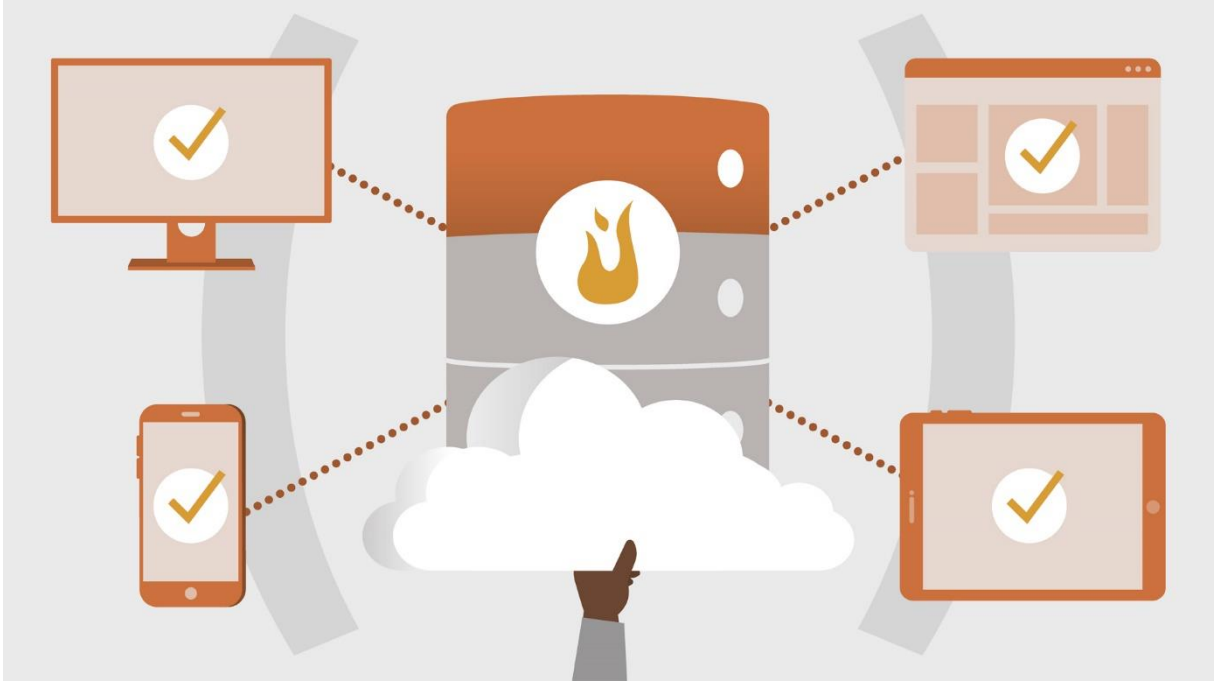


UçAndroid, Makine Öğrenmesi ile Otonom IHA sistemi üretimi ve Proje Tabanlı Üretim Eğitim Modüllerinin Geliştirilmesi

Bulut Teknolojileri - Firebase



Abidinpaşa Mesleki ve Teknik Anadolu Lisesi

2021

UçAndroid, Makine Öğrenmesi ile Otonom IHA sistemi üretimi ve Proje Tabanlı Üretim Eğitim Modüllerinin Geliştirilmesi

Bu proje T.C. Ankara Kalkınma Ajansı tarafından finanse edilmektedir.

TR51/19/ÜTMEGP/0052

İleri Teknolojili Ürün Ticarileştirme ve İleri Teknoloji Alanında Mesleki Eğitimin
Geliştirilmesi Mali Destek Programı

2019

İçindekiler

Giriş.....	4
BULUT TEKNOLOJİSİ NEDİR?.....	4
BULUT TEKNOLOJİSİ NASIL ÇALIŞIR?	5
BULUT BİLİŞİM TÜRLERİ.....	6
BULUT TEKNOLOJİSİNİN KULLANILDIĞI ALANLAR	7
BULUT TEKNOLOJİSİNİN FAYDALARI	8
FIREBASE.....	9
Firebase Hizmetleri.....	9
Neden kullanmalıyız?	10
FIREBASE PROJESİ OLUŞTURMA.....	11
1. Projemize isim verme	12
2. Google Analitik yapılandırma	12
3. Projeyi Oluşturma.....	13
FIREBASE AUTHENTICATION (KİMLİK DOĞRULAMA)	14
FIREBASE AUTHENTICATION AYARLARI.....	14
ANDROID STUDIO PROJESİNE FIREBASE PROJESİNİ BAĞLAMA.....	18
UÇANDROİD UYGULAMASINDA FIREBASE BULUT YAPISI	25
FIREBASE VERİTABANI İŞLEMLERİ.....	28
Tablo Oluşturma	30
Kayıt Eklme	30
Uçandroid Projesi için İhtiyaç Duyulan Tabloların Oluşturulması	31
Tuyeler Tablosu	31
Yapılar Tablosu	32
Uyeler Tablosu.....	32
Gorevler Tablosu	33
ANDROID STUDIO İLE FIREBASE REALTIME(GERÇEK ZAMANLI) VERİTABANI BAĞLANTISI OLUŞTURMA.....	34
Uçandroid Android Programı Veri Tabanı işlemleri.....	41
FIREBASE STORAGE DOSYA SAKLAMA SERVİSİ.....	44
Uçandroid Android Programı Dosya işlemleri	45
Kaynaklar	48

Giriş

Bu modülde bulut teknolojisi nedir, türleri nelerdir ve nasıl tasarlandıkları hakkında bilgiler verilecektir.

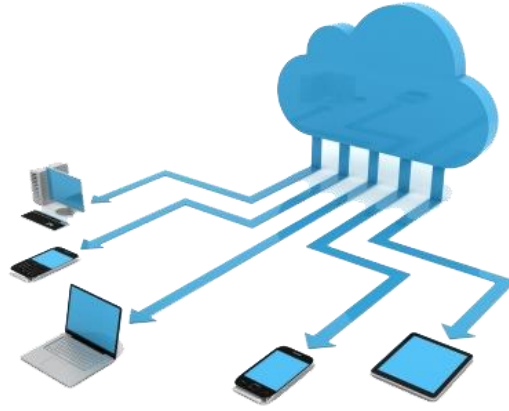
Hedef Kitle : Modülün hedef kitlesi mobil programcılığı hakkında çalışma yapmak isteyen tüm bireyleri kapsamaktadır.

Süre : Modül eğitim süresi 10-14 saattir.

Ön koşul : Modül lise seviyesi üzerinde uygulanacaktır.

Amaç : Bu modülle öğrenci için gerekli ortam sağlandığında; bulut teknolojilerini tanıyan, mobil uygulama ve bulut yapısı arasındaki bağlantıları gerçekleştirebilen, çok kullanıcı mobil ve masaüstü sistemler oluşturmayı gerçekleştirebilen bireylerde beceriler oluşturmaktır.

BULUT TEKNOLOJİSİ NEDİR?

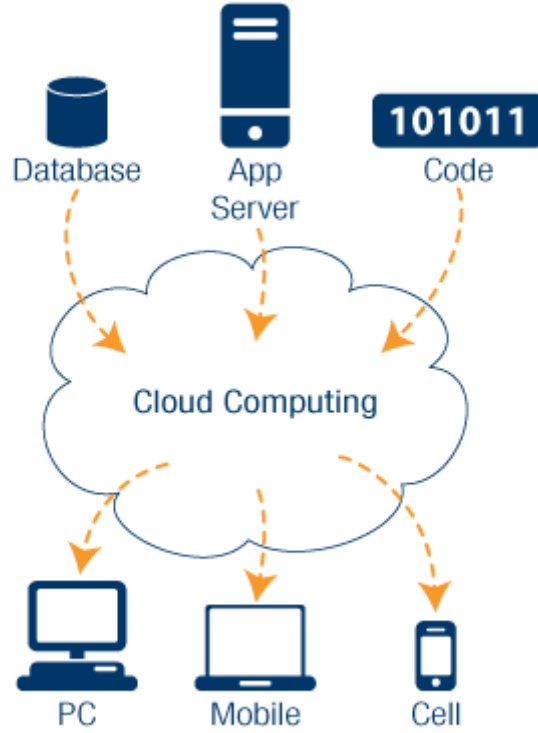


Bulut bilişim teknolojisi, kullanıcıların internete bağlı cihazları (bilgisayarlar, akıllı telefonlar ve tabletler) aracılığıyla depolamaya, dosyalara, yazılıma ve sunuculara erişimini sağlar. Bulut bilişim sağlayıcıları, verileri son kullanıcılardan ayrı bir konumda depolar ve işler. Güvenlik için yüksek şifreleme algoritmaları kullanır. Tüm bunların gerçekleşmesinde en önemli etken internet ve bağlanabilirlik teknolojilerinin gelişmesi ve hızlanmasıdır.

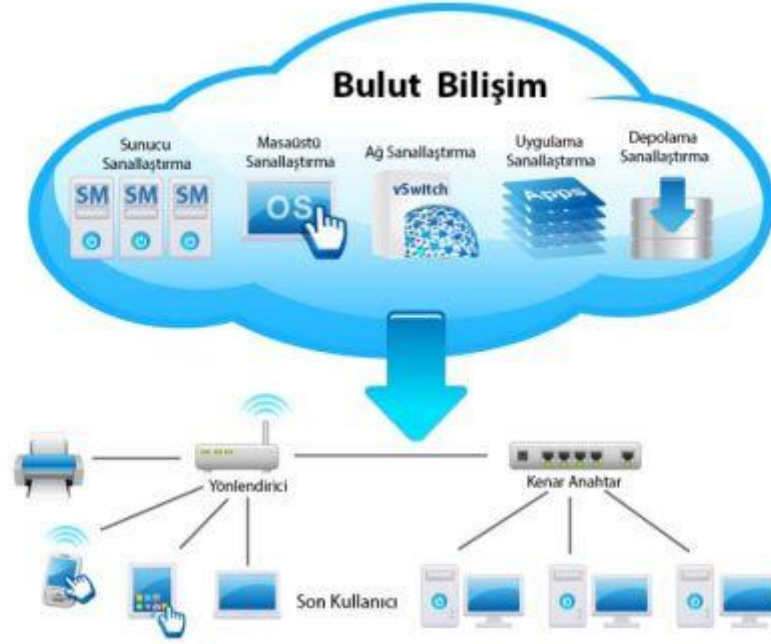
Temel olarak, bulut bilişim, verileri ve programları bir bilgisayardaki sabit sürücü yerine internet üzerinden depolama ve bunlara erişme yeteneğine sahip olmak anlamına gelmektedir. En basit bir ifadeyle, bulut internette **sanal bir depolama alanıdır**.

BULUT TEKNOLOJİSİ NASIL ÇALIŞIR?

