ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4537 PROJE SUNUMU

FLUTTER ile STİCKER APLİKASYONU

SERKAN CEVİZ

18290010

ENVER BAĞCI

01.2023

ÖZET

Ankara Üniversitesi Bilgisayar Mühendisliği Bölümü derslerinden BLM4537 kodlu İOS ile Mobil Uygulama Geliştirme dersi kapsamında aldığım “Flutter ile Sticker Aplikasyonu” konulu bitirme projesi için çalışmalar ve araştırmalar gerçekleştirmiş bulunmaktayım. Yapılan çalışmalar, araştırmalar ve örnekleri proje raporu kapsamında sizlere sunmak için hazırladım. Projemizde gerekli araştırmaları yaptıktan sonra uygulama aşamalarına geçtim. Gerekli üyelikler ve kurulumlardan sonra projenin mimarisini hazırlamaya başladım. Öncelikle gerekli tabloları ve bu tablolardaki Index’leri ayarladım. Daha sonrasında bu tablolara test için veriler atadım. Öncelikle Backend kısmı ile ilgilendim. Daha sonrasında Frontend tarafına başlayıp gerekli görünümleri ayarladım. Gerekli API’leri yazdıktan sonra uygulamam hazır oldu.

İÇİNDEKİLER

1.GİRİŞ

Flutter ile Sticker Aplikasyonu konulu projemin raporunda öncelikle Java Spring hakkında bilgi vermek istiyorum. Spring framework Java ile özdeşleşmiş bir frameworktür. Hatta o kadar özdeşleşmiştir ki bütün java iş ilanlarında dahi görünmektedir. API yazımında ve database ilişkilerinde usta olmakla birlikte altında birçok framework barındırmaktadır. Bu sebeplerden kaynaklı backend geliştirmede en çok tercih edilen frameworklerden birisidir.

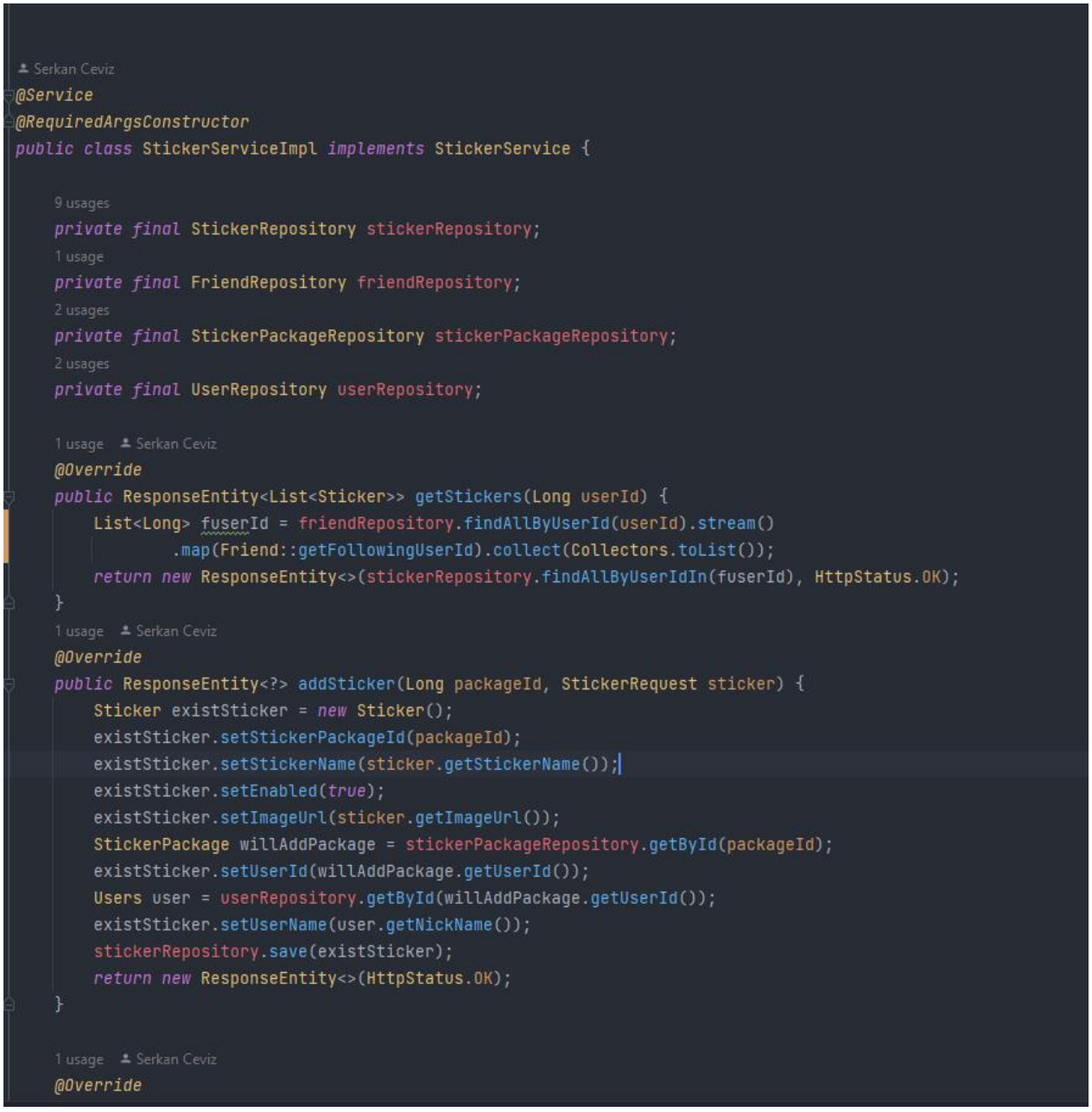
Sticker aplikasyonu gibi gerçek zamanlı bir proje yapılması için gerekli olan bu frameworkler sayesinde web sayfaları rahatlıkla yapılabilir olmuştur. Projenin hem çok etkileşimli olması hem de kullanıcının serbest olması amaçlanmıştır.

