



# Flutter ile To Do List

Berat Akay

19290204

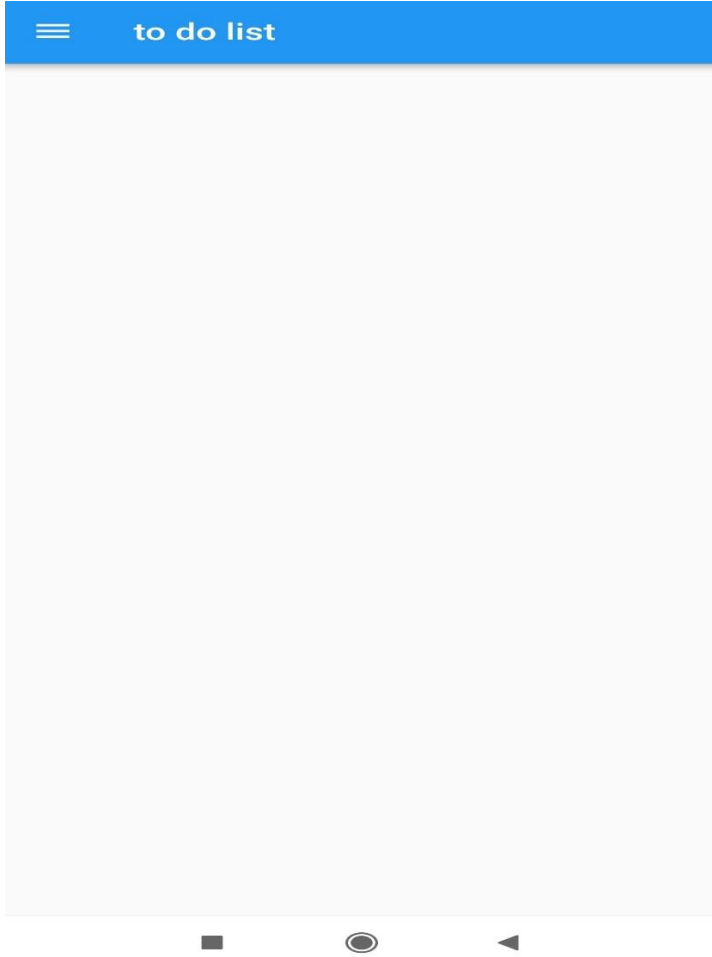
BLM4537

## PROJE TANITIMI

Bu proje dart dili vasıtasıyla flutter ile to do uygulaması geliştirilmiştir. Öncelikle, android studio geliştirme ortamında yeni bir Flutter projesi kurdum. Yapılacaklar listenizin düzenini tasarladım. Bu, yapılacaklar listesi, yeni öğeler eklemek için bir giriş alanı ve öğeleri tamamlandı olarak işaretlemek veya listeden kaldırmak için düğmeler içeren yapı olarak tanımlayabiliriz. Görev adı ve tamamlanma durumu gibi özellikler de dahil olmak üzere her yapılacak iş öğesini temsil edecek bir veri modeli oluşturdum. Giriş alanını ve bir düğmeyi kullanarak listeye yeni yapılacaklar ekleme özelliğini uygulamaya çalıştım. Öğeler listesini durum bilgisi olan bir pencere öğesinde saklamanız ve yeni bir öğe eklendiğinde durumu güncellemeniz gerekir. Öğeleri tamamlandı olarak işaretleme veya listeden kaldırma özelliğini uygulamaya çalıştım. Bu, widget'ın durumunun güncellenmesini de içeriyor. Öğeleri düzenleme veya yeniden düzenleme veya listeyi uygulama başlatmaları arasında devam ettirme gibi istediğiniz ek özellikleri de uygulama üzerinden yapabiliriz.