Bulut Teknolojisi diğer adıyla Bulut Bilişim, kullanıcıların sanal alanda depolanan dijital kaynakları ağlar yoluyla kullanmalarına sağlayarak, fiziksel konumlarının kısıtlanması olmadan internet üzerinden bilgi ve uygulamaları paylaşımlarına olanak tanır. Bulut yapısından önce üzerinde çalışma yapılan bilgi ve dosyaların başka bilgisayarlarda kullanmak istendiğinde harici belleklerle dosyaların taşınması gerekmektedir. Fiziksel olarak yapılan bu taşıma işlemi öncelikle zaman kaybına yol açmaktadır. Dosyaların farklı konumlarda benzer isimlerle saklanması aynı isimde birden fazla dosya bulunması karışıklığa sebep olabilmektedir. Bir yere taşınan çalışmanın tüm bilgisayarlar veya mobil cihazlara aktarılması karışıklığa sebep olmaktadır.



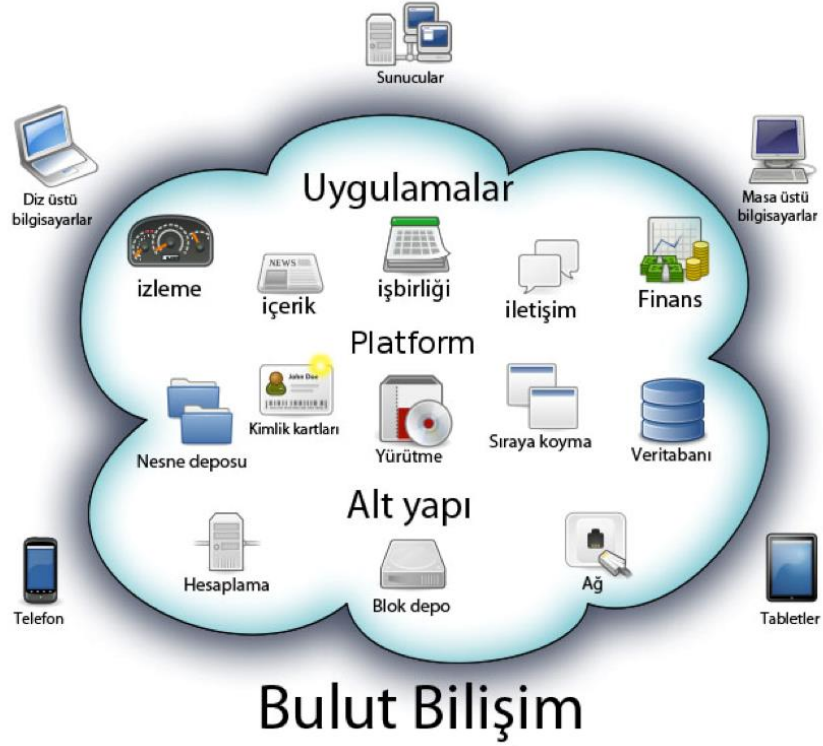
Tüm bunlara çözüm olarak internet altyapısının hızlanmasıyla internet ağını kullanarak verilerin bir yapı altında saklanması tüm bilgisayarlardan dosyalara erişim sağlanması buna çözüm olarak kullanılmıştır. Böylece sadece bilgisayarlarla değil mobil cihazlar gibi farklı platformlardan dosyalara erişim sağlanmıştır. Ayrıca bağlantı olanağının olması veritabanı işlemleri, dosya işlemleri, yapay zeka uygulamaları, kullanıcı kontrollü uygulamalar geliştirme imkanı, cihazlara anlık bildirimler gönderme gibi birçok imkanın oluşmasına sebep olmuştur. 512 bit gibi şifreleme algoritmaları ile saklanan verilerin güvenliği ve sadece yetkili kişiler tarafından kullanılabilir olması, veri güvenliği açısından çok önemlidir.



Bulut bilişimin çalışması için üç ana bileşenin mevcut olması gerekir;

1. **Ön Uç(Front-end)** : İstemcinin bilgisayarını veya mobil cihazını ve bulut bilişim sistemine erişmek için gereken uygulamayı ifade eder. Çoğu durumda, uygulama bir web tarayıcısıdır, ancak diğer sistemler özel uygulamalar gerektirebilir.
2. **Arka Uç(Backend)**: Bilgisayarlar, sunucular, işletim sistemleri ve depolama cihazları gibi bulut sağlayıcının sahip olduğu bilgisayar altyapısını ifade eder. Bulutta depolanan tüm veriler genellikle depolama cihazlarında saklanırken, bulutta çalışan yazılım uygulamaları bilgisayarlarda çalıştırılır. Her yazılım uygulamasında ayrıca özel bir sunucu bulunurken, trafiği ve istemci taleplerini izlemek ve her şeyin gerektiği gibi sorunsuz çalışmasını sağlamak için merkezi bir sunucu kullanılacaktır.
3. **Ağ(Network)**: Yukarıdaki iki bileşenin internet üzerinden bağlanmasına izin verdiği için temel bileşendir. Bulut hizmeti sağlayıcıları, çeşitli bulut hizmeti modellerini kullanarak bilgi işlem hizmetlerinin çeşitli kullanıcılara sorunsuz bir şekilde sunulmasını sağlamak için tüm bu bileşenleri bir araya getirir. Bulut bilişim kullanan tipik bir örneği ele alalım. Gmail, Yahoo, Yandex veya Hotmail gibi bir e-posta hizmetine sahip bir e-posta hesabınız varsa, bilgisayarınıza veya mobil cihazınıza bir e-posta programı yüklemeyen web üzerinden e-posta hesabınızda oturum açabilirsiniz. E-posta hesabınızdaki tüm bilgilere ve dosyalara web üzerinden de erişebilirsiniz. E-posta hesabınızın yazılımı ve deposu cihazınızda yoktur, bunun yerine hizmetin bilgisayar bulutunda mevcuttur.

BULUT BİLİŞİM TÜRLERİ



Bulut teknolojilerinin sağladığı imkanları değerlendirirsek üç çeşit bulut bilişim hizmeti bulunur. Bunlar;

1. **Platform hizmeti (PaaS)** : Geliştiricilere kendi yazılımlarını, web uygulamalarını veya diğer programlama projelerini oluşturmaları için kullanımı kolay bir platform sağlayan bir bulut bilişim çözümüdür.
2. **Yazılım hizmeti (SaaS)** : Kullanıcıların yazılım uygulamalarına ve çeşitli bileşenlerine cihazlarında veya sabit disklerinde indirmeye, yüklemeye veya depolamaya gerek kalmadan erişebildiği bir bulut bilişim biçimidir. Eposta, takvim Microsoft ofis 365 örnek olarak verilebilir.
3. **Altyapı hizmeti (IaaS)** : Kullanıcıların sunuculara, güvenlik duvarlarına, sanal makinelere, depolamaya ve diğer altyapılara erişimini sağlar. Kullanıcı geleneksel yöntemlerden farklı olarak sunucu, yazılım almak yerine hizmeti sağlayan firmaların sanallaştırılmış altyapısından işlem gücü ve depolama alanı gibi hizmetler satın alır.

BULUT TEKNOLOJİSİNİN KULLANILDIĞI ALANLAR

Bulut teknolojileri Linux, Windows, iOS gibi farklı işletim sistemlerinde çalışabilen uygulamalar geliştirmek için idealdir. Farklı işlemci mimarisine sahip cihazların komut setleri farklı olmaktadır. Bulut teknolojilerini kullanarak geliştirilen programlar ile farklı komut setlerine sahip donanımlara yazılım geliştirmek mümkün

olmaktadır. Mobil cihazlardan, tabletlere, bilgisayarlardan oyun konsollarına farklı cihazları sanallaştırılmış ortak bir platformda buluşturma imkanı verir.

Bulut Teknolojisi, verimliliği ve hizmet sunumunu iyileştirmek için neredeyse her alanda kullanılabilir. Bunlar;

1. Yeni uygulamalar oluşturma.
2. Uygulama test etme.
3. Veri depolama.
4. Yedekleme ve felaket kurtarma.
5. Makine öğrenimi ve yapay zeka (AI) kullanma.
6. Verileri analiz etme.
7. İsteğe bağlı yazılım sunma.
8. Ses ve görüntü akışı sağlama.



BULUT TEKNOLOJİSİNİN FAYDALARI

İş hayatında yapılan çalışmalarda yaşanan en büyük kayıp bilginin aktarılması için harcanan zamandan oluşmaktadır. Ağ teknolojileri sayesinde iş hayatında verimliliğin %70 oranında arttığı Cisco tarafından belirtilmektedir. Bulut teknolojileri sayesinde tüm ortamlarda tek tıkla iş ve işlemlerin yapılmasına olanak sağlayarak bu oranı artırmıştır. Günümüzde sıraya girme, bekleme, evrak taşıma, planlı hareket etme kabiliyetlerinde meydana gelen dijital avantajlar ile tapu, bankacılık, vergi işlemleri gibi kurumlar arası işlemlerin de hızlanarak gelişmesi büyük bir zaman kazancına sebep olmuştur.

Bulut teknolojileri;

1. Maliyetleri düşürür.
2. Altyapı karmaşasını ortadan kaldırır.
3. Çalışma alanını genişletir.
4. Verileri korur.
5. İstenilen zamanda bilgiye ulaşma imkanı verir.

FIREBASE



Firestore, mobil veya web geliştiricilerinin iOS, Android ve Web uygulamaları geliřtirmelerine olanak tanıyan, Google destekli backend hizmeti sunan bir **bulut teknolojisidir**.

Firestore geliřtirirken gerek zamanlı veritabanı (database), kimlik doęrulama (authentication), depolama (storage) bildirim (notification) gönderme gibi birçok hizmet sağlamaktadır. Firestore mobil veya web uygulamaları için bütün veriyi bulutta json formatında saklamaktadır. Tek bir platform üzerinden bulutta saklanan verilere web, ios veya mobil uygulamalar üzerinden istedięimiz zaman ulaşabiliriz. Aynı zamanda sunucu taraflı kodlamaya gerek olmadığı için geliřtiriciler için zaman kazandırmaktadır.

Uygulama yönetim, kullanım takip, veri depolama, bildirim gönderme gibi işlemleri, extra bir sunucuya ve sunucu taraflı kod yazmaya gerek kalmadan halleden Firestore, yeni geliřtirici dostu arayüzünde Realtime Database, Notification, Remote Config gibi özelliklerle donatılmış her uygulama için ayrı ayrı ulaşım imkanı veriyor. FCM - Firestore Cloud Messaging servisiyle de řimdiye kadar geliřtiricilerin anlık bildirim - push notification göndermek için kullandığı GCM - Google Cloud Messaging' in yerini alacak olan Firestore, bildięimiz tablolar ve sql yerine verileri root-child řeklinde organize edilen json veri paracıklarında saklıyor.