Bu bağlamda Sticker Aplikasyonuna direkt giriş yapmak yerine öncelikle alt başlıklara göz attık ve okuyucuyu bilgilendirerek sunumu tamamladım.

2. PROJE BACKEND KODLARI

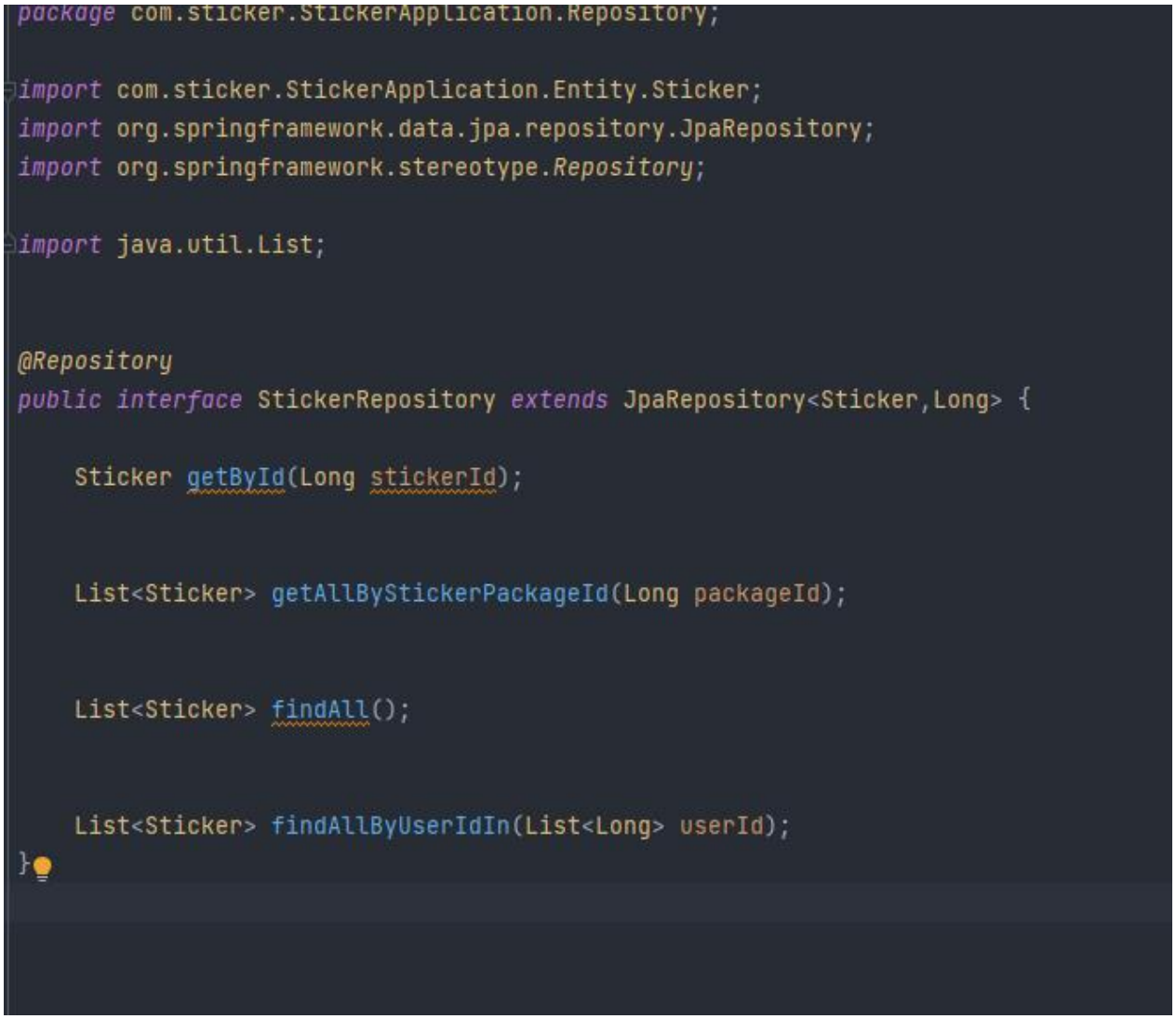
Öncelikle projemde gerekli işlemleri listeledim ve neler yapmam gerektiğine karar verdim. Daha sonrasında gerekli API ‘ler için mimarimi oluşturdum. Controller Classlarımı , Service Interface ve Classlarımı ayarladım. Database’e erişmek için Repository’mi ayarladım ve son olarak databasemde bulunacak tabloları şekillendirmek ve oluşturmak için entitylerimi ayarladım.

