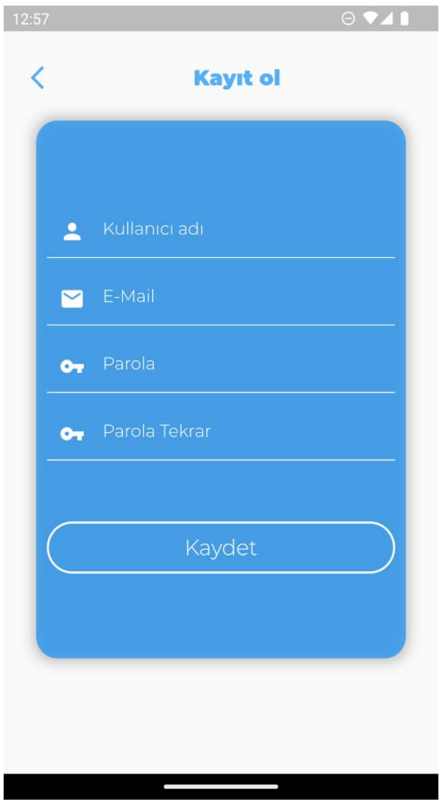
2.3 Uygulamanın FrontEnd'inin Yazılması

2.3.1Giriş ve Kaydol Sayfası



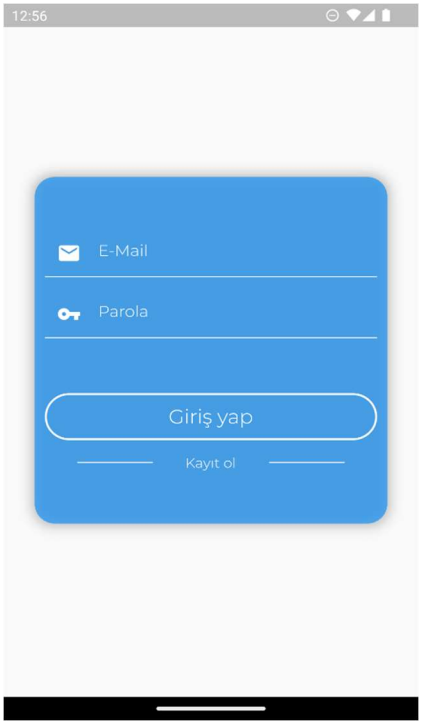
Kayıt ekranını yaparken öncelikle bir Container oluşturdum. Container'in içerisine kullanıcıdan girmesini istediğim bilgileri belirleyip daha sonrasında Text Field ile Input decoration kullanarak bir column icerisine verileri hazırladım.

Kaydet butonuna tıklandığında bizi Giriş yap erkanına yönlendirmesini istedim bu yüzden

```
MaterialPageRoute(
builder: (context) => LoginPage()));
```

kullandım.

Şekil 1 Kayıt ol sayfası



Giriş Sayfasında Kaydol sayfasında kullandığım temayı devam ettirdim. Bu sefer kullanıcının Email ve Parola girmesi yeterli olduğu için sadece bu iki veriyi istedim.