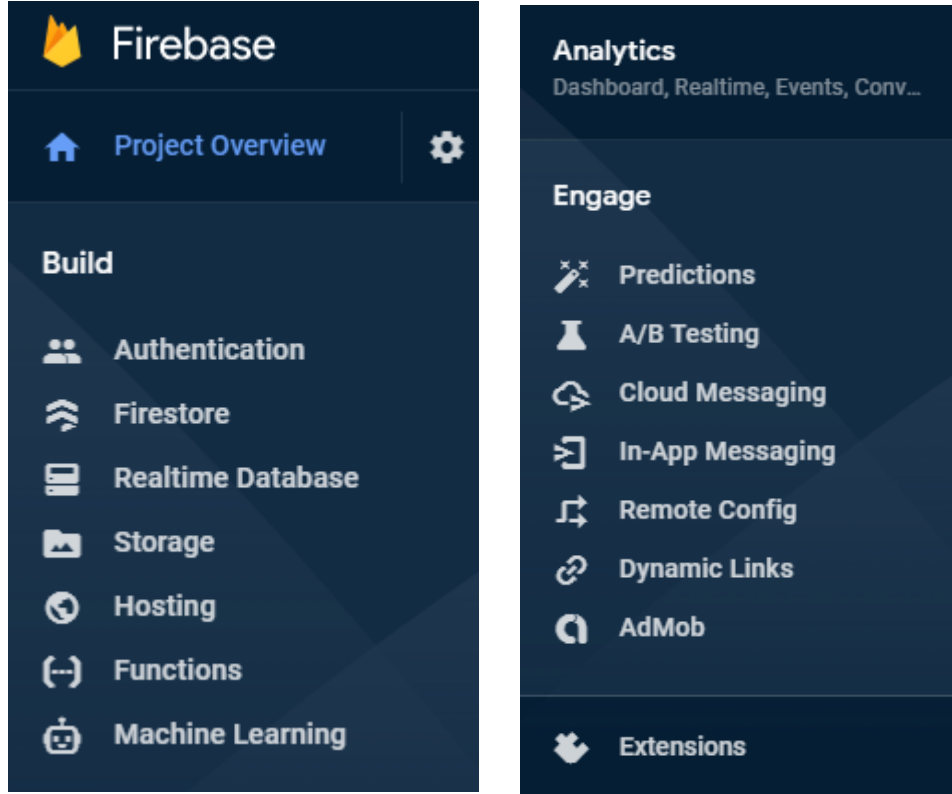
Firestore Hizmetleri

Bir firestore projesi oluşturduğumuzda ařağıdaki 4 hizmeti de kullanabilmekteyiz. Bu hizmetlerin dışında yapay zeka uygulamaları gerekleřtirilmesi gibi başka hizmetler de bulunmaktadır. Ancak bizim alışmamızda yapay zeka konusunu ayrı bir modülde işlediğimiz için firestore AI hizmeti burada anlatılmayacaktır.

- **Authentication** : Uygulamaya kaydolun kullanıcıların bilgilerini doęrulamak için bu servisi kullanılır.
- **Realtime Database**: İsminden de anlaşılacağı üzere gerek zamanlı bulut üzerinde alışan bir NoSQL veritabanı sistemidir. Veri JSON olarak saklanır. Veriyi asenkron olarak eker. Verilere anlık olarak ulaşabiliyoruz
- **Storage**: Web sayfalarındaki hosting gibi düşünebiliriz. Bu servis ile bilgisayarımızda veya sunucumuzda bulunan resim, video ve metin gibi dosyaları firestore bulut teknolojisinde saklayabiliriz. Kullanıcılar bu dosyaları istedięi zaman indirip gerekli güncellemeleri yapıp tekrar gönderebilir. Anlık

olarak ses, resim , video ve diğer dosyaları aktarmada kullanabiliriz. Mesajlaşma programlarında olduğu gibi.

- **Notification:** Cloud messaging olarak adlandırılmaktadır. Bulut üzerinden uygulamayı kullanan bütün kullanıcılara anlık mesaj gönderebiliriz.

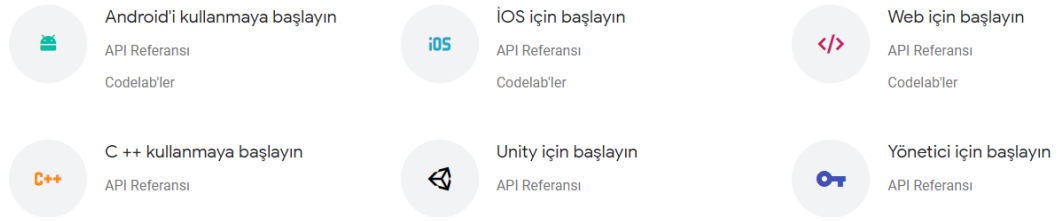


Neden kullanmalıyız?

Basit ve kullanışlı bir ara yüze sahip olmasından dolayı hiç android bilmeyen bir kullanıcı bile Firebase sayesinde bir uygulama yazabilir, bu uygulamada kullandığı verileri kullanıcıları Firebase'in sitesinden rahatlıkla takip edebilir. Masaüstü bilgisayarı veya bir sunucu üzerinde python kullanarak program geliştirmek gerekiyorsa bulut yapısı sayesinde bu oldukça kolay olmaktadır. Farklı platformlarda kullanılan verilerin dönüştürülmesi gibi işlemlerin yapılması ihtiyacı ortadan kalmış olmaktadır.

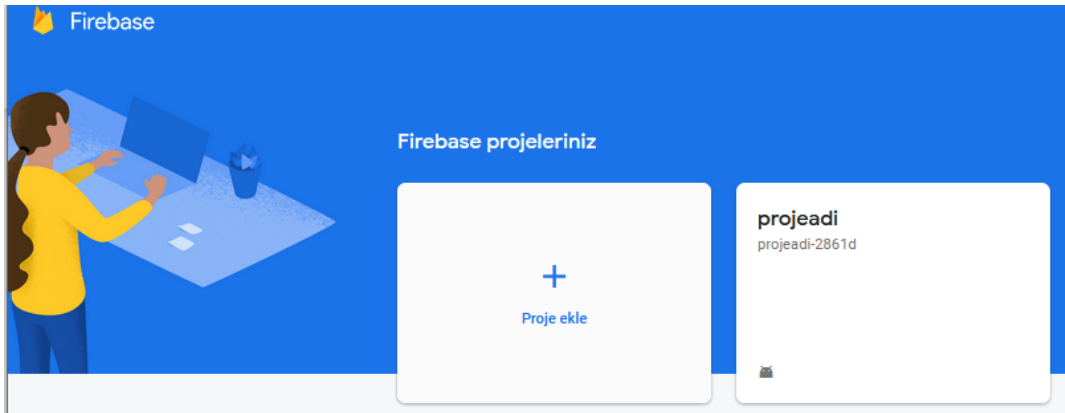
Platforma göre Firebase

Firebase, yüksek kaliteli uygulamalar geliştirmek, kullanıcı tabanınızı büyötmek ve daha fazla para kazanmak için size araçlar sunar. İşletmenizden para kazanabilmeniz ve kullanıcılarınıza odaklanabilmeniz için temel noktaları ele alıyoruz.



Firebase kullanımı için pek çok Google servisinde olduğu gibi bir Google kimliğine sahip olmanız gerekmektedir. Bu aşamada varsa sahip olduğunuz Google Mail (GMail) hesabınızdan faydalanabilir, Firebase web sayfası üzerinden giriş yaparak kullanıcı paneline ulaşabilirsiniz. Android i Kullanmaya Başlayın butonunu tıklayarak ya da sağ üst bölümde yer alan Konsola git bağlantısını tıklayarak ilerlenebilir. Bir proje uygulama için platformlar arası bir kapsayıcıdır. Bu sayede Android, iOS ve web uygulamaları arasında Veritabanı, Kullanıcı Yönetimi ve Uzak Yapılandırma gibi özellikler paylaşılabilir hale gelmektedir.

<https://console.firebase.google.com/> adresinden Proje Adı, Proje Kimliği, Konumlar ve ilgili diğer alanları da uygun şekilde doldurduktan sonra Proje Oluştur butonu ile proje oluşturma sürecini de tamamlamış olmaktadır.



FİREBASE PROJESİ OLUŞTURMA

Her bir Google hesabı için birden fazla proje oluşturma imkanı bulunmaktadır. Böylelikle birden fazla yazılım geliştirmek için ayrı hesaplar açma ihtiyacı ortadan kaldırılmış olmaktadır.

<https://console.firebase.google.com/> adresinden giriş yaparak işlemlerimize başlayalım.

1. Projemize isim verme

Consol arabiriminde Yeni Proje seçerek projemize isim vererek işlemlerimize başlıyoruz.



2. Google Analitik yapılandırma

Bu adımdan sonra Devam edildikten sonra aşağıdaki ekran gelmektedir.

Google Analytics; internet sitenize veya mobil uygulamanıza giren kullanıcıların sitenizle olan etkileşimleriyle ilgili raporlar sunan, kullanımı ücretsiz bir web analiz aracıdır. Analytics 360 adı altında ücretli versiyonu da bulunmaktadır.

Google Analytics 4 temel ana bileşen üzerine kurulmuştur.

Veri toplama: Google Analytics'te veriler javascript kod parçacıkları yardımıyla toplanır. Bu kod satırı kullanıcıların hangi URL'i ziyaret ettiğini, hangi browser'dan veya cihazdan giriş yaptığını analytics hesabınıza aktarır.

Veri işleme: Toplanan veriler Google'ın sunucularına gönderilir ve kategorize edilerek anlamlı hale getirilir.

Konfigürasyon: Veriler işlendikten sonra veritabanına kaydedilir. Veri tabanına kaydedilen veriler bir daha değiştirilemez. Örneğin filtrelenmiş hesabınızda filtreyi kaldırırsanız bile eski verilere ulaşamazsınız.

Raporlama: Toplanan verilerin Google Analytics kullanıcılarına sunulduğu bileşendir.

Bu kısımda Google firebase için Google Analytics eklentisini kurmasını önermektedir.

