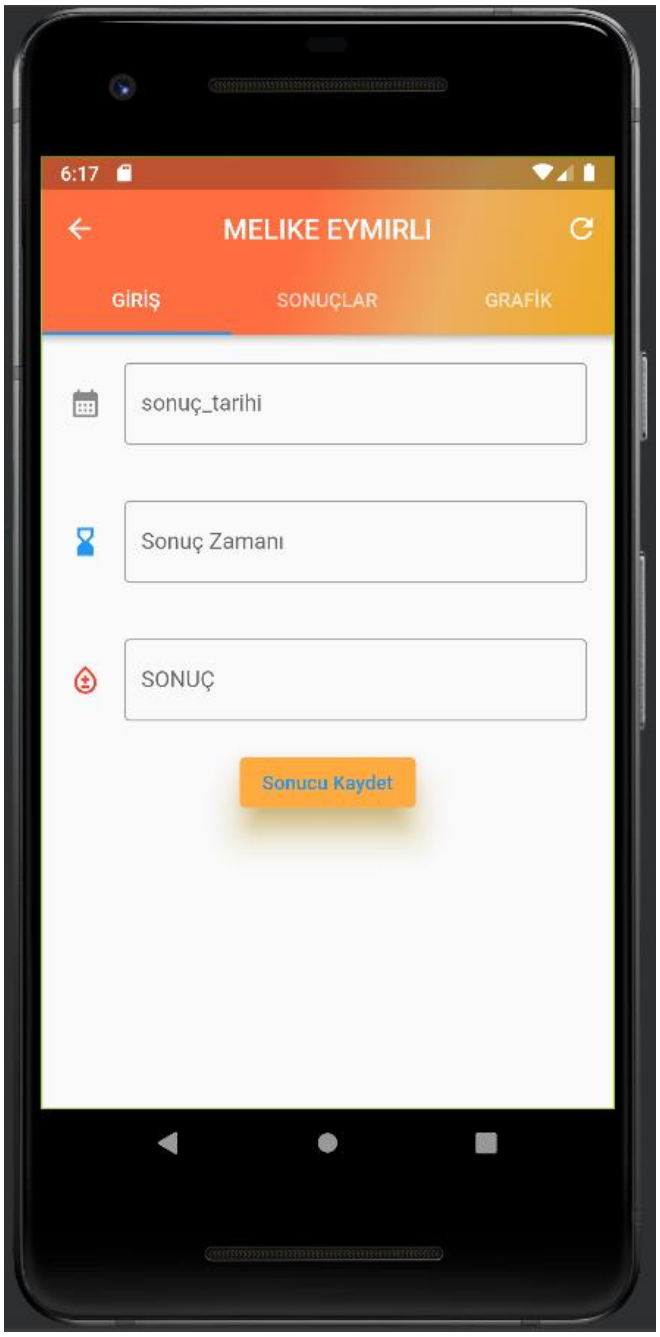
Yeni üye işlemi için GestureDetector ve showDialog widget ları kullanılmıştır.

showDialog için 4 adet Textfield ve 1 adet Container widget’ı kullanılmıştır. Sqflite ile hastalar_kullanici tablosu içerisine burada yazılı olan bilgiler eklenerek yeni kullanıcı oluşturulmaktadır.



Oluşturulan bu kullanıcı bilgileri daha sonra giriş ekranında bulunan TC şifre gibi alanlarda sqflite databese ile kontrol edilerek kullanıcı girişi sağlanmaktadır.

Kullanıcı girişleri yapıldıktan sonra bir parametre yardımı ile kişi bilgileri yine sqflite ile sorgulanarak bir sonraki açılacak ekrana gönderilmektedir.



MyHomePage.dart dosyasındaki Anasayfa Class içine gönderilen parametreler bir Appbar yardımı ile başlık olarak gözükmete ve yine appbar içerisinde DefaultTabController kullanılarak tab yapısı oluşturulmuştur.