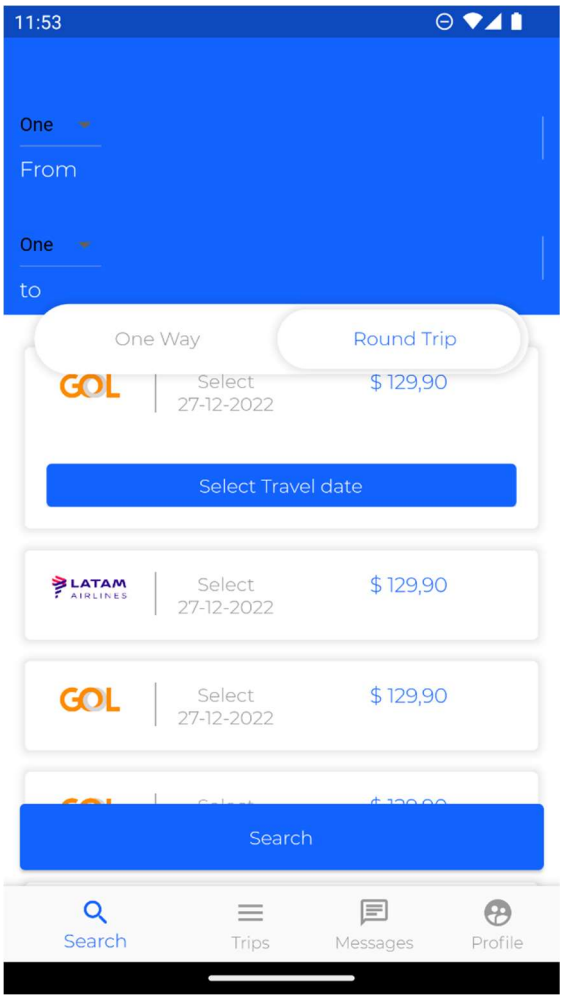
Giriş yap butonuna basıldığında; eğer e mail ve parola veri tabanında bulunuyorsa bizi Ana sayfaya yönlendirmesini sağlamak istedim. Bunu da Firebase kullanarak gerçekleştirdim.

Eğer kullanıcı bulunmuyorsa uygulama bir hata döndürüyor.

Sayfanın sonuna eğer kullanıcı kayıtlı değilse diye Kayıt ol ekranına yönlendiren bir buton koydum.

Şekil 2 Giriş Sayfası

2.2.2 Ana Sayfa

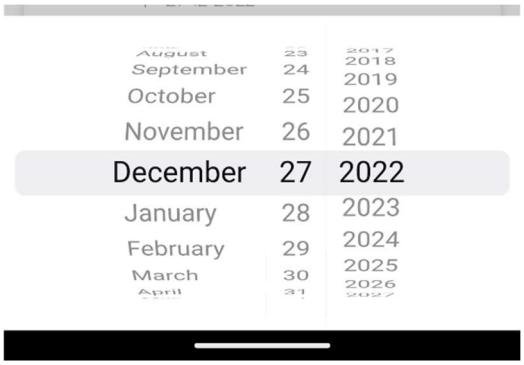


Şekil3 Ana Sayfa

Ana Sayfada bizi iki tane dropdown karşılıyor. Yola çıkılacak konum ve gidilecek yeri seçtiğimizde bize seferleri listeliyor. Gidiş ve ya Gidiş- Dönüş sekmesinden seferleri görüntülüyorsunuz.

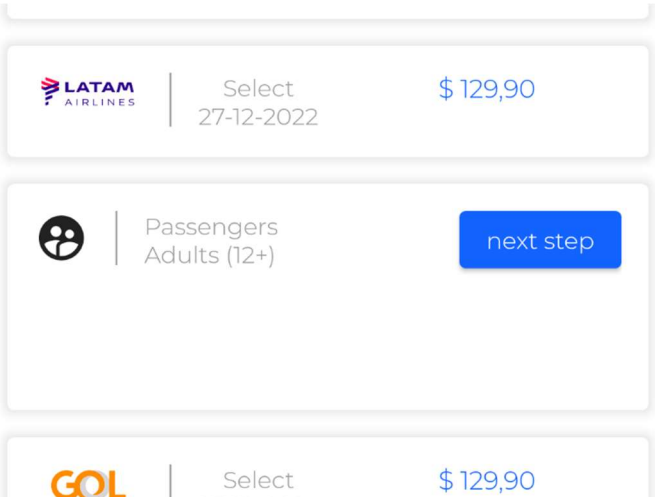
Seferler de önce şirketin logosu daha sonra gidilecek tarih ve fiyatı listeleniyor.

Select Travel date butonuna bastığınızda gün ay yıl seçebileceğiniz bir pop up açılıyor.