Google Analytics'i yapılandırın

Google Analytics hesabı seçin veya oluşturun ⓘ

Default Account for Firebase

Bu hesapta otomatik olarak yeni bir mülk oluşturun ✎

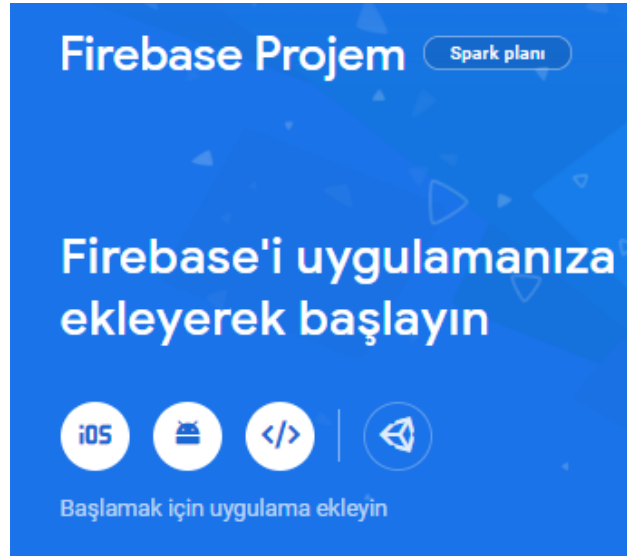
Proje oluşturulduktan sonra seçtiğiniz Google Analytics hesabında yeni bir Google Analytics mülkü oluşturulur ve Firebase projenize bağlanır. Bu bağlantı, ürünler arasında veri akışını etkinleştirir. Google Analytics mülkünüze Firebase'e aktarılan veriler, Firebase hizmet şartlarına tabidir. Google Analytics'e aktarılan Firebase verileri ise Google Analytics hizmet şartlarına tabidir. [Daha fazla bilgi edinin](#)

Önceki

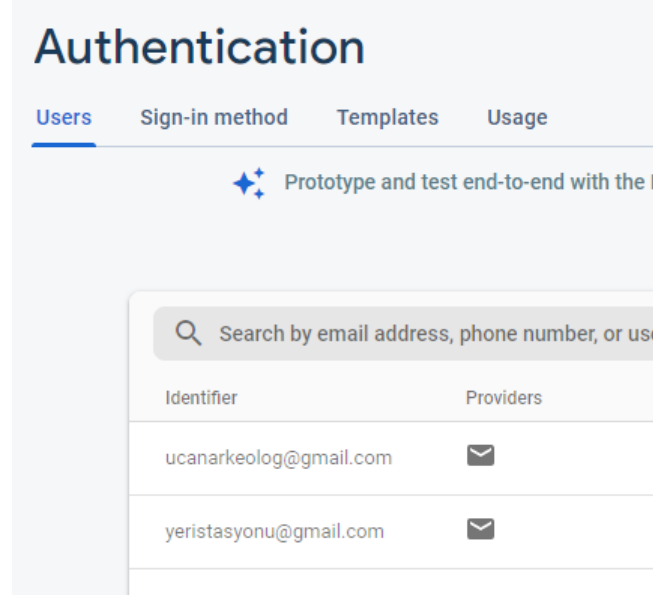
Proje oluşturun

3. Projeyi Oluşturma

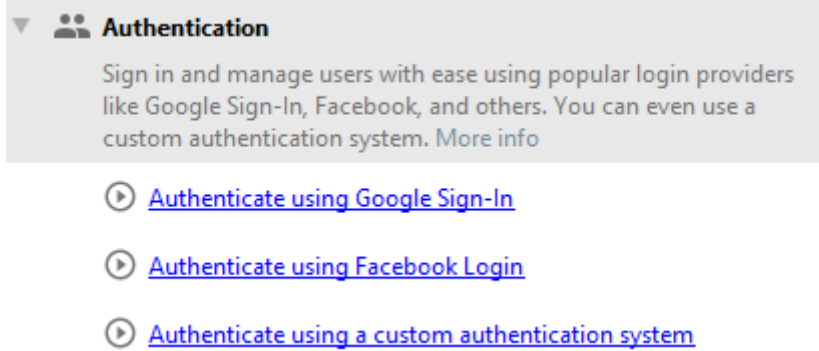
Aşağıdaki ekran ile karşılaşmış isek projemiz başarı ile kurulmuştur. Her yeni projede bu adımlar takip edilmelidir.



FIREBASE AUTHENTICATION (KİMLİK DOĞRULAMA)



Android Studio da **Firebase Authentication** ile ilgili işlemlerimizi gerçekleştirebilmemiz için projemizi Firebase konsolda tanıtmalıyız. Kimlik doğrulama ile ilgili gerekli fonksiyonları kullanabilmemiz için **Android Studio** da **Tools-> Firebase-> Authentication** kısmını açıyoruz.



Authenticate using Google Sign-in seçeneğini tıkladıktan sonra karşımıza çıkan adımları tek tek uyguluyoruz.

FİREBASE AUTHENTICATION AYARLARI

1. Connect your app to Firebase
2. Add Firebase Authentication to your app

Authenticate using Google Sign-In

You can let your users authenticate with Firebase using their Google Accounts by

[Launch in browser](#)

① Connect your app to Firebase

[Connect to Firebase](#)

② Add the Firebase Authentication SDK to your app

[Add the Firebase Authentication SDK to your app](#)

NOTE: After adding the SDK, here are some other helpful configurations to c

- **Are you using Kotlin?**
You can use the [Firebase KTX libraries](#) to write idiomatic Kotlin. Chan
- **Do you want an easier way to manage library versions?**
You can use the [Firebase Android BoM](#) to manage your Firebase libr

Bu adımları sırayla uyguladığımızda projemizi Firebase' e tanıtmış olduğumuz authentication ile ilgili kodları ve gerekli dosyaları da projemize otomatik olarak eklemiş oluruz.

① Connect your app to Firebase

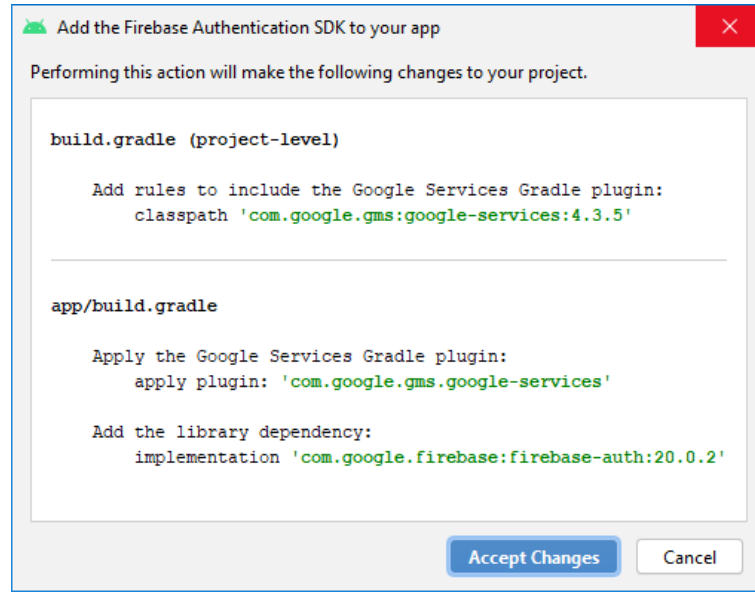
✓ Connected

② Add the Firebase Authentication SDK to your app

[Add the Firebase Authentication SDK to your app](#)

NOTE: After adding the SDK, here are some other helpful configurations to

- **Are you using Kotlin?**
You can use the [Firebase KTX libraries](#) to write idiomatic Kotlin. Ch
- **Do you want an easier way to manage library versions?**
You can use the [Firebase Android BoM](#) to manage your Firebase lib










build.gradle dosyasını incelediğimizde

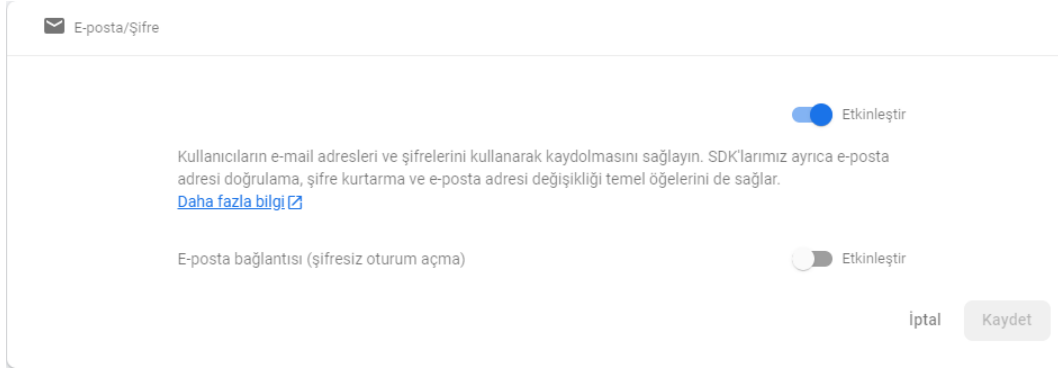
```
implementation 'androidx.appcompat:appcompat:1.2.0'  
implementation 'com.google.android.material:material:1.2.1'  
implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
implementation 'com.google.firebase:firebase-auth:20.0.2'  
testImplementation 'junit:junit:4.+'  
androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
```

implementation 'com.google.firebase:firebase-auth:20.0.2' satırının eklenmiş olduğu görülecektir.

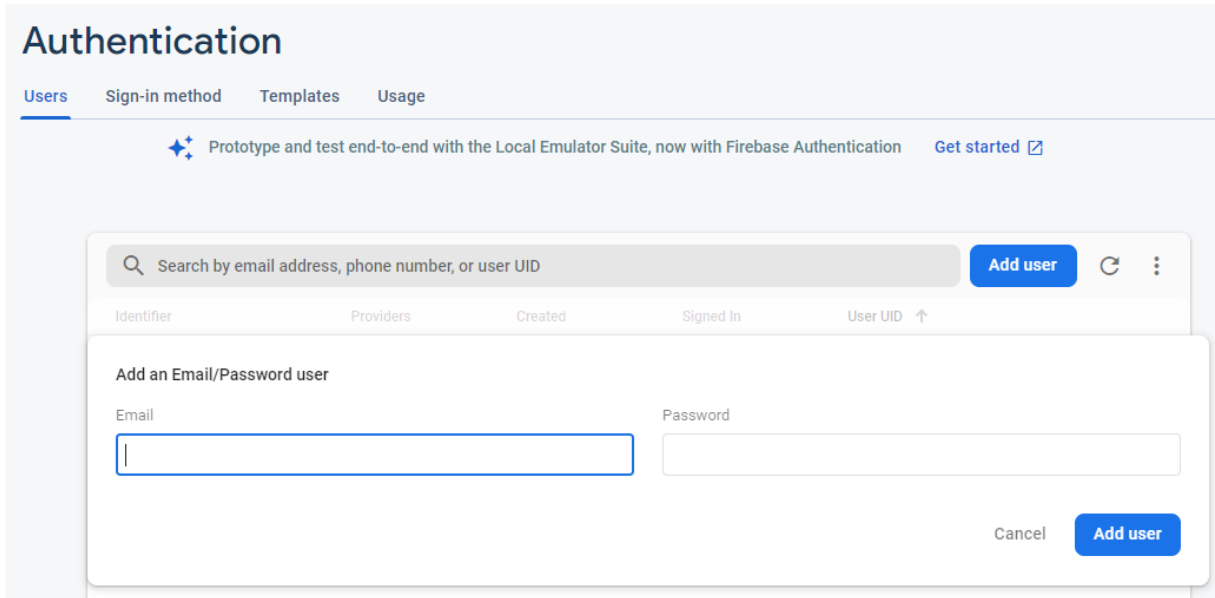
Firebase authentication için bir çok kimlik doğrulama yönetimi sunmaktadır.