Anasayfa class'i StatefulWidget ile yaptığım görülmekte

StatefulWidget sınıfı, Flutter'da kullanıcı arayüzünün durumunu tutan ve dinamik olarak güncellenebilen widget'ları oluşturmak için kullanılan bir sınıftır.

Bir StatefulWidget widget'i, bir durum nesnesi olan State ile eşleştirilir ve bu durum nesnesi widget'in görünümünü güncellemek için kullanılır.

StatefulWidget'lar, uygulamadaki değişkenlik gerektiren durumları yönetmek için kullanılır. Örneğin, bir sayaç uygulamasında, sayaç değeri zaman içinde artabilir veya azalabilir. Bu durumu tutmak ve güncellemek için StatefulWidget kullanılır.

StatefulWidget sınıfının kendisi genellikle değişmezdir ve build yöntemi aracılığıyla State nesnesini oluşturur. State nesnesi, StatefulWidget'ın durumunu temsil eder ve StatefulWidget'ın yaşam döngüsü boyunca güncellemeler yapılabilen verileri içerir.

StatefulWidget ve State arasındaki ilişki şu şekildedir: Her bir StatefulWidget, bir State nesnesi ile ilişkilendirilir ve State nesnesi, StatefulWidget'ın durumunu tutar. StatefulWidget'daki bir değişiklik, State nesnesindeki durumu günceller ve bu durum değişikliği otomatik olarak widget'in yeniden çizilmesini tetikler.

StatefulWidget sınıfı, kullanıcı arayüzünün dinamik olarak güncellenmesi gereken durumları yönetmek için kullanışlı bir yapı sağlar. Bu sayede, uygulama içindeki veri veya kullanıcı etkileşimleri gibi değişiklikleri yansıtmak ve güncellemek mümkün olur. Bende bu uygulamada “SONUÇLAR” ,”GRAFİK” tablarında ekranın yenilenmesi ve verilerin ekranda güncellenmesi için bu yapıyı kullanmam gerekiyordu.

“GİRİŞ” tab için veri girişlerinin yapılabilmesi için gerekli TextField ler kullanarak veri girişlerinin sağlanmasını ve bu verilerin sqflite ile db üzerine yazılması için “Sonucu Kaydet” butonu eklendi.

‘sonuc_tarihi’ alanı için kullanıcıların her birinin giriş şekilleri farklı olabileceğini düşünerek bu alanların sabit olması için elle girişi engelledim. TextField widget i için bir controller ve onTap özellikleri ekledim. onTap özelliği ile showDatePicker fonksiyonu kullanarak tarih bilgisi seçebileceği bir yapı eklenmiştir.

showDatePicker fonksiyonu, Flutter framework'ünde kullanılan bir yöntemdir ve bir tarih seçiciyi kullanıcılara göstermek için kullanılır. Bu fonksiyon, kullanıcıya bir takvim görüntüsü sunar ve kullanıcının bir tarih seçmesine izin verir.

showDatePicker fonksiyonunun birçok parametresi vardır ve bunlar şunlardır:

context: BuildContext türünde bir değerdir ve mevcut widget ağacındaki konumu belirtir.

initialDate: Tarih seçici açıldığında varsayılan olarak gösterilecek tarihi belirtir.

firstDate: Kullanıcının seçebileceği ilk tarihi belirtir. Örneğin, bu parametre bugünkü tarihi belirleyerek, kullanıcının geçmiş tarihleri seçmesini engelleyebilir.

lastDate: Kullanıcının seçebileceği son tarihi belirtir. Örneğin, bu parametre gelecekteki bir tarihi belirleyerek, kullanıcının sadece gelecek tarihleri seçmesini sağlayabilir.

builder: Opsiyonel bir parametredir ve kullanıcıya takvim görünümünü nasıl özelleştirebileceğini belirtir.

locale: Takvim görünümünün hangi dil ve bölgeye ait olacağını belirtir. Örneğin, "tr_TR" Türkçe dil ve Türkiye bölgesini temsil eder.

useRootNavigator: Eğer değeri true ise, kök navigator kullanılır. Bu genellikle Dialog içinde kullanırken işe yarar.