Şekilde görüldüğü üzere burası Sticker Controller kısmım. Bu kısımda anasayfadan tutun paketlerin stickerlarına erişimine kadar her şey düşünülüp gerekli API bağlantıları sağlanmış durumdadır. Controllerdan Service ‘e erişimi ise bir “Instance” yardımıyla yapıyorum.



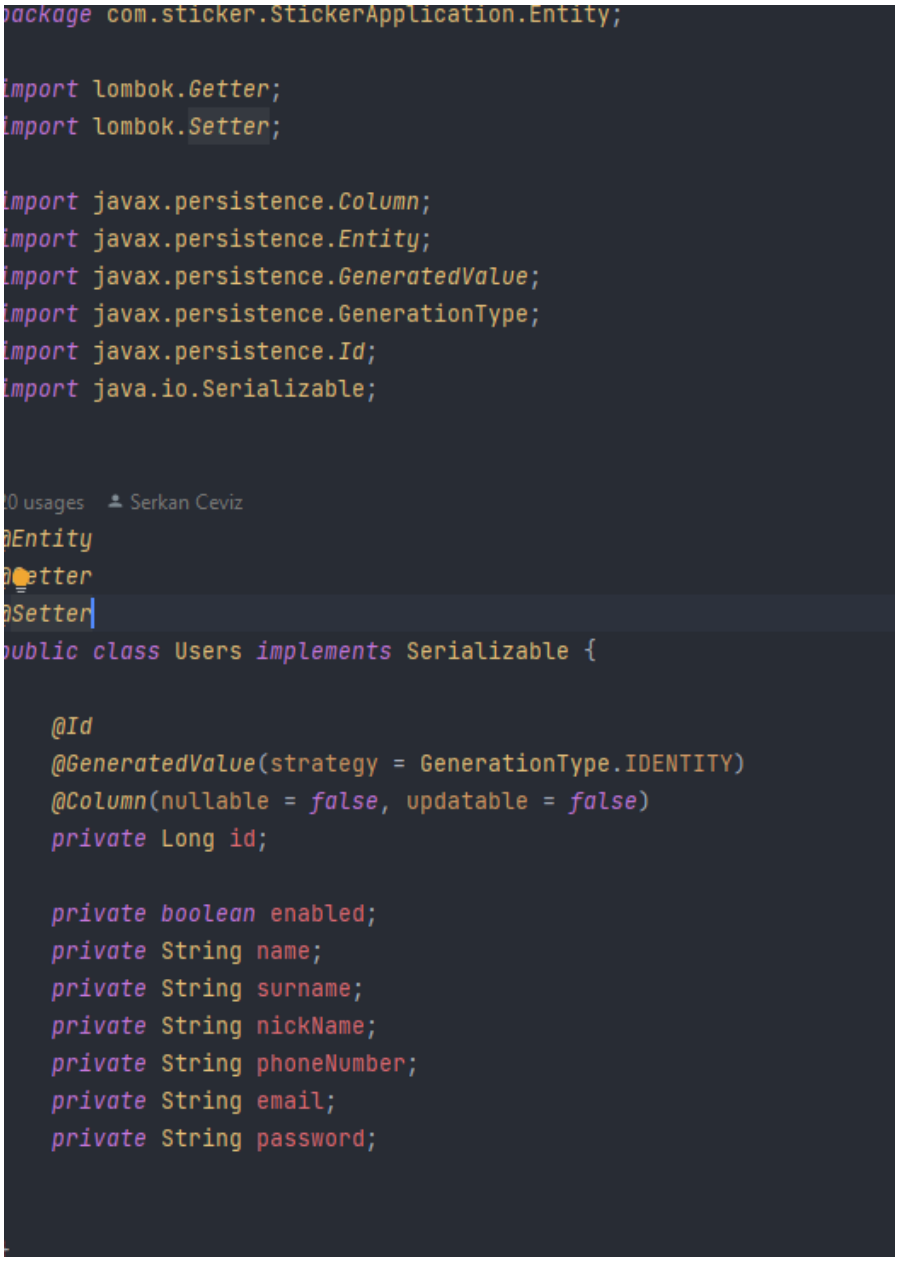
```
Serkan Ceviz
@Service
@RequiredArgsConstructor
public class StickerServiceImpl implements StickerService {

    9 usages
    private final StickerRepository stickerRepository;
    1 usage
    private final FriendRepository friendRepository;
    2 usages
    private final StickerPackageRepository stickerPackageRepository;
    2 usages
    private final UserRepository userRepository;

    1 usage  Serkan Ceviz
    @Override
    public ResponseEntity<List<Sticker>> getStickers(Long userId) {
        List<Long> fuserId = friendRepository.findAllByUserId(userId).stream()
                .map(Friend::getFollowingUserId).collect(Collectors.toList());
        return new ResponseEntity<>(stickerRepository.findAllByUserIdIn(fuserId), HttpStatus.OK);
    }
    1 usage  Serkan Ceviz
    @Override
    public ResponseEntity<?> addSticker(Long packageId, StickerRequest sticker) {
        Sticker existSticker = new Sticker();
        existSticker.setStickerPackageId(packageId);
        existSticker.setStickerName(sticker.getStickerName());
        existSticker.setEnabled(true);
        existSticker.setImageUrl(sticker.getImageUrl());
        StickerPackage willAddPackage = stickerPackageRepository.getById(packageId);
        existSticker.setUserId(willAddPackage.getUserId());
        Users user = userRepository.getById(willAddPackage.getUserId());
        existSticker.setUserName(user.getNickName());
        stickerRepository.save(existSticker);
        return new ResponseEntity<>(HttpStatus.OK);
    }

    1 usage  Serkan Ceviz
    @Override
```