## Giriş Ekranı

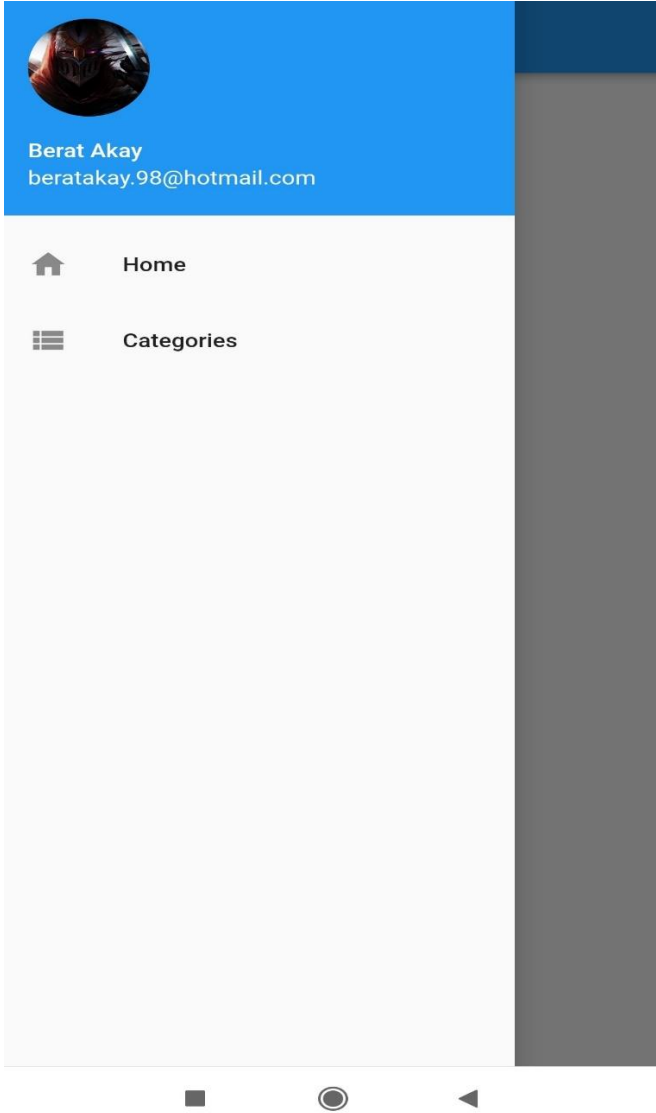
Uygulamanın giriş ekranıyla alakalı görsel aşağıdadır. Giriş ekranı kullanıcının uygulamaya daha odaklanabilir olması amaçlanarak daha sade tutulmaya çalışılmıştır.



Uygulama'nın ana ekranında uygulamanın ismi ve bir tane bar bulunmaktadır. Bar'a basıldığında kategori ve ana sayfa ekranları çıkmaktadır.

## Bar Sekmesi

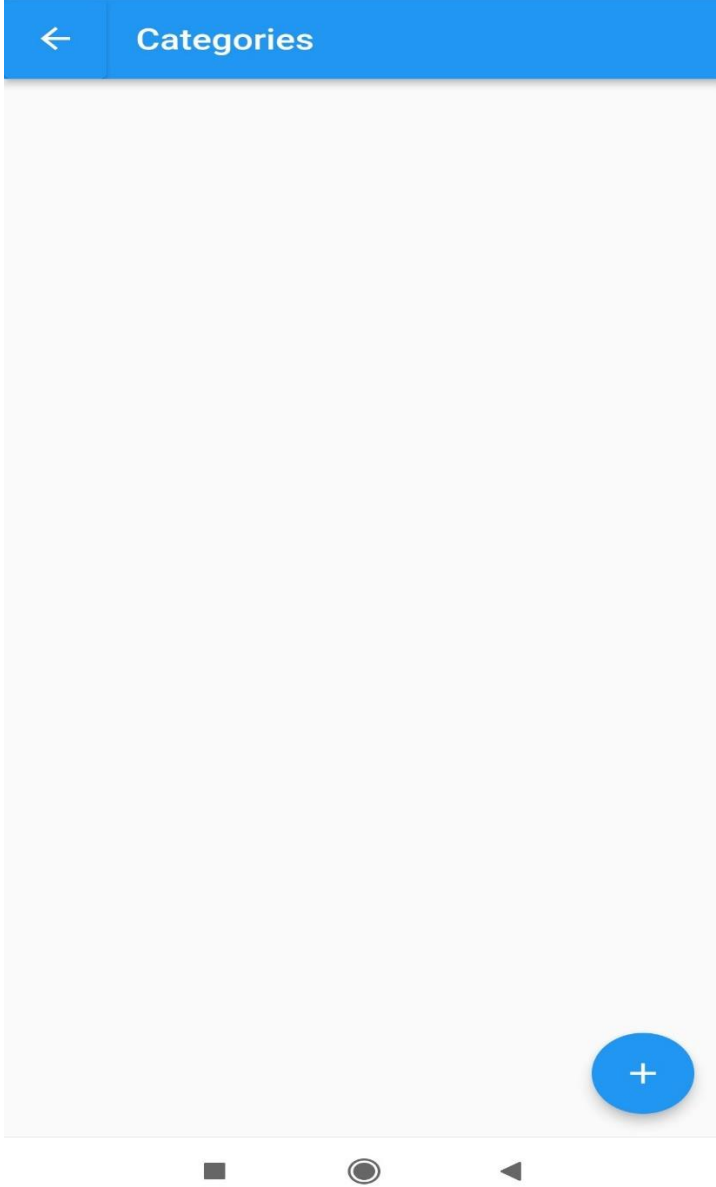
Ana sayfa'da bar sekmesine basılmasının ardından 2 tane buton bulunmaktadır. Bunlar kategori ve ana sayfa butonlarıdır. Ayrıca sol üst köşede ise kullanıcının profil fotosu ve email adresi yer almaktadır.