Programdaki kullanımı:

```
onTap: () async{
  DateTime? pickeddate= await showDatePicker(context: context,  locale : const Locale('tr','TR'),
initialDate: DateTime.now(),
      firstDate: DateTime(2000), lastDate: DateTime(2101));
  if (pickeddate !=null){
    setState(() {
      sonuc_tarihiController.text=DateFormat('dd.MM.yyyy').format(pickeddate);
    }
    );
  }
},
```

Programdaki Görsel:

Sonuç Zamanı TextField için ise showTimePicker fonksiyonu kullanılmıştır.

showTimePicker fonksiyonu, Flutter framework'ünde kullanılan bir yöntemdir ve bir saat seçiciyi kullanıcılara göstermek için kullanılır. Bu fonksiyon, kullanıcıya bir saat seçimi arayüzü sunar ve kullanıcının bir saat seçmesine izin verir.

showTimePicker fonksiyonunun birçok parametresi vardır ve bunlar şunlardır:

context: BuildContext türünde bir değerdir ve mevcut widget ağacındaki konumu belirtir.

initialTime: Saat seçici açıldığında varsayılan olarak gösterilecek saat değerini belirtir.

builder: Opsiyonel bir parametredir ve kullanıcıya saat görünümünü nasıl özelleştirebileceğini belirtir.

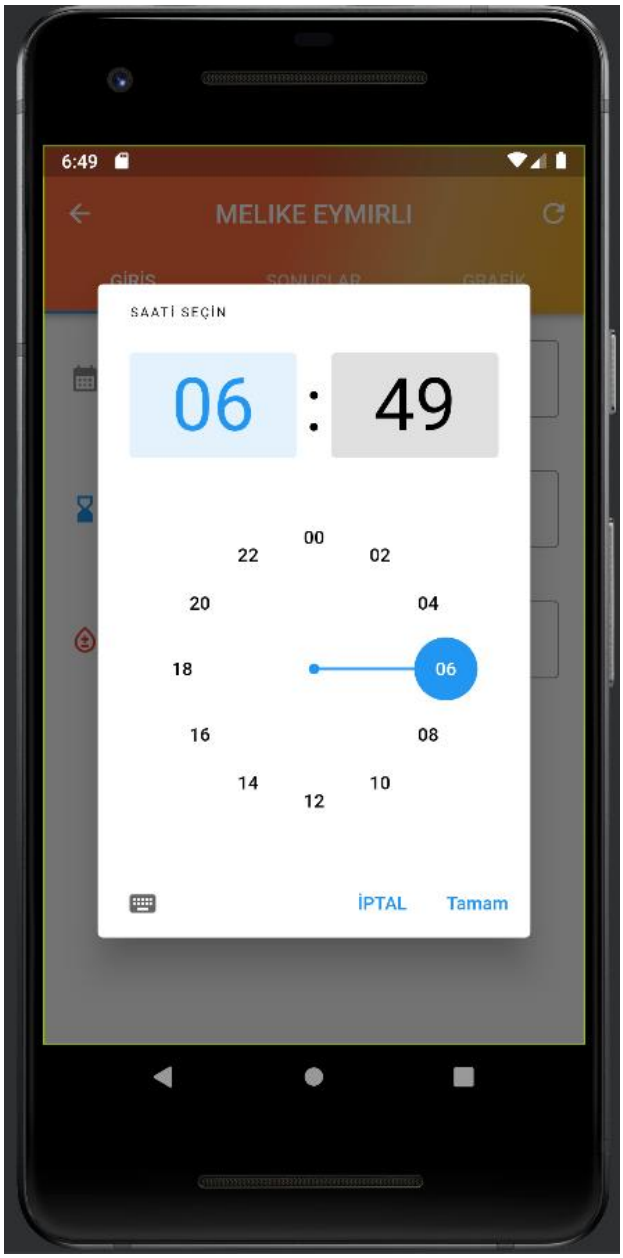
useRootNavigator: Eğer değeri true ise, kök navigator kullanılır. Bu genellikle Dialog içinde kullanırken işe yarar.