Bu fotoğraf ise Bussines katmanında bulunan Sticker Service’e aittir. Bu kısımda gerekli iş kodlarını yazmış bulunmaktayım ve Repository’ye bağlanmaktayım.

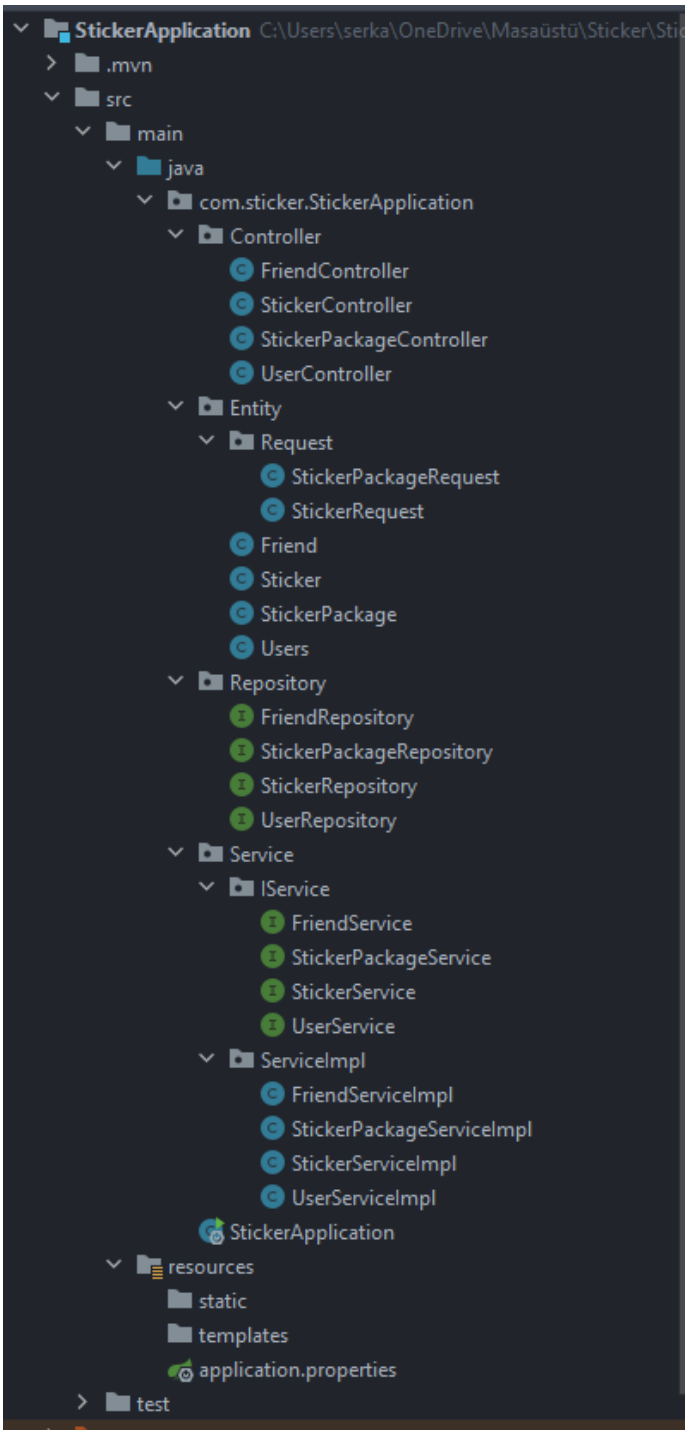


```
package com.sticker.StickerApplication.Repository;

import com.sticker.StickerApplication.Entity.Sticker;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;


@Repository
public interface StickerRepository extends JpaRepository<Sticker,Long> {

    Sticker getById(Long stickerId);


    List<Sticker> getAllByStickerPackageId(Long packageId);


    List<Sticker> findAll();


    List<Sticker> findAllByUserIdIn(List<Long> userId);
}
```

Bu fotoğrafta ise Repository'deki sorguları görmektesiniz. Bu sorgular yardımıyla Database'ye erişip gerekli bilgileri çekiyorum ve Controller'ıma kadar iletip gerekli frontend client'ına iletiyorum. Sticker API'sinin son katmanını oluşturmaktadır.