Şekil 4 Tarih Seçme Widgeti

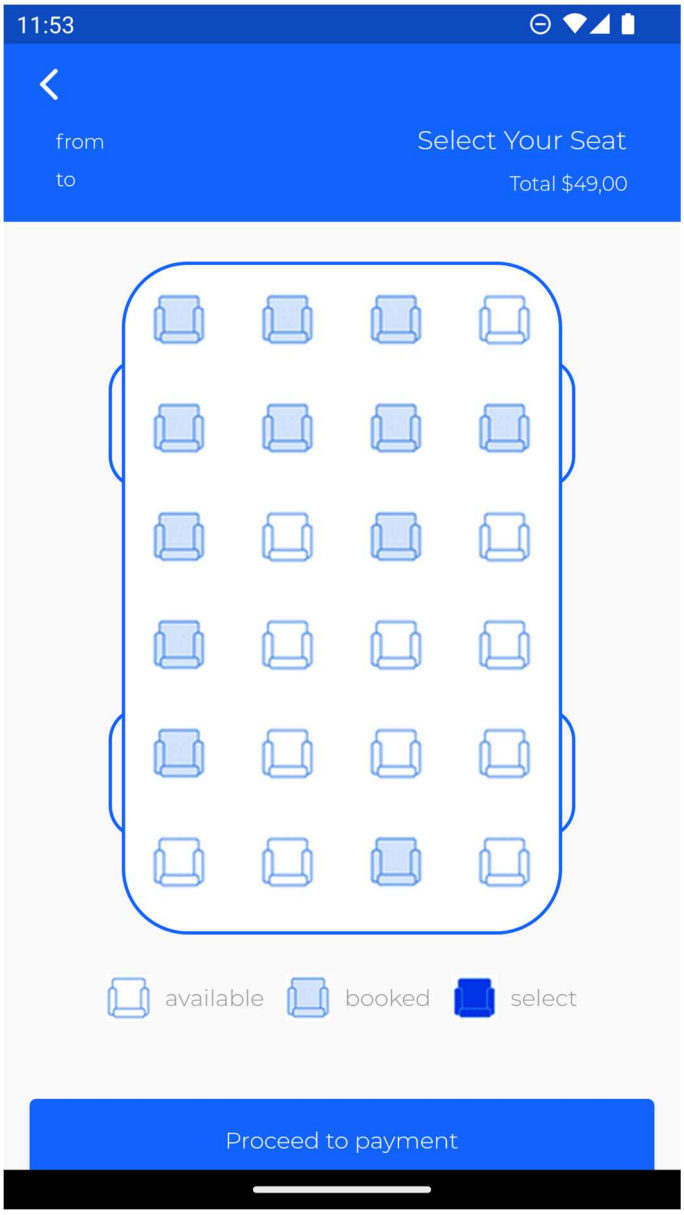
Secim yapmak için çıkan seferlerden seçtiğinizi yana kaydırarak şu ekrana geliyorsunuz.



Şekil 5 Sefer Bilgileri

İlk olarak yolcunun yaşı hakkında bir uyarı görülüyor. Eğer 12 yasından küçük ise tek başına yolculuk edemeyeceğine dair. Next step butonuna tıklandığında ise otobüsün içini görebileceğiniz ve koltuk seçebileceğiniz bir sayfaya yönlendiriyor.