Programdaki Kullanım Şekli:

```
onTap: () async{

 var pickedtime= await showTimePicker(
  context: context,
  initialTime: TimeOfDay.now(),
  builder: (context, child) {
   if (MediaQuery.of(context).alwaysUse24HourFormat) {
    return child!;
   } else {
    return Localizations.override(
     context: context,
     locale: Locale('tr', 'TR'),
     child: child!,
    );
   }
  },
 );

 if (pickedtime!=null){
  setState(() {
  sonuc_zamaniController.text= '${pickedtime.hour.toString().padLeft(2,
"0")}:${pickedtime.minute.toString().padLeft(2, "0")}';
  });
```

```
  //DateFormat() is from intl package, you can format the time on any pattern you need.
 }else{
  sonuc_zamaniController.text=("Time is not selected");
 }
},
```

Programdaki Görsel:



SONUÇLAR tab'ı için ise bir ListView Widget kullanılmıştır.

ListView, Flutter framework'ünde kullanılan bir widget'tır ve dikey veya yatay yönlü bir liste oluşturmak için kullanılır. ListView, birçok widget'ı dikey veya yatay bir

düzende sıralamak ve kullanıcının liste içinde kaydırmasını sağlamak için kullanılabilir.

ListView, elemanlarını kaydırmalı bir şekilde görüntülemek için otomatik olarak kaydırma işlevselliğini sağlar. Bu sayede, liste içinde fazla sayıda eleman bulunması durumunda kullanıcının içeriği görüntülemek için kaydırmasına olanak tanır.

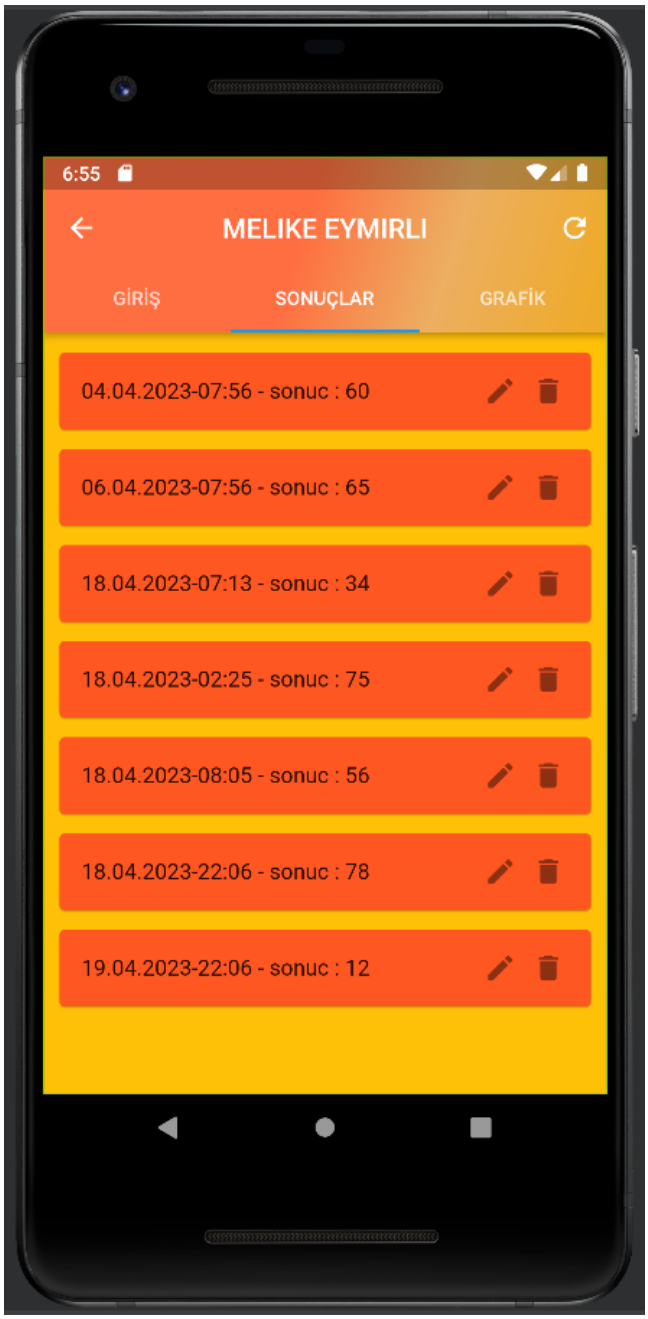
ListView widget'ı, verileri statik veya dinamik olarak listelemek için kullanılabilir. Öğeler, ListView.builder veya ListView.separated gibi farklı constructor'lar kullanılarak oluşturulan widget'larla liste içine eklenir.