Şekilde Kullanıcılar için tasarlanmış bir Entity görmektesiniz. Burada oluşturulan variable'lar database'e başarılı bir şekilde aktarılmakta ve tablo oluşturulmaktadır.

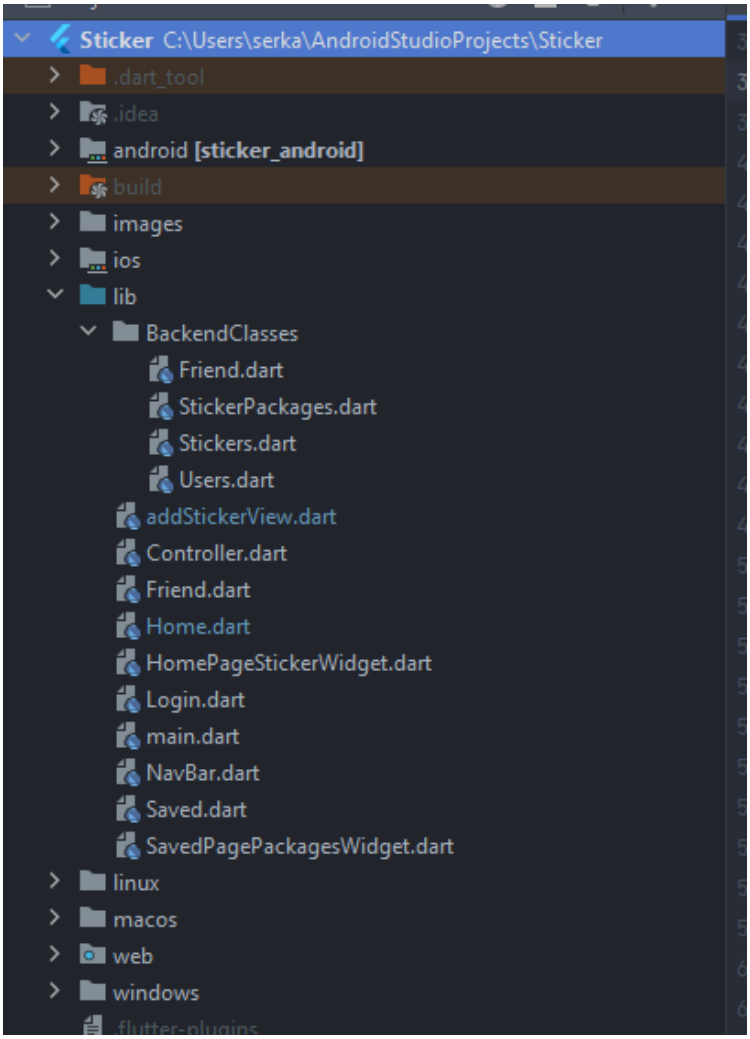


Son olarak Backend projemin son halini ve paketleri görmektesiniz. Mimariyi bu şekilde tasarlamış bulunmaktayım.

3.PROJE FLUTTER KODLARI

Öncelikle projenin görünüşüyle ilgili araştırmalar yapmaya başladım. Gerekli Statefull ve Stateless Widget'ları yazdıktan sonra her birini kod okunurluğu ve proje mimarisinin rahatlığı amacıyla ayrı bir Dart dosyası şeklinde ayarladım. Daha sonrasında backend projesiyle gerekli API bağlantıları sağlaması ve gerekli bilgileri sürekli olarak projede tutmak amacıyla gerekli paketler indirdim. Bunlardan bazıları GetX adındaki popüler bir uygulama. AnimatedBottomNavigationBar paketini de yükleyip Nav Bar kısmını ayarladım. Gerekli Routing'leri de ayarladıktan sonra projem hazır oldu.

Öncelikle projenin mimarisi şu şekilde :



Backend classlar API ler aracılığıyla JSON formatında veri göndermek ve çekmek amacıyla yazılmış classlardır.

```dart
import 'dart:convert';

StickerPackages stickerPackagesFromJson(String str) => StickerPackages.fromJson(json.decode(str));

String stickerPackagesToJson(StickerPackages data) => json.encode(data.toJson());

class StickerPackages {
  StickerPackages({
    this.id,
    this.userId,
    this.enabled,
    this.name,
    this.imageUrl,
  });

  int? id;
  int? userId;
  bool? enabled;
  String? name;
  String? imageUrl;

  factory StickerPackages.fromJson(Map<String, dynamic> json) => StickerPackages(
    id: json["id"],
    userId: json["userId"],
    enabled: json["enabled"],
    name: json["name"],
    imageUrl: json["imageUrl"],
  );

  Map<String, dynamic> toJson() => {
    "id": id,
    "userId": userId,
    "enabled": enabled,
    "name": name,
    "imageUrl": imageUrl,
  };
}
```