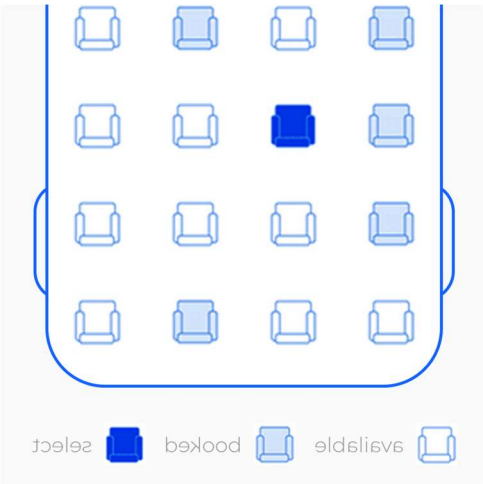
2.2.3 Otobüs Koltuk Seçme Sayfası



Şekil 6 Koltuk Sayfasi

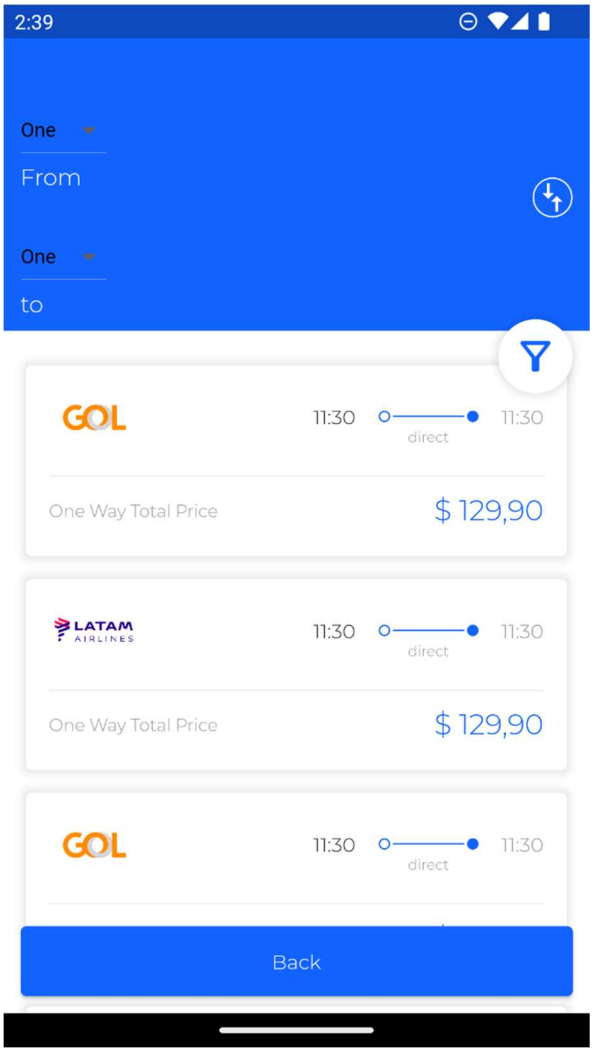
Koltuk seçme ekranına geldiğimiz de oncelikle ust kosede gidilen yolculuğun bilgileri belirtiliyor. Alt tarafta ise bir otobüs görüntüsü karşılıyor . 3 çeşit koltuk rengi ile birlikte. Beyaz renkteki koltuklar seçim yapılmamış koltuklar, açık mavi olan koltuklar daha önceden alınmış, koyu mavi olan koltuk ise sizin seçtiğiniz koltuğu belirtir.

Koltuk seçildikten sonra ödeme kısmına geçmek için bir buton bulunuyor.



Şekil 7 Seçili Koltuğun Gösterimi

2.2.4 Arama Sayfası



Şekil 8 Arama Sayfası

Arama sayfasının Ana sayfadan sadece bir kac farkı var. Bunlardan ilki gidiş dönüş rotaları arasında tek tuşa basarak değişiklik yapılma imkanı sunması. Bu şekilde gidiş dönüş rotalarını değiştirme kolaylıgı sunar.

Şekil 9 Şehir Değiştirme 1

Şekil 10 Şehir Değiştirme 2

Seferlerin gösterildiği kısımda ise otobüs firmasının logosu ve seferin yapılacak saatleri yer alır. Sefer fiyatı ve bilet türü ekranda belirtilir.

Gidiş dönüş rotasını değiştirmek için bir değişken oluşturdum. Değişkeni false olarak tanılmadım. Butona basıldığında false ise true true ise false olmasını sağlayan bir if else yapısı kurdum. Oluşturduğum değişken false iken olan değerler ilk girilen rota bilgisi oldu. True olduğunda da bu iki değerin değiştirilmiş hali olan rota kabul edilen rota haline geldi.