Oturum açma sağlayıcıları	
Sağlayıcı	Durum
 E-posta/Şifre	Etkin
 Telefon	Devre dışı
 Google	Devre dışı
 Play Oyunlar	Devre dışı
 Game Center	Devre dışı
 Facebook	Devre dışı
 Twitter	Devre dışı

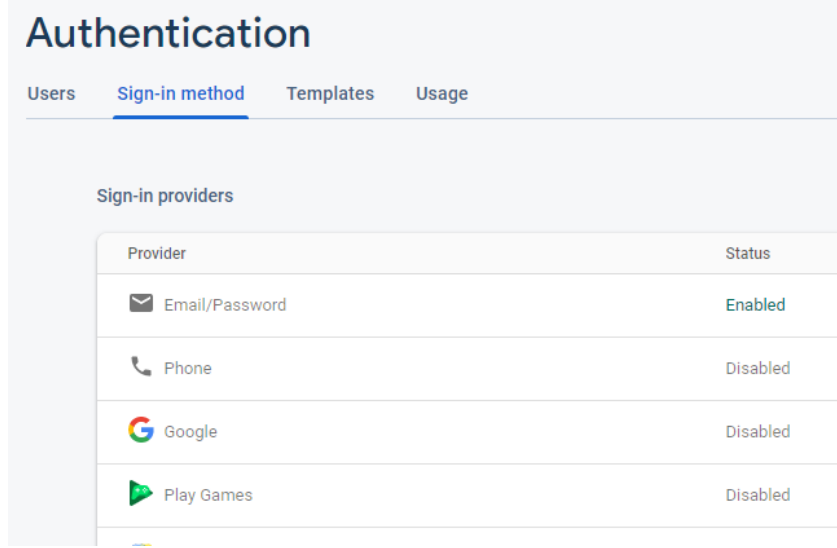
Yukarıdaki ekran görüntüsünde E-posta ve parola ile girişi kullanılmaktadır; uygulama içinde kullanabilmemiz içinde bu yöntemi aktif hale getirmemiz gerekiyor. Aşağıdaki ekran görüntüsünde gördüğünüz gibi ilgili projemizi seçip **Authentication->Sign-in Method->Eposta/Şifre** yi etkinleştirmemiz gerekmektedir.



Firebase de otantikasyon yapılması için kullanıcılar tanımlanabilir. Her kullanıcının farklı bir platformdaki bir proje için kullanılabilir. Böylece her bir projenin aktivitelerinin takibi yapılabilir. Bir kullanıcı tanımı eklemek için **Users** sekmesi altından **Add User** komutu seçilir. Eposta ve şifre girilerek kullanıcı eklenebilir.

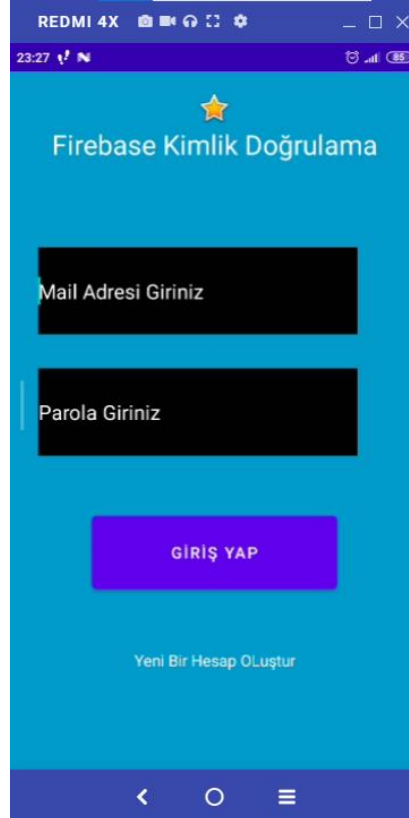


Giriş yönteminin seçilmesi için Sign-in method sekmesi altından farklı giriş teknikleri kullanılabilir. Bizim ufulamalarımızda sadece eposta üzerinden giriş gerçekleştirilmesi için ayarlama yaptık. Diğer yöntemleri güvenlik gerekçesi ile kapattık.



ANDROID STUDIO PROJESİNE FIREBASE PROJESİNİ BAĞLAMA

Uygulama çalıştırıldığında ilk çalışacak olan MainActivity sınıfımızdır. MainActivity çalıştığında authenticate olmuş bir kullanıcı yok ise LoginActivity sayfasını açtırıyoruz ve karşımıza gelen ekranda login butonu ve altında da register sayfasına yönlendirdiğimiz bir textview bulunmaktadır. signInWithEmailAndPassword methodu kullanıcıdan email adresi ile parola alıyor ve login isteğinde bulunuyor. Daha sonra addOnCompleteListener ile de işlemin başarılı olup olmadığını kontrol ediyoruz.



Mainactivity.java

```
package com.example.firebasekimlikdogrulama;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.ToolbarWidgetWrapper;
import androidx.appcompat.widget.TooltipCompat;
import androidx.appcompat.widget.Toolbar;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {
    private FirebaseAuth FirebaseAuth;
    private androidx.appcompat.widget.Toolbar mToolbar;

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        super.onOptionsItemSelected(item);
        if(item.getItemId()==R.id.action_signout)
        {
            FirebaseAuth.signOut();
            Toast.makeText(getApplicationContext(),"oturum
kapatıldı", Toast.LENGTH_SHORT).show();
            Intent loginIntent=new Intent(MainActivity.this, LoginActivity.class);
            startActivity(loginIntent);
        }
        return true;
    }

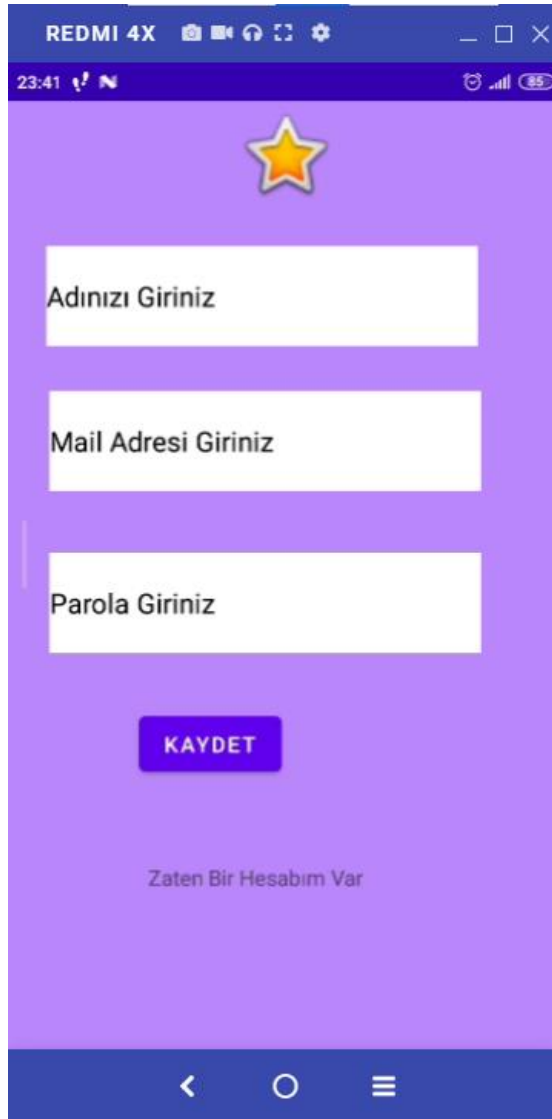
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FirebaseAuth=FirebaseAuth.getInstance();
        mToolbar=(Toolbar) findViewById(R.id.mainToolbar);
        setSupportActionBar(mToolbar);
        getSupportActionBar().setTitle("Main Activity");

        if(FirebaseAuth.getCurrentUser()==null)
        {
            Intent loginIntent=new Intent(MainActivity.this, LoginActivity.class);
            startActivity(loginIntent);
        }
    }
}
```

```
Toast.makeText(getApplicationContext(),"Lütfen Giriş  
Yapınız",Toast.LENGTH_SHORT).show();  
  
}  
  
}  
  
}
```

Eğer kullanıcı sisteme kayıtlı değilse kimlik doğrulamasını sağlamak için Yeni Bir Hesap Oluştur kısmından RegisterActivity sayfasına yönlendiriyoruz.

createUserWithEmailAndPassword methodu email ve password ile yeni bir kullanıcı oluşturuyor, **addOnCompleteListener** ile de işlemin başarılı olup olmadığını kontrol ediyoruz eğer kayıt işleminde sorun yok ise kullanıcıyı MainActivity sayfasına yönlendiriyoruz aksi takdirde hata mesajı ile kullanıcıyı uyarıyoruz.



RegisterActivity.java

```
package com.example.firebasekimlikdogrulama;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class RegisterActivity extends AppCompatActivity {
    private EditText regName;
    private EditText regEmail;
    private EditText regPassword;
    private TextView regToLogin;
    private Button regButton;
    private ProgressDialog registerProgress;
    private FirebaseAuth fireAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);
        regButton=(Button) findViewById(R.id.registerButton);
        regName=(EditText) findViewById(R.id.registerName);
        regEmail=(EditText) findViewById(R.id.registerEmail);
        regPassword=(EditText) findViewById(R.id.registerPassword);
        regToLogin=(TextView) findViewById(R.id.registerToLogin);
        registerProgress =new ProgressDialog(this);
        fireAuth=FirebaseAuth.getInstance();

        regToLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent loginIntent=new Intent
(RegisterActivity.this, LoginActivity.class);
                startActivity(loginIntent);
            }
        });
        regButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String name=regName.getText().toString();
                String password=regPassword.getText().toString();
                String email=regEmail.getText().toString();
```

```

if(TextUtils.isEmpty(name)||!TextUtils.isEmpty(password)||!TextUtils.isEmpty(email
))
    {
        registerProgress.setTitle("Kaydediliyor");
        registerProgress.setMessage("hesabınızı oluşturuyoruz, lütfen
bekleyiniz...");
        registerProgress.setCanceledOnTouchOutside(false);
        registerProgress.show();
        register_user(name,password,email);
    }
    });
}

private void register_user(String name, String password, String email) {

fireAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful())
        {
            registerProgress.dismiss();
            Intent mainIntent=new
Intent(RegisterActivity.this,MainActivity.class);
            startActivity(mainIntent);

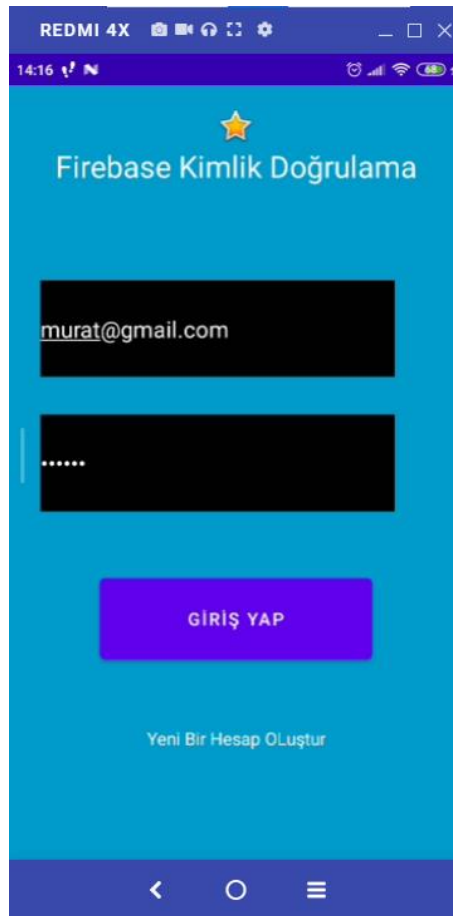
        }
        else
        {
            registerProgress.dismiss();

Toast.makeText(getApplicationContext(),"hata:"+task.getException().getMessage(),To
ast.LENGTH_SHORT).show();

        }
    }
});
}
}