Görüldüğü gibi Json formatına çevirme ve decode – encode etme gibi fonksiyonlar da bulunmakta.

```dart
final controller = Get.put(Controller());

late Future<List<Stickers>> _stickerList;

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

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {

  bool isOk = false;
  String userId = controller.getUserId().toString();
  String demoUrl = "http://10.0.2.2:8080/stickers/home/";

  Future<List<Stickers>> getStickers() async {

    String getStickersUrl=demoUrl+userId;

    final url = Uri.parse(getStickersUrl);
    final response = await http.get(url);

    List<Stickers> stickerList = [];

    if(response.statusCode==200){
      var stiList = jsonDecode(response.body);

      if(stiList is List){
        stickerList = stiList.map((e) => Stickers.fromJson(e)).toList();
      }else{
        return [];
      }
      isOk=true;
    }
    return stickerList;
  }
```

Anasayfa ekranının kodları bu şekildedir. Görüldüğü üzere 10.0.2.2 yani Localhost'a istek atmaktadır. Gelen cevabı bir listede tutup o listedeki elemanları Sticker olarak ekranda göstermektedir.

```dart
import 'package:sticker/Controller.dart';
import 'Home.dart';
import 'package:http/http.dart' as http;
import 'package:animated_snack_bar/animated_snack_bar.dart';

final LocalStorage storage = LocalStorage('key');


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

  @override
  State<MyApp> createState() => _MyAppState();
}


class _MyAppState extends State<MyApp> {
  String demoUrl = "http://10.0.2.2:8080/users/findByNickname/";
  bool isOk = false;
  final controller = Get.put(Controller());


  Future findUser() async {
    final url = Uri.parse(demoUrl);
    print("SAAAAA");
    final response = await http.get(url);

    if(response.statusCode==200){

      var result = userFromJson(response.body);
      print(result.name);
      isOk=true;

      storage.setItem('id', result.id);
      controller.setUserId(result.id);

    }

  }
```

Login sayfasının kodları bu şekildedir. Kullanıcının girdiği kullanıcı adı ve parolayı backend'e yollayıp gelen cevaba göre alertify yardımıyla ekrana basıp kullanıcıya bildirmektedir.

```
}

class _SavedState extends State<Saved> {

  String userId = controller.getUserId().toString();
  String demoUrl = "http://10.0.2.2:8080/stickerPackages/";
  Future<List<StickerPackages>> getPackages() async {

    String getStickersUrl=demoUrl+userId;

    final url = Uri.parse(getStickersUrl);
    final response = await http.get(url);

    List<StickerPackages> stickerList = [];

    if(response.statusCode==200){
      var stiList = jsonDecode(response.body);

      if(stiList is List){
        stickerList = stiList.map((e) => StickerPackages.fromJson(e)).toList();
      }else{
        return [];
      }

    }
    return stickerList;
  }

  @override
  void initState() {
    _stickerList = getPackages();
    super.initState();
  }
```

Sticker Paketleri için oluşturduğum Kaydedilenler dosyasında ise Backend'e kullanıcının Id'siyle birlikte istek atmaktadır. Gelen cevabı listede tutup ekrana bastırmaktadır. Bu fonksiyon görüldüğü üzere "initState" kısmında çağırılmaktadır. Yani bu sayfaya giriş yapılır yapılmaz bir butona basılmaksızın istek yapılmaktadır.