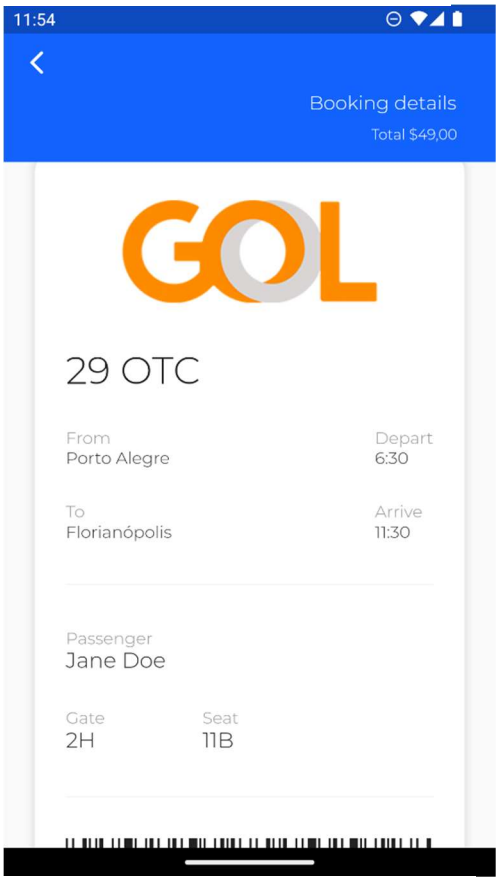
Şekil 11 Bilet Geçmişi Sayfası

2.2.5 Bilet Geçmişi sayfası

Bilet geçmişi sayfasında kullanıncınınn daha önce aldığı biletler gorunur. Widgetlar seklinde bilet bilgileri ekrana verilir. Gidilen fırmanın adı, kişinin oturduğu koltuk , nerden nereye gittiği, kalkış ve varış saati bulunur.

Gesture dedector widgeti kullanarak bilet ekranının harekete duyarlı hale getirilmesi saglanmıstır. Üzerine basıldığında sizi yeni bir sayfaya götürmesi ayarlandı. Bu yeni sayfada bilet detaytlarını daha acık bir şekilde görülmesi sağlandı.

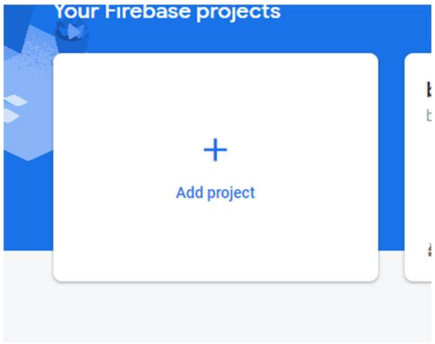
```
GestureDetector(
                onTap: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
                              TripTicketPage(),
                    ),
                  );
```



Detay sayfasında Bilet hakkında detaylı bilgiler verilmiştir.

Şekil 12 Bilet Detayı

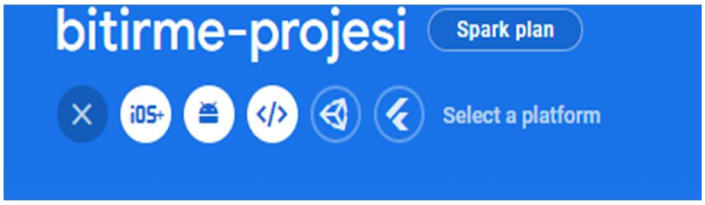
3.2 Firebase'ı Flutter Uygulamasına Kurma