Program içi Kullanımı:

```
ListView.builder(
  padding: EdgeInsets.all(8),
  itemCount: hastalar.length+1,
  itemBuilder: (BuildContext context, int index){
    if (index==hastalar.length){
      return  null;
      //  SingleChildScrollView(
      //  child: DropdownButton<String>(value:dropdownvalue,
      //  items: sonuc_tarihleri.map((item) => DropdownMenuItem<String>(value:item, child:
Text(item,style:TextStyle(fontSize: 24)),
      // )).toList(),
      //onChanged: (item)=>setState(() => dropdownvalue=item),
      //),
      //);*/
    }
    return Container(
      height: 70,
      child: Center(
        child: Card(
          color: Colors.deepOrange,
          child: ListTile(
            title:  Text(
              '${hastalar[index].Sonuctarihi}-${hastalar[index].Sonucsaati} - sonuc :
${hastalar[index].Sonuc} '
            ),trailing: Container(
            width: 70,
            child: Row(
              children: [
                Expanded(child: IconButton(onPressed: (){
                  int? id1=hastalar[index].id;
                  sonuctarihiUpdateController.text ='${hastalar[index].Sonuctarihi}';
                  sonucsaatiUpdateController.text ='${hastalar[index].Sonucsaati}';
                  showDialog(context: context, builder: (context)=>SimpleDialog(
                    children: [
                      SingleChildScrollView(
                        child: Column(
```

```
                    children: [
                      Container(
                        padding:EdgeInsets.all(20),
                        child: TextField(
                          controller: sonucUpdateController,
                          decoration: InputDecoration(
                              border: OutlineInputBorder(),
                              labelText: 'SONUÇ',
                              icon: Icon(Icons.bloodtype_outlined,color: Colors.red,)
                          ),
                        ),
                      ),

                      Container(width:100,height:50,
                        child: OutlinedButton(
                            child: Text("Guncelle"),
                            onPressed: (){
                              int? id =id1;
                              String sonuc=sonucUpdateController.text;
                              String sonuctarihi=sonuctarihiUpdateController.text;
                              String sonucsaati=sonucsaatiUpdateController.text;
                              _Update(id,tc1,sonuctarihi,sonucsaati,sonuc);

                            }

                        ),
                      ),
                    ],

                  ),
                ),
              ],
            )
            );
          }, icon: Icon(Icons.edit))),
          Expanded(child: IconButton(onPressed: (){
            var a = '${hastalar[index].id}';
            _delete(int.parse(a));
          }, icon: Icon(Icons.delete))),
        ],
      ),
    ),
    ),
  ),
),
);

},
)
```

Program Görseli:



List içerisinde düzenleme ve silme işlemleri yapılabilmektedir. Yapılan değişiklerin görünmesi ve Güncellemeler için ise TabView içinde bulunan sağ üt taraftaki refresh butonu kullanılmaktadır.