```

LoginActivity.java



Kullanıcı sisteme kayıtlı ise eposta ve parola bilgileri girildikten sonra bunları kontrolü yine LoginActivity sayfasında kontrol edilmektedir.

```
package com.example.firebasekimlikdogrulama;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
```

```

public class LoginActivity extends AppCompatActivity {
    private ProgressDialog logProgress;
    private FirebaseAuth fireAuth;
    private EditText logEmail;
    private EditText logPassword;
    private Button logButton;
    private TextView logtoRegister;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        fireAuth=FirebaseAuth.getInstance();
        setContentView(R.layout.activity_login);
        logEmail=(EditText)findViewById(R.id.LoginEmail);
        logPassword=(EditText)findViewById(R.id.LoginPassword);
        logButton=(Button)findViewById(R.id.LoginButton);
        logtoRegister=(TextView)findViewById(R.id.LoginNeedAccount);
        logProgress=new ProgressDialog(this);

        logtoRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent registerIntent=new
Intent(LoginActivity.this,RegisterActivity.class);
                startActivity(registerIntent);
            }
        });
        logButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String eposta=logEmail.getText().toString();
                String parola=logPassword.getText().toString();
                if(!TextUtils.isEmpty(eposta)||!TextUtils.isEmpty(parola))
                {
                    logProgress.setTitle("Oturum Açılıyor...");
                    logProgress.setMessage("lütfen bekleyiniz");
                    logProgress.setCanceledOnTouchOutside(false);
                    logProgress.show();

                    fireAuth.signInWithEmailAndPassword(eposta,parola).addOnCompleteListener(new
                    OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult> task) {
                            if (task.isSuccessful())
                            {
                                logProgress.dismiss();
                                Toast.makeText(getApplicationContext(),"Giriş
Başarılı",Toast.LENGTH_SHORT).show();
                                Intent mainIntent=new
                                Intent(LoginActivity.this,MainActivity.class);
                                startActivity(mainIntent);
                            }
                            else
                            {
                                logProgress.dismiss();
                                Toast.makeText(getApplicationContext(),"Giriş
Başarısız!" +task.getException().getMessage(),Toast.LENGTH_SHORT).show();

```



```
}  
  
}  
});  
  
}  
});  
  
}
```

UÇANDROİD UYGULAMASINDA FIREBASE BULUT YAPISI

Uçandroid uygulamasında yaptığımız eklentileri inceleyelim. Firebase veritabanı, dosya servisi, kullanıcı otantikasyon hizmetlerinin eklenmesi için ilgili implementation satırlarını ekliyoruz.

build.gradle dosyamızda;

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'com.google.android.material:material:1.0.0-alpha3'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.firebase:firebase-database:18.0.1'
    implementation 'com.google.firebase:firebase-storage:18.1.1'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test:runner:1.2.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    // Add the Firebase SDK for Google Analytics
    implementation 'com.google.firebase:firebase-analytics:17.0.1'

    // Add the SDKs for any other Firebase products you want to use in your app
    // For example, to use Firebase Authentication and Cloud Firestore
    implementation 'com.google.firebase:firebase-auth:18.1.0'
    implementation 'com.google.firebase:firebase-firestore:20.2.0'

    // Getting a "Could not find" error? Make sure that you've added
    // Google's Maven repository to your root-level build.gradle file
}
```

```
'com.google.firebase:firebase-database:18.0.1',
'com.google.firebase:firebase-storage:18.1.1',
'com.google.firebase:firebase-auth:18.1.0', 'com.google.firebase:firebase-
firestore:20.2.0' sürümleri farklı görünse de kullandığımız bulut teknolojisinde
birbiriyle uyumlu olan sürümlerdir.
```

activity_main.xml dosyası içerisinde bulut sistemine bağlantı gerçekleştirildiğinde aktif olan, bağlantı gerçekleştirmediğinde pasif olan bir switch ekliyoruz.

```
activity_main.xml;
```

```
<Switch
    android:id="@+id/sw_firebase"
    android:layout_width="match_parent"
    android:layout_height="30dp"
    android:layout_above="@+id/btn_pixbaglan"
    android:layout_marginBottom="25dp"
    android:text="Bulut Baglantisi"
    android:textOff="YOK"
    android:textOn="VAR"
    android:textSize="20sp" />
```

MainActiviy.Java içerisinde ilgili kütüphanelerimizi ekliyoruz

```
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
```

MainActiviy.Java içerisinde değişken tanımlamalarımızı yapıyoruz.

```
public class MainActivity extends AppCompatActivity implements
TextToSpeech.OnInitListener {
```

metodu içerisinde bulut yapısında olan değişikliklerin anlık olarak işleme alınması için aşağıdaki değişkenleri tanımlıyoruz.

```
private FirebaseAuth auth;
private FirebaseAuth.AuthStateListener authListener;
FirebaseStorage storage = FirebaseStorage.getInstance();
StorageReference storageRef;
FirebaseDatabase database;
DatabaseReference dbref;
public Uyeler uyeler;
//DatabaseReference myRef = database.getReference("message");
String KullaniciAdi;
String Eposta;
String sifre;
String KullaniciId;

ArrayList<Gorev> gorevler = new ArrayList<Gorev>();
int SuankiGorev=0;
```

Bulut yapısına bağlanabilmek için Init fonksiyonunda verilerimizi hazırlıyoruz.

```
public void init()
```

içerisine verilerimizi hazırlıyoruz. Burada tanımlanan giriş bilgilerini kendi projenize uygun olarak strings.xml dosyası içerisinden düzenlemelisiniz. Bu bilgileri strings içerisinden okuyarak değişkenlerimize saklıyoruz.

```
auth = FirebaseAuth.getInstance();
database = FirebaseDatabase.getInstance();
dbref = database.getReference().child("Gorevler");
storageRef = storage.getReference();

swFirebase= (Switch) findViewById(R.id.sw_firebase);
swDron= (Switch) findViewById(R.id.sw_pixhawk);
KullaniciAdi=getString(R.string.Kullaniciadi);
Eposta=getString(R.string.Eposta);
sifre=getString(R.string.Sifre);
tts = new TextToSpeech(context, this);

loginKullanici();
```

Bu aşamada loginKullanici metodu ile giriş işlemlerimizi gerçekleştiriyoruz. Giriş işleminin gerçekleştirilmesi internet bağlantısının olup olmaması durumuna göre değişeceğinden `addOnCompleteListener` ile takip edilmektedir. Program aktif olarak kullanılmaya devam ederken bağlantı sağlanana kadar “Sunucu bağlantısı yapılmadı” ifadesi sesli uyarı ve switch üzerinde görünecektir. Bağlantı sağlandığı anda `onComplete` methodu çalıştırılacaktır. Bağlantı oluşturulduğunda ilgili switch açılacak ve kullanıcı ID bilgisi saklanacaktır.