Yukarıda verilen şekilde görüldüğü üzere home butonuna basıldığı takdirde tekrardan ana sayfaya yönlendirecektir. Categories butonuna basıldığı zaman ise kategori sekmesine götürecektir.

## Kategori Ekranı

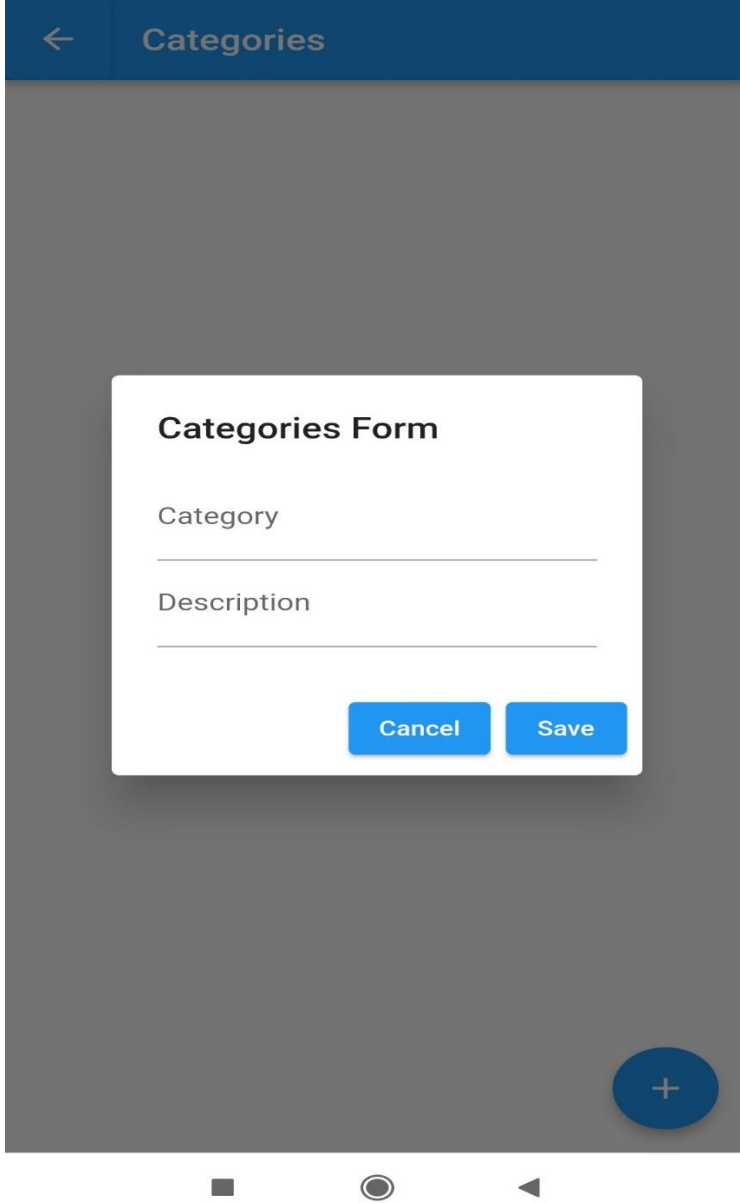
Bar bölmesinde kategori butonuna basıldığında kategori sayfasına yönlendirilir. O sekmede ise kategori ekleme tuşu bulunmaktadır.