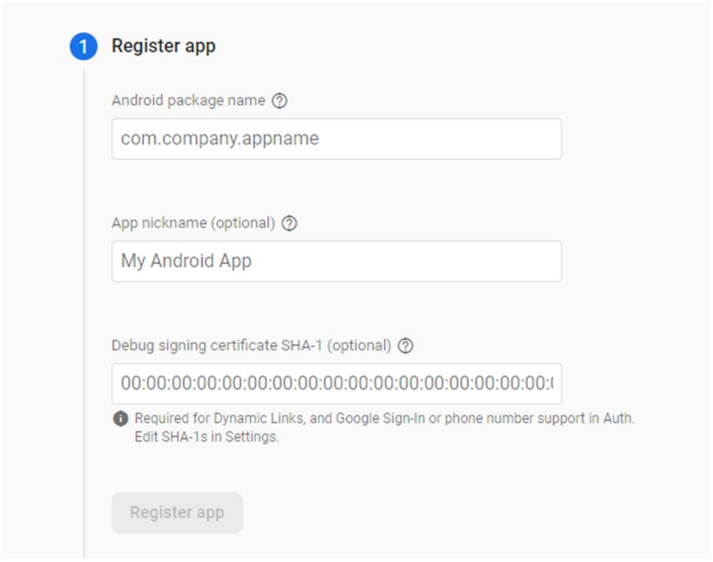
Şekil 13 Firebase Proje Ekleme Butonu

İlk olarak Firebase Console web sayfasına giriş yapıyoruz daha sonra console ekranında bulunan add Project butonuna basıyoruz. Bizden bir proje ismi istiyor proje ismini girdikten sonra bizden Google analytics ın verileri kullanıp kullanmamasına izin vericegimiz bir sayfa çıkıyor o sayfayı geçtiğimizde sizin için bir proje oluşturuyor.

Proje sayfasına girdiğimizde girişte bizi su şekilde app ekleme kısmı karşılıyor



Uygulamayı hagi platformda yapıyorsanız o icona bastığınızda sizi bir sayfaya yönlendiriyor.



Şekil 14 Firebase Android Proje Ekleme Sayfası

Android paket adının uygulama seviyesindeki build.gradle dosyasında bulunur. Sonraki adımda yapılandırma dosyasını indirip Android studio'da açıp proje görünümüne geçin. Yapılandırma dosyasını Android uygulama modulundeki kok dizinine taşıyın.

Eklenen yapılandırma dosyasını proje/android/build.gradle'nin içinde olduğunu belirtmek için dependencies bolumune şu kodu ekliiyoruz:

```
dependencies {
    classpath 'com.android.tools.build:gradle:7.1.2'
    classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    classpath 'com.google.gms:google-services:4.3.13'
  }
```

Bir sonraki adımda android/app/build.gradle dosyasının içerisine şu kodu eklememiz yeterli .

```
apply plugin: 'com.google.gms.google-services'
```

Son olarak main dosyanızı;

```
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
 await Firebase.initializeApp();
 runApp(MyApp ());
}
```

Bu şekilde değiştirmeniz gerekmektedir.

Bu işlemlerden sonra android uygulamanıza firebase eklenmiş olucaktır.