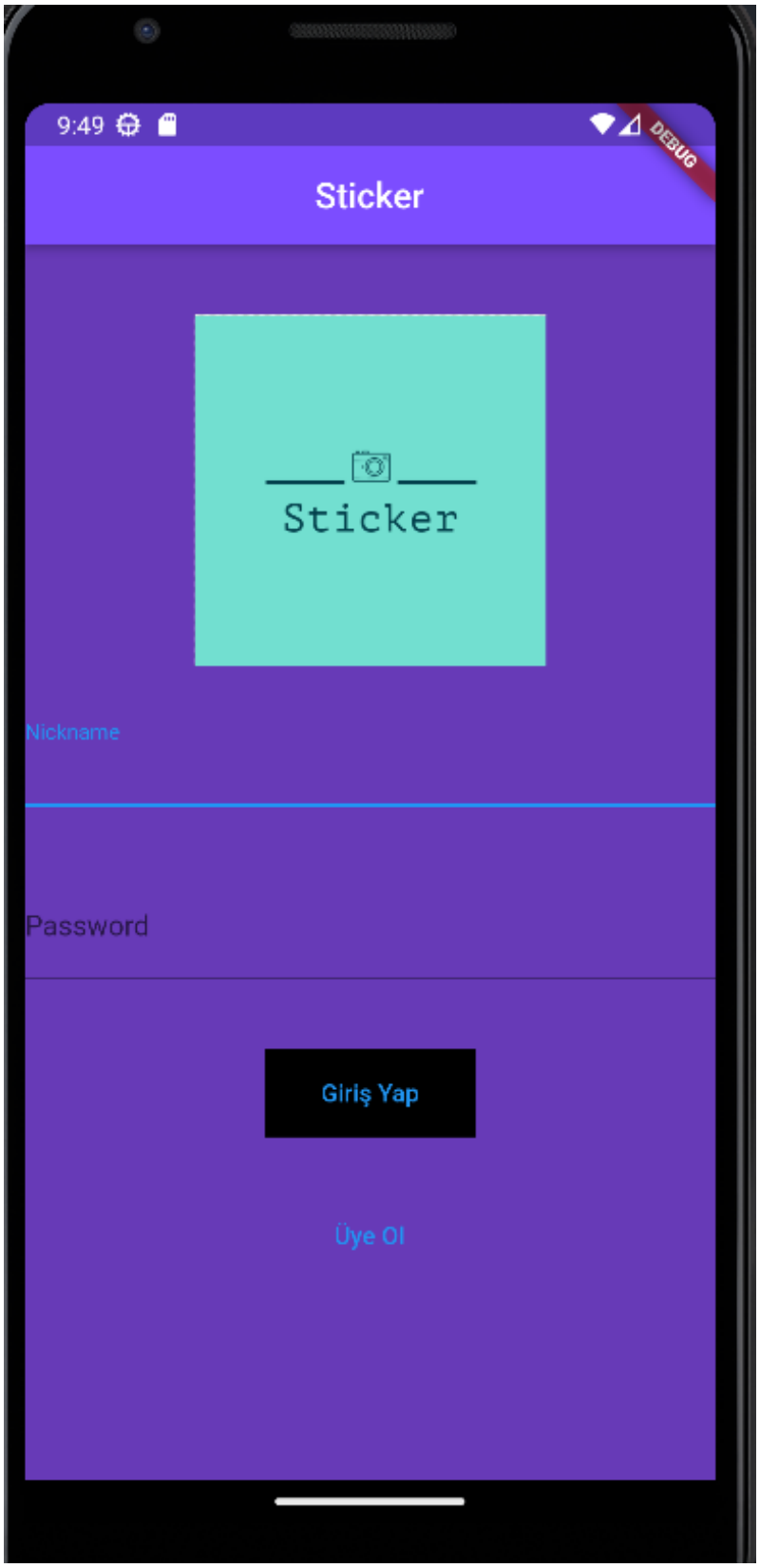
4.STİCKER APLİKASYONU

Sticker aplikasyonu nedir diye soracak olursak; İnsanların kendi yaptıkları stickerları paylaştığı, birbirleriyle takipleşebildiği ve bu stickerları kendi paketlerine ekleyebildikleri bir platform. Bir nevi Instagram'ın sticker versiyonu diyebiliriz. Bu projedeki amaç nedir diye soracak olursak bir uygulama vesilesiyle 20-30 adet stickerın olduğu bir paketi indirebiliyorsunuz. Fakat bu paket içerisindeki stickerların sadece birkaçı gerçekten güzel olabilir ve siz kalanları da indirmek zorundasınız. Bu paketler ise bir uygulamanın sağlamış olduğu belli başlı paketler şeklindedir. Bu aplikasyonla birlikte insanlar artık görüp beğendikleri stickerları tek tek paketlerine aktarabilecek ve diğer kullanıcılarla aktif bir şekilde etkileşimde kalacak.

4.1 Login Sayfası

Login sayfamda öncelikle flutter stateless widget kullanarak genel hattı oluşturdum.

Flutter ile gerekli kurulumları ve istekleri hazırladım. Kullanıcı girişi başarıyla sonuçlanırsa "Alertify" yardımıyla bir pop-up şeklinde "Giriş Başarılı" yazısı çıkıyor. Eğer giriş başarısız olursa kırmızı renkte bir pop-up ile "Giriş Yapılamıyor" yazısı çıkıyor. Son olarak kullanıcı girişi doğruysa Flutter-routing sayesinde Anasayfaya yönlendirdim.



Şekil 4.1.1 Login Sayfası

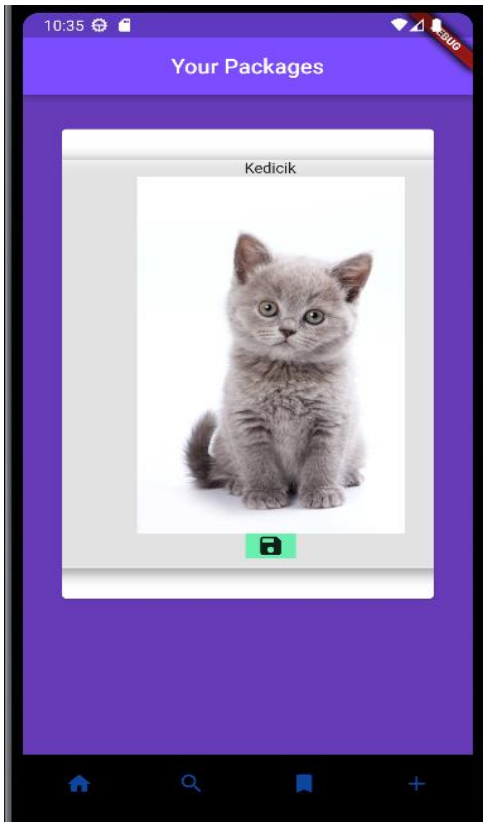
4.2 Anasayfa

Başarılı giriş sonrası kullanıcının takip ettiği kişilerin paylaştığı stickerları getiren Anasayfa'ya erişmiş oldu. Bu kısımda kullanıcıların paylaştıkları stickerları görüntüleyebilen kullanıcımız dilediğini kendi paketine ekleme hakkına sahip oluyor.