Öncelikle daha kullanışlı ve odaklanabilmesi kolay olması açısından projenin genelinde ekranlar sade tutulmaya çalışılmıştır. Kategori eklemek için sağ alt köşeye bir buton eklenmiştir. Bu vesileyle eklenecek kategorileri database vasıtasıyla depolanabilecektir. Ve sorgular yapılacaktır.

## Kategori Ekleme

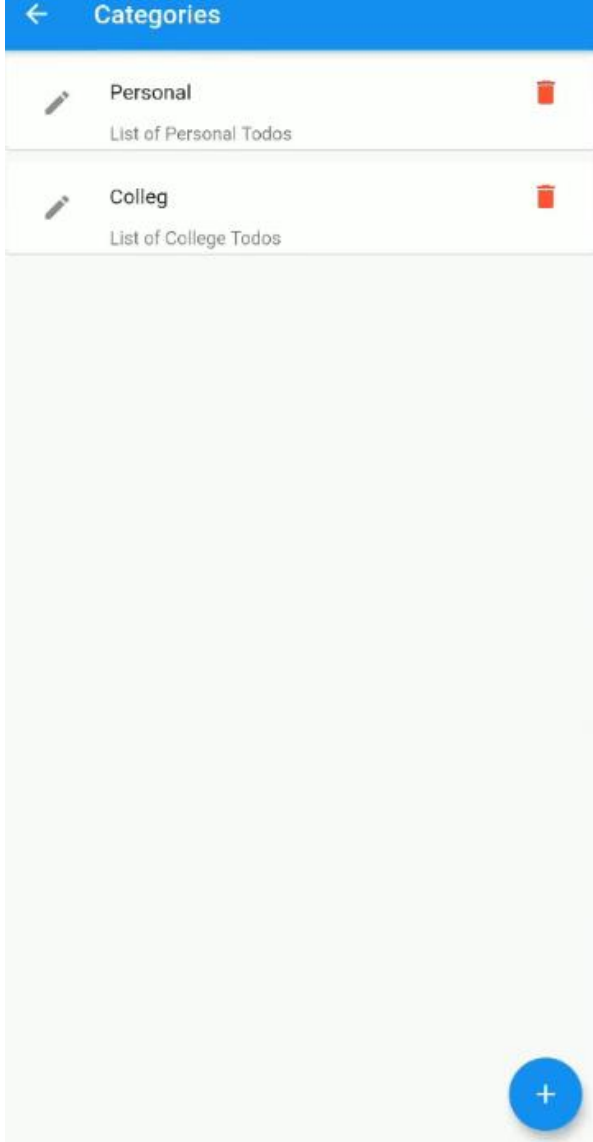
İlk başta kategori ekleme ekranından sağ altta bulunan butona basılıp kategori ekleme kısmına geçiş yapılır.

The screenshot shows a mobile application interface. At the top, there is a dark blue header bar with a white back arrow on the left and the word 'Categories' in white text. The main background is a solid grey color. In the center, a white modal box titled 'Categories Form' is displayed. Inside this modal, there are two text input fields: the first is labeled 'Category' and the second is labeled 'Description'. Below these fields, there are two blue buttons: 'Cancel' on the left and 'Save' on the right. In the bottom right corner of the grey background, there is a dark blue circular button with a white plus sign. At the very bottom of the screen, there are three small, faint icons: a square, a circle, and a triangle.

Butona basıldıktan sonra categories form adında bir form çıkar ve bu form kategoriyle alakalı bilgilerin alacağı bölümdür. İlk labelde kategorinin ismi 2. Labelde ise kategorinin açıklaması yapılır. Sonrasında ise Save butonu vasıtasıyla database'e kaydedilir. Cancel butonuyla ise işlem iptal edilir.