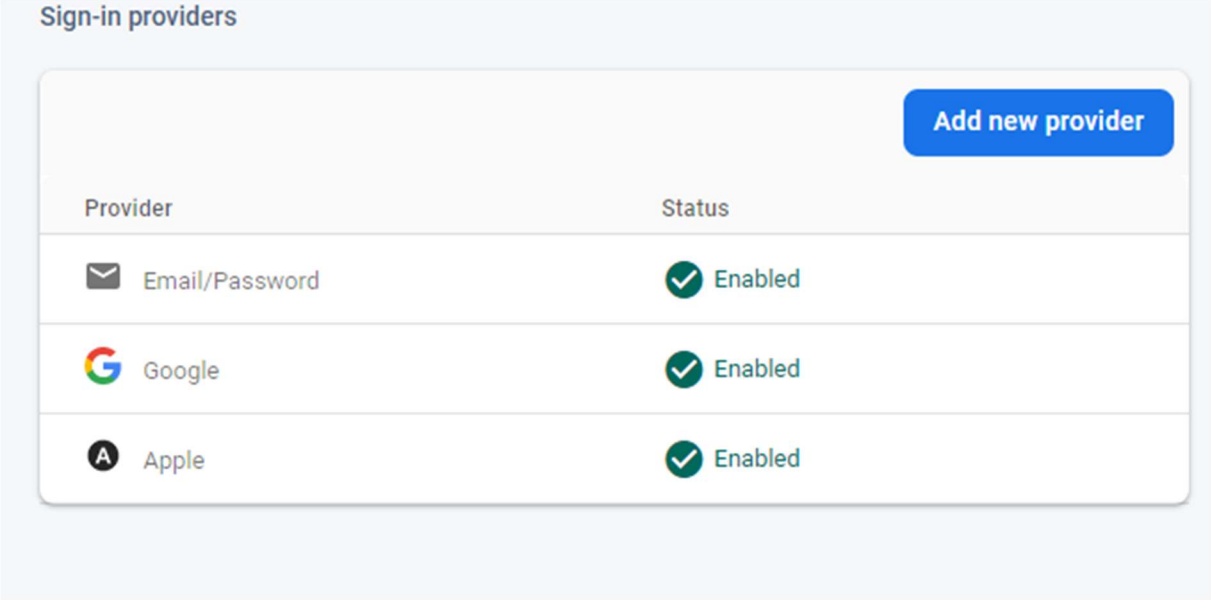
3.3 Firebase ile Kimlik Doğrulama

Uygulamaya kullanıcı eklemek için firebase kimlik doğrulama özelliğini kullandım. Firebase kimlik doğrulama kullanarak, e posta ve şifre, telefon numarası ve Twitter ve Facebook gibi popüler birleşik kimlik sağlayıcıları gibi oturum açma yöntemlerini ortaya çıkarır. Firebase kullanmanın faydaları ise:

- Kullanımı ücretsizdir.
- Kullanımı ve yönetimi kolaydır.
- Popüler kimlik doğrulama standartları kullanır.
- Diğer firebase hizmetleri ile entegre olur.
- Geliştirme zamanı kısadır.
- İşlemci tarafından desteklenen UI kitaplıklarını kullanıma sunar.

3.3.1 Firebase Kimlik Doğrulamayı Uygulamaya Entegre Etmek

İlk olarak firebase console a giriş yapıyoruz Build kısmından Authentication seçeneğine tıklıyoruz. Bizden bir oturum açma sağlayıcısı seçmemizi istiyor. Uygulamada kullanmak istediklerinizi seçiyorsunuz.



Proje de bizim işimize yarayacak olan paketleri de pubspec.yaml dosyasına ekliyoruz

```
firebase_auth: ^4.2.3
firebase_core: ^2.4.0
```

Projeye öncelikle servislerimizi yazdığımız bir dosya ekliyoruz. Dosya da giriş yapmak, çıkış yapmak ve kayıt ol fonksiyonları bulunuyor.

```dart
class AuthService {
  final FirebaseAuth _auth = FirebaseAuth.instance;
  final FirebaseFirestore _firestore = FirebaseFirestore.instance;

  //giriş yap fonksiyonu
  Future<User?> signIn(String email, String password) async {
    var user = await _auth.signInWithEmailAndPassword(
        email: email, password: password);
    return user.user;
  }

  //çıkış yap fonksiyonu
  signOut() async {
    return await _auth.signOut();
  }

  //kayıt ol fonksiyonu
  Future<User?> createPerson(String name, String email, String password) async {
    var user = await _auth.createUserWithEmailAndPassword(
        email: email, password: password);

    await _firestore
        .collection("Person")
        .doc(user.user!.uid)
        .set({'userName': name, 'email': email});

    return user.user;
  }
}
```

Giriş yap ve Kayıt ol sayfalarımıza AuthService classını ekliyoruz. Kullanıcıdan girilen veri kadar TextEditingController ve AuthService tipinde bir değişken tanımlıyoruz.

```dart
class _RegisterPageState extends State<RegisterPage> {
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _passwordAgainController =
      TextEditingController();

  AuthService _authService = AuthService();
```

Bu kısmı iki dosya için de aynı şekilde yapıyoruz.

Kayıt ol dosyasında yeni bir kullanıcı eklediğimiz için createPerson fonskiyonunu kullanıyoruz.

```
onTap: () {
  _authService
      .createPerson(
          _nameController.text,
          _emailController.text,
          _passwordController.text)
      .then((value) {
    return Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => LoginPage())); // MaterialPageRoute
  });
},
```

Giriş yap sayfasında giriş yapıldığı için signIn fonksiyonu kullanılır.

```
onTap: () {
  _authService
      .signIn(
          _emailController.text, _passwordController.text)
      .then((value) {
    return Navigator.push(
        context,
        MaterialPageRoute(
            builder: (context) => SearchPage())); // MaterialPageRoute
  });
},
child: Container(
```