```
// Firebase Login işlemleri
private void loginKullanici() {

    auth.signInWithEmailAndPassword(Eposta, sifre).addOnCompleteListener(new
    OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(task.isSuccessful()) {
                Konus("Sunucu Bağlantısı Yapıldı");
                swFirebase.setChecked(true);
                swFirebase.setText("Sunucu Baglantisi Var");
                KullaniciId = auth.getCurrentUser().getUid();
            }
            else
            {
                Konus("Sunucu Bağlantısı Yapılamadı");
                swFirebase.setChecked(false);
                swFirebase.setText("Sunucu Baglantisi Yok");
            }
        }
    });
}
```

Otantikasyon işlemi gerçekleştirildiğinde uygulamanın veri tabanı işlemleri ve dosya işlemlerine erişim hakkı sağlanmış olacaktır. Tüm aktivitelerde işlemlerin yürütülmesi için öncelikle otantikasyon işlemlerinin gerçekleştirilmesi gerekmektedir.

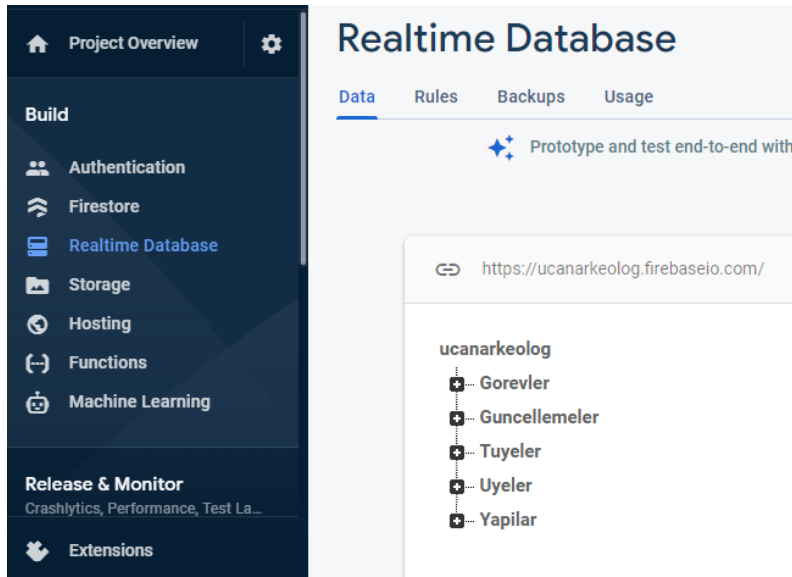
Android programlama tarafında gerçekleştirilen diğer işlemleri Mobil Programlama modülümüzde bulabilirsiniz.

FIREBASE VERİTABANI İŞLEMLERİ

Firebase de veri tabanı oluşturmak için iki teknik bulunur. Bunlar Realtime veritabanı ve cloud veritabanıdır.

Bir projemizde Firebase'in Realtime Database(Gerçek Zamanlı Veritabanı) bulut servisini kullanacağız. Bu veri tabanı verileri depolayıp almak için "tag"ler ve veri çiftleri kullanır. Alışlagelen her bir veri kaydının bir ID numarası ile saklandığı modellerden farklı bir yapısı vardır. Verileri bulabilmek için için ID ve sütun adı yerine tag ler kullanılır. Bu tag o veriye verilen isimdir. Firebase Realtime veritabanı sade ve basit bir kullanımı mevcut olup geliştirici için esnek bir yapı sağlıyor. Bu şekilde bir obje referans edip kaydedebileceğiniz gibi string bir ifadeyi ya da herhangi bir primitif değeri de kaydedebilirsiniz. Bir tag üzerine birden fazla kayıt yaparsanız herbiri yeni bir kayıt olarak eklenmez. Her bir kayıt önceki verinin üzerine kaydedilir. Eski kayıt silinir yeni kayıt kullanılır hale gelir.

FirebaseDatabase kütüphanesi ile veri tabanından veri okuma/ekleme/güncelleme/silme işlemleri, veri tabanındaki değişiklikleri dinleme, Rules kuralları ile veri tabanı güvenliği ile ilgili işlemler yapılmaktadır.



Realtime veri tabanının özelliği veritabanında olan değişiklikler anlık olarak otantikasyonu yapılmış tüm cihazlara bildirilir. Android bir programda callback interface kullanarak verideki herhangi bir değişikliği anlık olarak alıp bir viewde görüntüleyebiliyoruz. Yani telefondaki program kullanıcının elinde çalışır halde iken veri tabanında bir değişiklik meydana geldiğinde telefona bilgi değişikliği gönderilir. OnCompleteListener gibi bir metot ile alınır ve anlık olarak görüntü değiştirilerek güncellenen görüntünün kullanıcı tarafından görünmesini sağlar. Bu özellik olmasaydı kullanıcının ilgili activiteden çıkıp yeniden giriş yaptığında değişikliği görmesi gerekirdi. Bu da realtime veritabanının kullanımındaki avantajdır.

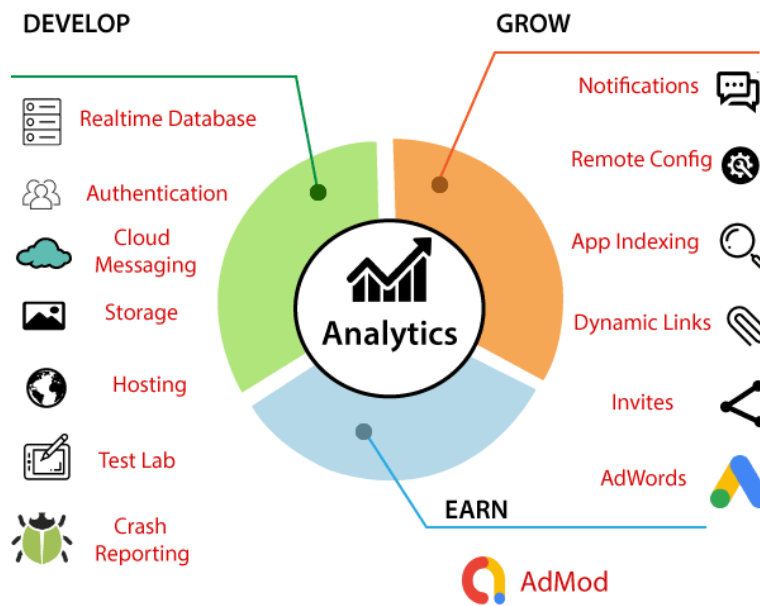
Firebase ayrıca daha önceden anlatıldığı gibi NOSQL bir veri tabandır yani hiç bir SQL sorgusu yazmadan gerekli tabloları oluşturup json formatı ile kontrol edebiliriz.

Örneğin üye giriş, çıkış ve kayıt işlemlerini authentication özelliğini kullanarak firebase in bize sunmuş olduğu servis sayesinde kısa yoldan halledebiliriz. Geliştirilen bir çok uygulama platform bağımsız olarak düşünüldüğünde aynı veriye her cihazdan erişmek ister. Uygulamalarda en çok önemli olan kaynakta veridir.

Geliştirdiğiniz uygulamada login(oturum açma) sistemi yer alıyorsa kullanıcı bilgilerine erişme, saklama ; uygulamanızı kullanan kişilerin bir takım bilgilerden haberdar olmasını istiyorsanız bildirim gönderme; uygulamanızın verilerini analiz edip, kaç crash(programın çökmesi) yaşanmış hangi cihazlarda yaşanmış vb. gibi ayrıntılı crash raporlarına ulaşma ; ayrı bir veritabanı ve sunucu desteği oluşturmadan gerçek zamanlı verilere ulaşma gibi bir çok özelliği içinde barındıran firebase platformunu daha kapsamlı bir şekilde kendi sitesinden inceleyebiliriz.

Kısaca özetlemek gerekirse

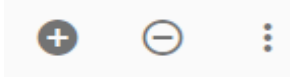
1. Firebase Realtime Database, istemciler arasında gerçek zamanlı olarak veri depolanmasını ve senkronize edilmesini sağlayan, gecikme süresi az NoSql veritabanıdır.
2. Firebase Realtime Databasede tutulan veriler tüm platformlara realtime olarak tetiklenir. Herhangi bir güncelleme, ekleme, silme işleminde db'yi dinlediğiniz methodlara tetiklenir ve bu sayede veri değişimini ekranlara doğrudan yansıtabilirsiniz.
3. Firebase Realtime Database veri barındırma ihtiyacınızı karşıladığı için bir sunucu, veritabanı ve api'ye ihtiyacınız kalmaz.
4. Kimlik doğrulaması ekleyerek erişimin güvenliğini sağlayabilirsiniz. Ayrıca veri alanlarına kurallar koyabilirsiniz örneğin: String tipindeki bir alanın maksimum uzunluğu 50 karakter olması.
5. Çevrim dışı olduğu durumlarda, Firabase Realtime Database Sdk'ları cihazların önbelleğini kullanarak depolama işlemini yapar ve bağlantı kurulduğunda veri gönderme işlemine devam edebilir.



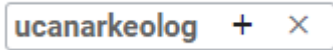
Tablo Oluřturma

Projemizde kullanmak üzere veritabanımızı oluřtururken parent child iliřkisi olacak řekilde alanlarımızı oluřturabiliriz. İlk realtime veritabanımızı oluřtururken sunucu bilgilerini girmeden sadece projeveritabanına ayırt edici bir isim vererek iřlemlerimize bařlayabiliriz.

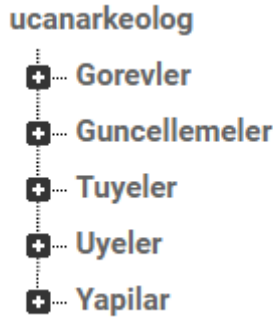
Realtime Database ekranında saę tarafta bulunan + sembolünü seęerek veritabanımızı oluřturuyoruz.



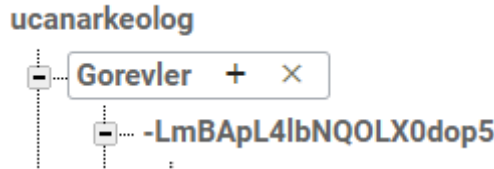
Ucandroid isminden ayırtedici olması aęısından projede kullanırken karıřıklıkları önlemek adına farklı bir isim veriyoruz. Veritabanımızın isminin yan tarafında bulunan + sembolü ile child yapısı oluřturuyoruz. Bu Child yapısının altına bařka child yapıları oluturulabilir. Böylelikle Veri tabanındaki bilgileri gruplandırabiliriz.



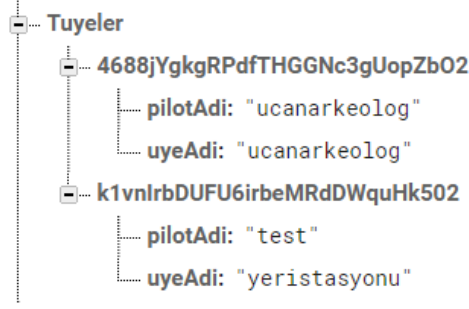
Türkçe karakter kullanmadan **Gorevler**, **Guncellemeler**, **Tuyeler**, **Uyeler**, **Yapılar** isimli ana tablolarımızı oluřturuyoruz.



Kayıt Eklme



Herhangi bir grup bařlığının yanındaki + sembolüne tıklayarak yeni bir kayıt ekleyebiliyoruz. Örneęin projemiz için **Tuyeler** kayıtları içerisindeki kayıtlar önceden giriřini yapmamız gerek alanlardır. Tuyeler bařlığında “**pilotAdi**” etiketiyle “**ucanarkeolog**” verilerini eklediğimizde realtime database ilgili tag ile veriyi eřleřtirerek kayıt yapmaktadır. İkinci kullanıcı olarak yeni bir kayıt alanına “**pilotAdi**” etiketiyle “**test**” verisini kaydediyoruz. İkinci alana “**uyeAdi**” etiketiyle “**yeristasyonu**” verisini kaydediyoruz.



Örneğin **Gorevler** başlığı altına bir görev kaydı eklemek için android programında yazılım kodları ile kayıt yapacağız. Kodlarla kayıt yapabildiğimiz için consol arabiriminde bir veri girişi yapmıyoruz.

Uçandroid Projesi İçin İhtiyaç Duyulan Tabloların Oluşturulması

Gorevler, Guncellemeler, Tuyeler, Uyeler, Yapilar isimli ana tablolarımızı oluşturuyoruz. Bu tablolar ne amaçla kullanılacaktır?

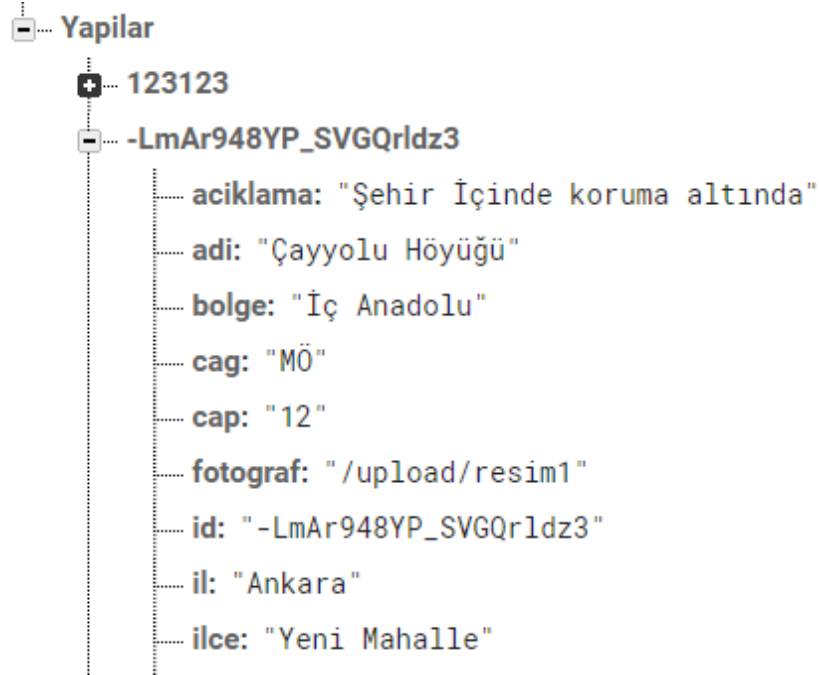
Tuyeler Tablosu

Tuyeler tablosu içerisinde bulunan kullanıcı bilgileri sisteme tanımlı farklı yetkili kullanıcıları ifade etmektedir. ucanarkeolog kullanıcısı iha üzerinde kullanılacak mobil cihazın yetkilerini belirleyen kullanıcıyı ifade eder. İHA'ya bağlı olan android telefonda çalışan program bu kullanıcı ile oturum açmaktadır. Bu kullanıcı uçuş bilgilerini anlık olarak buluta aktarırken, bulut yapısından gelen görevleri yerine getirmek için gerekli işlemleri yapar. Fotoğraf çekimi yapıp bunu Firebase üzerinde dosya olarak saklar. İkinci kullanıcı tanımı ise yeristasyonu olarak geçen pilot ve yardımcısında bulunan android telefon ile buluta giriş yapan kullanıcılardır. Bu kullanıcılar İHA dan gelen verileri anlık olarak görme yetkisine sahiptir. Yeristasyonu kullanıcısı İHA'yı kontrol etmek için görev kayıtları oluşturup ilgili kayıtları Realtime veritabanında saklayabilir.



Yapılar Tablosu

Yapılar tablosu içerisinde yer alan kayıtlar tüm görevlerde kullanılacak alanlara ait özel bilgilerin önceden oluşturulmuş kayıtlarıdır. Bu kayıtlar her bir görev için referans oluşturmaktadır. Her bir görev gerçekleştirildiğinde elde edilen verilerin birbirine karışmasını önlemek için kullanılacak bir üst yapı olacaktır. Bu üst yapı ile zaman içinde oluşacak tüm kayıtlar hızlıca elde edilebilir.



Yapılar bloğu içerisinde her bir yapı için **aciklama**, **adi**, **bölge**, **cag**, **cap**, **fotoğraf**, **id**, **il**, **ilce**, **konumX**, **konumY**, **pilot**, **rakim**, **tahribatDurumu**, **tarih**, **tayexTescilNo**, **туру**, **yerdenyukseklık ve yon** kayıtları saklanır. Bütün görev alanları kaydedilmelidir. Bu kayıtlardan en az bir tanesi proje öncesinde eklenmelidir. Böylelikle sistemde hata olmasının önüne geçmiş oluruz. Değişikliklerin olması durumunda kayıtlar güncelleştirilmelidir. Yeni görev alanları olduğunda eklenmelidir.

Uyeler Tablosu



Aktif olarak giriş yapan ve uçuşunu gerçekleştiren İHA'ların anlık verilerinin tutulduğu tablodur. İHA'da bulunan ucandroid uygulaması kendisi ile ilgili bilgileri anlık olarak bu tabloya gönderir. Yardımcı pilotun elinde bulunan Yeristasyonu olarak çalışan android programı değişiklik olduğu anda anlık olarak bu tablodaki verileri alarak kullanıcıya göstermektedir. İHA kayıt yaptığında realtime olarak buluta giriş yapmış tüm cihazlara bu bilgi aktarılmaktadır. İHA tarafından yapılan her kayıt önceki kaydın üzerine yapıldığı için anlık veriler görüntülenmeye devam eder. İHA'dan yer istasyonu arasındaki bilgi akışının protokolünün merkezinde bu tablo bulunmaktadır.

Uyeler tablosunda id, konumX, konumY, pilotAdi, rakim, ucuşModu, uyeAdi, yerdenYukseklik, yön bilgileri bulunur. İHA id si kullanılarak tüm veriler elde edilir. Koordinatlar, enlem ve boylam şeklinde konumX, konumY olarak saklanır. Rakım bilgisi, yerden yükseklik, yön bilgisi ve İHA'nın uçuş modu bilgileri saklanır. İHA'nın yetkili olduğu kullanıcı bilgisi saklanır.

Görevler Tablosu

Görevler tablosu içerisinde **fotoğraf, id, kim, kime, komut, korrdinatX, koordinatY, tarih, yapıldimi, y tarih** isimli kayıtlar barındırır. İHA ile yer istasyonu arasındaki görev tanımlarının oluşturulduğu protokolü içerir. İki taraflı olarak çalışan bu yapıda İHA'dan yapılması istenen görevler komutlar olarak tanımlanır. Bu komutların içeriği ve detayları parametre olarak tanımladığımızı düşünelim. Hangi yer istasyonunundan hangi İHAya komut gönderildiği adres alanı ve görevin gerçekleştirilmesi istenen yapının ID'si, herbir kayıt komut ve diğer parametrelerden oluşur.

ucanarkeolog

Görevler

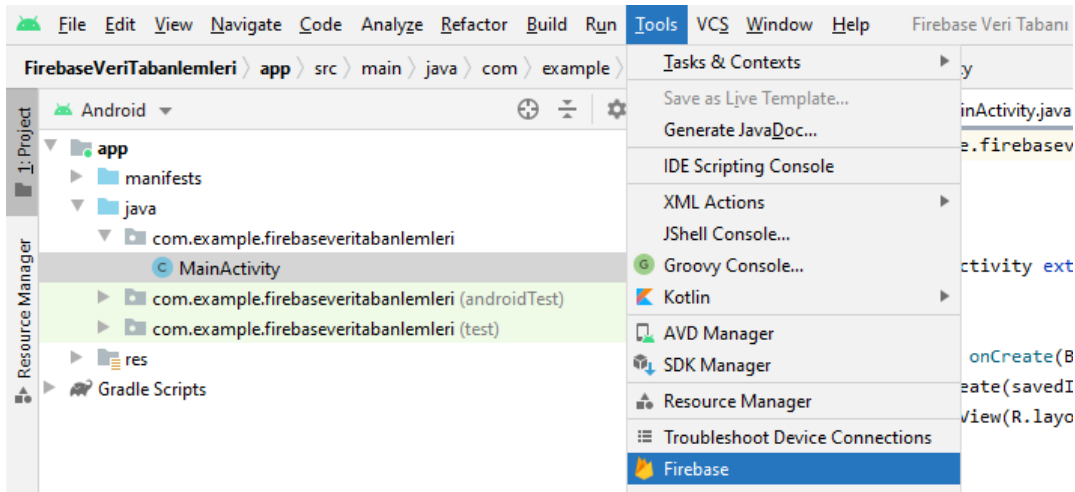
-LmBApL4lbNQOLX0dop5