## Kategori Ana Sayfa

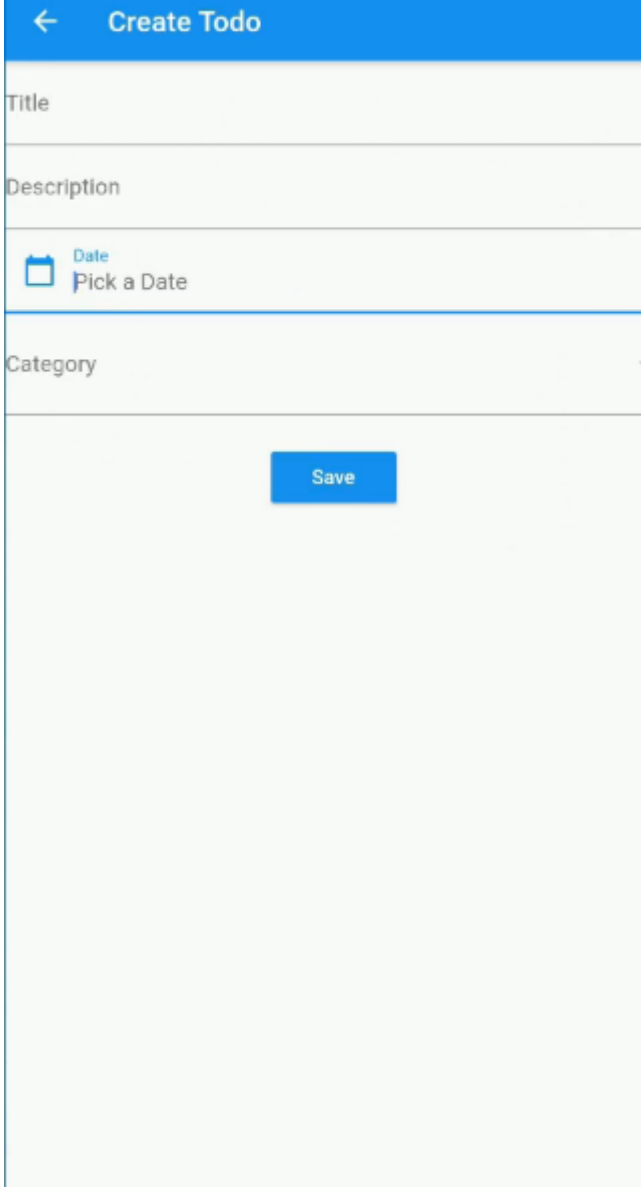
Kategori ekranında eklenen datalar, database'e eklenen veriler kategoriler sayfasında yer alacaktır.



Yukarıdaki şekilde de görüldüğü gibi personel ve colleg kategorileri oluşturulmuştur. Bu veriler databaseden çekilecektir. Bu kategoriler üzerinde düzenleme ve silme işlemleri de yapılabilmektedir.

## Yapılacakların Eklenmesi

Projemizin son kısmında ise oluşturulan kategorilerin içine yapılacaklar eklenir. Yapılacak iş kategorize edilerek daha kullanışlı ve daha planlı hareket edilmiş olur.

A screenshot of a mobile application interface for creating a todo item. The form has a blue header bar with a back arrow and the text 'Create Todo'. Below the header, there are four input fields: 'Title', 'Description', 'Date' (with a calendar icon and the text 'Pick a Date'), and 'Category'. At the bottom of the form, there is a blue 'Save' button.

Yukarıda görüldüğü üzere yapılacak işin ismi, açıklaması, hangi tarihte yapılacağı ve son olarak kategorisi belirtilir. Bu aşamalar gerçekleştirildikten sonra yapılacak iş kategorisine eklenmiş olur ve veritabanına eklenir.



```

import 'package:flutter/material.dart';
import 'package:todoist/helpers/drawer_navigation.dart';
import 'package:todoist/src/app.dart';

class HomeScreen extends StatefulWidget {
  const HomeScreen({Key? key}) : super(key: key);

  @override
  State<HomeScreen> createState() => _HomeScreenState();
}

class _HomeScreenState extends State<HomeScreen> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('to do list'),
      ), // AppBar
      drawer: DrawerNavigation(),
    ); // Scaffold
  }
}