Silme işlmeleri dbHelper.dart dosyası ile sqflite üzerinden id bilgisi gönderilerek silinmektedir.

Program kodları :

MyHomePage.dart içerisinde

```
void _delete(int id) async{
  final rowsdelete =await dbHelper.delete(id);
  _showMassageInScaffold("deleted $rowsdelete row(s):rows $id");
}
```

dbhelper.dart içerisinde

```
Future<int> delete(int id) async {
  Database db = await instance.database;
  return await db.delete(table, where: '$columnid = ?', whereArgs: [id]);
}
```

Şeklinde yazılmaktadır.

GRAFİK tab'ı için ise SfCartesianChart kullanılmıştır.

SfCartesianChart, Syncfusion tarafından geliştirilen bir Flutter paketi olan Flutter Charts'un bir parçasıdır. SfCartesianChart, verileri temsil etmek için çizgi grafikleri, alan grafikleri, sütun grafikleri, çubuk grafikleri, dağılım grafikleri ve daha birçok grafik türünü destekler.

SfCartesianChart, verileri alır ve bunları x ve y koordinatlarına göre bir grafikte görselleştirir. Bu grafikler, verilerin analiz edilmesi, eğilimlerin gösterilmesi veya görsel olarak sunumun yapılması gibi çeşitli veri görselleştirme senaryolarında kullanılabilir.

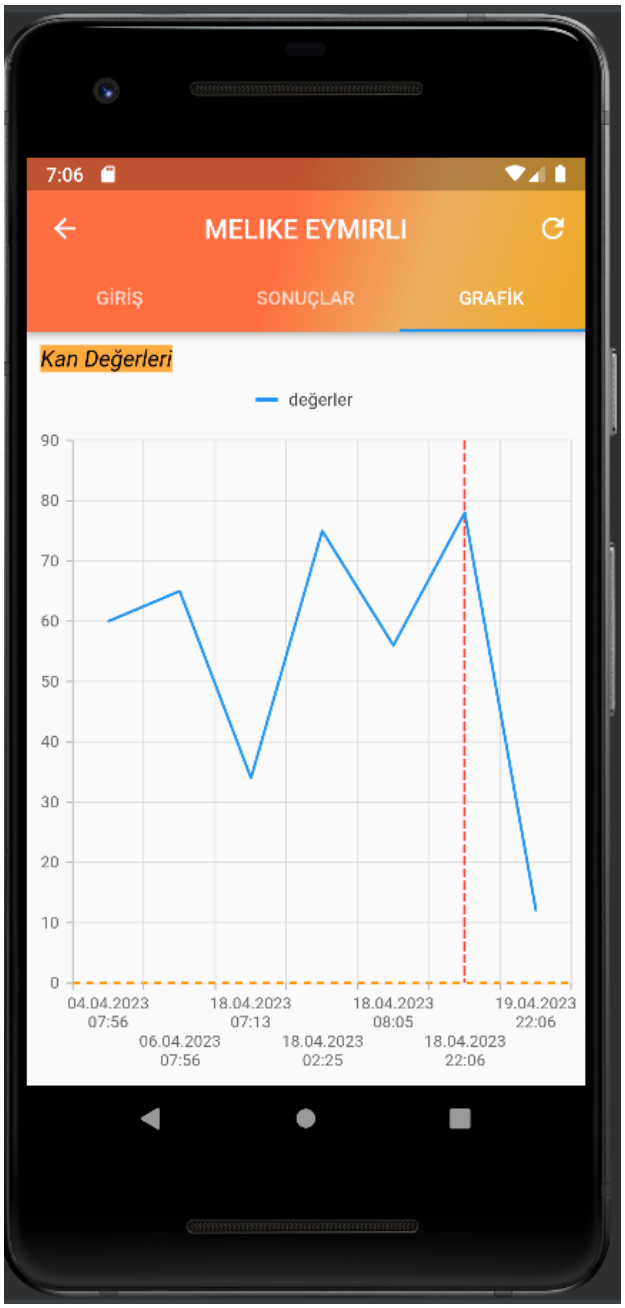
SfCartesianChart widget'ı, verileri grafiğe aktarmak ve grafiği özelleştirmek için bir dizi özelliğe sahiptir. Bunlar arasında veri serileri, eksenler, etiketler, başlıklar, arka planlar, kaydırma davranışı, lejantlar vb. yer alır.