Şekil 4.2.1 Anasayfa

4.3 Profil Sayfası

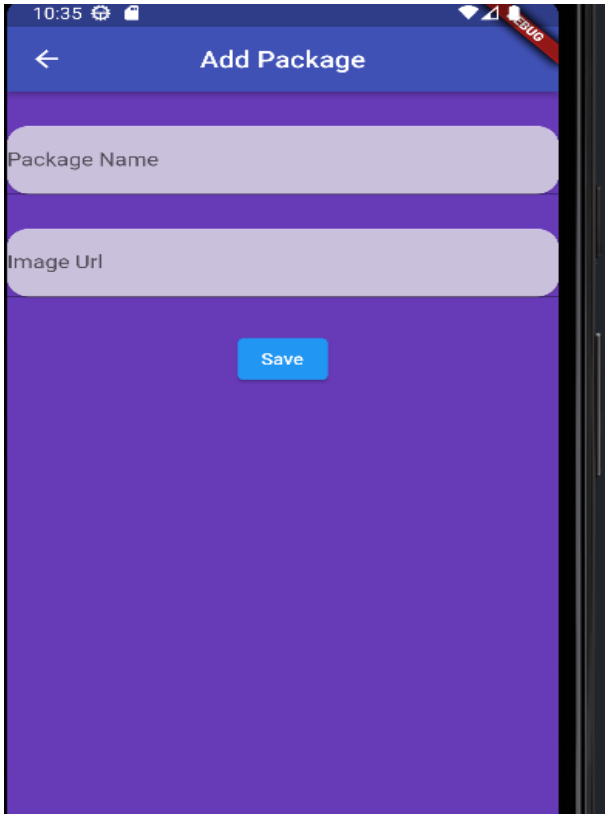
Bu kısımda kullanıcı kendi paketlerini ve bu paketlerin içindeki stickerları görüntüleyebiliyor. Genel anlamda hangi paketlerinin olduğunu ve bu paketlerin içerisindeki stickerların neler olduğuna dair fikir veren bu kısımda ilerleyen safhalarda Whatsapp'a entegrasyon düşünmekteyim.



Şekil 4.3.1 Profil sayfası sticker paketleri

4.4 Paket ve Sticker Ekleme Sayfası

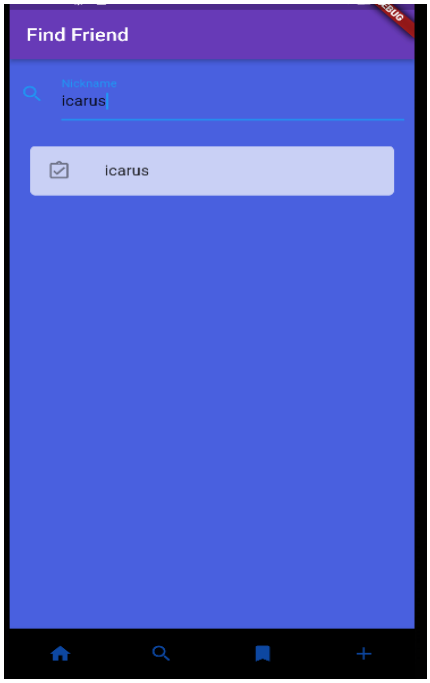
Bu sayfada ise kullanıcı dilerse paket ekleyebilir ya da sticker ekleyebilir. Paketleri, başka kullanıcılar tarafından görüntülenemez olurken eklediği sticker anında onu takip eden kullanıcılara gösterilebilir hale gelir. Sticker eklerken ise bir paket seçmek zorundadır. Yani yeni başlayan bir kullanıcının öncelikle paket oluşturması gerekmektedir.



Şekil 4.4.1 Sticker ve Paket ekleme sayfası

4.5 Arkadaş Ekleme Sayfası

Bu sayfada kullanıcı istediği kişiyi takip edebilir. Fakat gönderilen istek sonrası diğer kişi sizi takip etme durumuna girmez. Yalnızca takip isteğini atan kişi takip etmiş olur. Bu sayfa sayesinde aslında anasayfamız daha işlevsel hale geliyor. Kullanıcıların ürettikleri stickerları görüntüleyebiliyor hale geliyoruz. Eğer takip atacağımız kişiyi zaten takip ediyorsak bizi bilgilendiren bir ikon görünür hale geliyor. Eğer etmiyorsak ona göre farklı bir ikon görünür hale geliyor.



Şekil 4.5.1 Arkadaş ekleme sayfası ve takip edildiğini bildiren ikon