```

Bu kod, flutter paketinin material.dart kitaplığında bir HomeScreen parçacığı tanımlar. Durum bilgisi olan bir pencere ögesidir, yani parçacığın ömrü boyunca değişebilen değişken bir duruma sahiptir. HomeScreen widget'ı, StatefulWidget sınıfını genişleten ve \_HomeScreenState sınıfının bir örneğini döndürmek için createState yöntemini geçersiz kılan HomeScreen sınıfı kullanılarak oluşturulur. \_HomeScreenState sınıfının derleme yöntemi, bir çekmecedeki AppBar parçacığı ve DrawerNavigation pencere ögesi içeren bir İskele parçacığı döndürür. Scaffold widget'ı, bir uygulama için tutarlı bir görsel yapı sağlayan temel bir malzeme tasarımı yerleşim yapısıdır. Genellikle tek bir sayfa içerir ve bir üst uygulama çubuğu, bir alt gezinme çubuğu ve bir çekmece içerir. AppBar pencere ögesi, genellikle uygulamanın adı olan bir başlık görüntüleyen bir üst uygulama çubuğudur. DrawerNavigation widget'ı, çekmece\_navigation.dart dosyasında tanımlanan özel bir widget'tır. Alınabilecek öğelerin listesini içeren bir çekmeceyi göstermek için kullanılır. uygulamanın farklı bölümlerine gitmek için seçilir.

```

import 'package:todoist/repositories/database_connection.dart';
import 'package:sqflite/sqflite.dart';
import 'package:sqflite/sql.dart';

class Repository{
  DatabaseConnection? _databaseConnection;
  Repository(){
    _databaseConnection =DatabaseConnection();
  }

  Database? _database;

  Future<Database?> get database async{
    if(_database!= null) return _database;
    _database= await _databaseConnection?.setDatabase();
    return _database;
  }

  insertData(table, data) async{
    var connection = await database;
    return await connection?.insert(table, data);
  }

  readData(table)async{
    var connection = await database;
    return await connection?.query(table);
  }
}

```

Bu kod, bir veritabanına bağlantı sağlayan bir Repository sınıfını ve veritabanıyla etkileşim kurma yöntemlerini tanımlar. Depo sınıfı, DatabaseConnection türünde özel bir \_databaseConnection alanına sahiptir ve Database türünde bir özel alan \_database?, DatabaseConnection sınıfı, database\_connection.dart dosyasında tanımlanır ve veritabanına bağlantı oluşturmak için kullanılır. \_database alanı, oluşturulduktan sonra veritabanına bir referans depolamak için kullanılır. Repository sınıfı, \_databaseConnection alanını DatabaseConnection sınıfının bir örneğiyle başlatan bir oluşturucuya sahiptir. Ayrıca, \_database alanının değerini içeren bir Gelecek döndüren bir veritabanı alıcı yöntemine de sahiptir. \_database alanı boşsa, veritabanı alıcı yöntemi, \_databaseConnection alanının setDatabase yöntemini çağırarak veritabanıyla zaman uyumsuz bir bağlantı oluşturur ve elde edilen veritabanı başvurusunu \_database alanında depolar.

```
import 'package:todolist/models/category.dart';
import 'package:todolist/repositories/repository.dart';

class CategoryService{
  Repository? _repository;

  CategoryService(){
    _repository = Repository();
  }

  saveCategory(Category category) async{
    return await _repository!.insertData('categories',category.categoryMap());
  }

  readCategories() async{
    return await _repository?.readData('categories');
  }
}
```

Bu kod, bir kategori veritabanı tablosuyla etkileşim için yöntemler sağlayan bir CategoryService sınıfını tanımlar. CategoryService sınıfı, Havuz türünde özel bir \_repository alanına sahiptir ve \_repository alanını Repository sınıfının bir örneğiyle başlatan bir oluşturunucudur. CategoryService sınıfı, bir Category nesnesini parametre olarak alan ve \_repository alanının insertData yöntemini kullanarak veritabanındaki kategoriler tablosuna eş zamansız olarak bir satır ekleyen bir saveCategory yöntemine sahiptir. Category sınıfı, category.dart dosyasında tanımlanır ve kategoriler tablosundaki sütun adlarına karşılık gelen anahtarlarla kategori nesnesinin bir harita temsilini döndüren bir CategoryMap yöntemine sahiptir. CategoryService sınıfı ayrıca, \_repository alanının readData yöntemini kullanarak veritabanındaki kategoriler tablosundaki tüm satırları eş zamansız olarak okuyan bir readCategories yöntemine de sahiptir.