Burada SfCartesianChart için çizgi grafik türü kullanılmıştır.

Programdaki kullanımı:

```
Container(
                child: SfCartesianChart(title: ChartTitle(
                  text: 'Kan Değerleri',
                  backgroundColor: Colors.orangeAccent,
                  // Aligns the chart title to left
                  alignment: ChartAlignment.near,
                  textStyle: TextStyle(
                    color: Colors.black,
                    fontFamily: 'Roboto',
                    fontStyle: FontStyle.italic,
                    fontSize: 14,
                  ),
                ),
                  legend: Legend(isVisible:true),
                  primaryXAxis: CategoryAxis(
                    axisLine: AxisLine(
                      color: Colors.orange,
                      width: 2,
                      dashArray: <double>[5,5]),
                    labelIntersectAction: AxisLabelIntersectAction.multipleRows,
   plotBands: <PlotBand>[
   PlotBand(color: Colors.red,
borderWidth: 2,
borderColor: Colors.red,
dashArray: const <double>[4,5],
   start: position,
   end: position,
   ),]
                ),
                  series: <ChartSeries>[
                    LineSeries<Hastalar,String?>(dataSource: hastalar, name: 'değerler',color:
Colors.blue,xValueMapper: (Hastalar details,_)=>('${details.Sonuctarihi}\n${details.Sonucsaati}'),
                      yValueMapper: (Hastalar details,_)=>int.parse(details.Sonuc.toString())),
                  ],
                ),
              ),
```

Program Görseli:



Grafik üzerinde değeri en yüksek olan hasta ölçümü için kırmızı kesik çizgi kullanılmıştır.

.

Sonuç:

Bu çalışmada, glukoz takip programının etkinliğini değerlendirdik ve elde ettiğimiz sonuçlar aşağıdaki gibi özetlenebilir:

Glukoz takip programı, kullanıcıların glukoz düzeylerini sürekli olarak izlemelerine ve takip etmelerine olanak sağlamıştır.

Kullanıcıların glukoz düzeylerini program üzerinden kaydetmeleri, veri toplama sürecini kolaylaştırmış ve kullanıcıların glukoz düzeyleri üzerindeki değişimleri izlemelerini sağlamıştır.

Programın veri analiz özellikleri, kullanıcıların glukoz düzeylerindeki eğilimleri ve paternleri tespit etmelerine yardımcı olmuştur.

Kullanıcıların glukoz düzeyleri hedef aralıklarının dışına çıktığında, program kırmızı renk ile kullanıcılara uyarılar göndermiştir. Bu özellik, kullanıcıların glukoz düzeylerini kontrol altında tutmalarına yardımcı olmuştur.

Sonuç olarak, glukoz takip programı, kullanıcıların glukoz düzeylerini etkin bir şekilde takip etmelerine ve glukoz kontrolünü sağlamalarına yardımcı olmuştur. Bu programın diyabet yönetimi için önemli bir araç olduğu sonucuna varabiliriz. Ancak, daha fazla araştırma ve kullanıcı deneyimi geri bildirimleri, programın iyileştirilmesi ve kullanılabilirliğinin artırılması açısından önemlidir.

Kaynaklar

https://pub.dev/

Özgeçmiş

Adı Soyadı: Melike Eymirli

Eğitim:

2022–2023 İzmir Kâtip Çelebi Üniversitesi, Yazılım Müh. Bölümü

İş Deneyimi:

2012 – Probel Yazılım A.Ş - ARGE İş Analisti
Kd. ARGE Test Uzmanı