```
fotoğraf: "/upload/resim1"  
id: "-LmBApL4lbNQOLX0dop5"  
kim: "Web-Admin"  
kime: "ucanarkeolog"  
komut: "FotoCek"  
koordinatX: "40.0282748"  
koordinatY: "32.4678937"
```

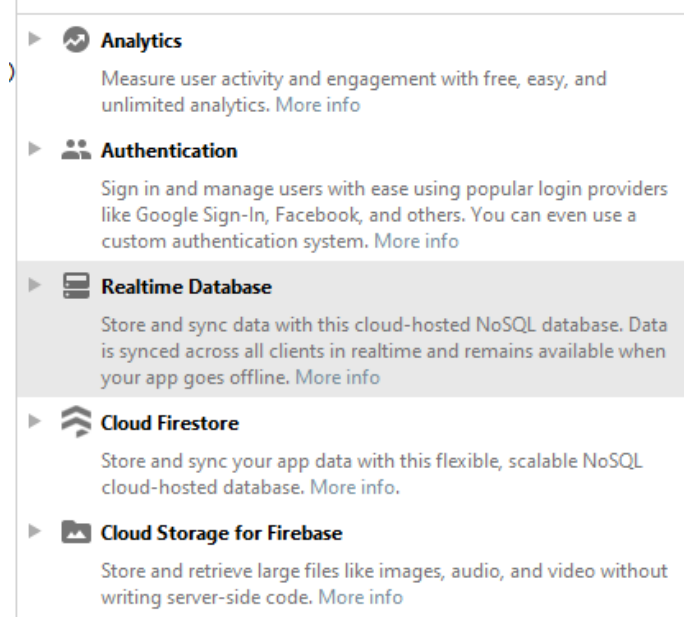
Alan Adı	Amacı	Değeri
kim	Komutun hangi yer istasyonundan geldiğini belirtir	Yeristasyonu kullanıcı adı
kime	Komutları gerçekleştirmesi istenen İHA bilgisi	İHA kullanıcı adı
id	Görevin gerçekleştirildiği yapının tanımlayıcı IDsi	Yapılar Idsi
komut	FotoCek, VideoCek gibi gerçekleştirilmesi istenen komut	string
korrdinatX	Görevin gerçekleştirilmesi istenen enlem bilgisi	float
koordinatY	Görevin gerçekleştirilmesi istenen boylam bilgisi	float
fotoğraf	Görevden elde edilen görüntü dosyasının kaydedileceği konum ve dosya adını içerir.	String yol ifadesi ve dosyaadı
tarih	Görevin kaydedildiği tarih	Gün Saat bilgisi
yapildimi	Görevler peşpeşe kayıt edilebilmektedir. Birden fazla görev geldiğinde kayıt sırasına göre görevlerin yerine getirilmesi istenir. Eğer görev yapılmışsa 1, yapılmamışsa 0 değerini alır.	1, 0
ytarih	Görevin yapıldığı tarih bilgisini kaydeder.	Gün saat bilgisi

ANDROID STUDIO İLE FIREBASE REALTIME(GERÇEK ZAMANLI) VERİTABANI BAĞLANTISI OLUŞTURMA

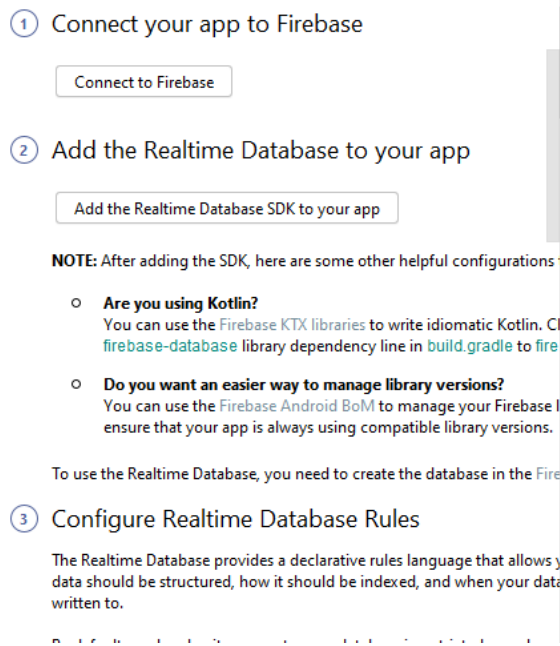
Android Firebase Veritabanı CRUD İşlemlerini oluşturmak için öncelikle android studio uygulamamızı açıp Tools menüsünden Firebase seçeneğine tıklamalıyız.



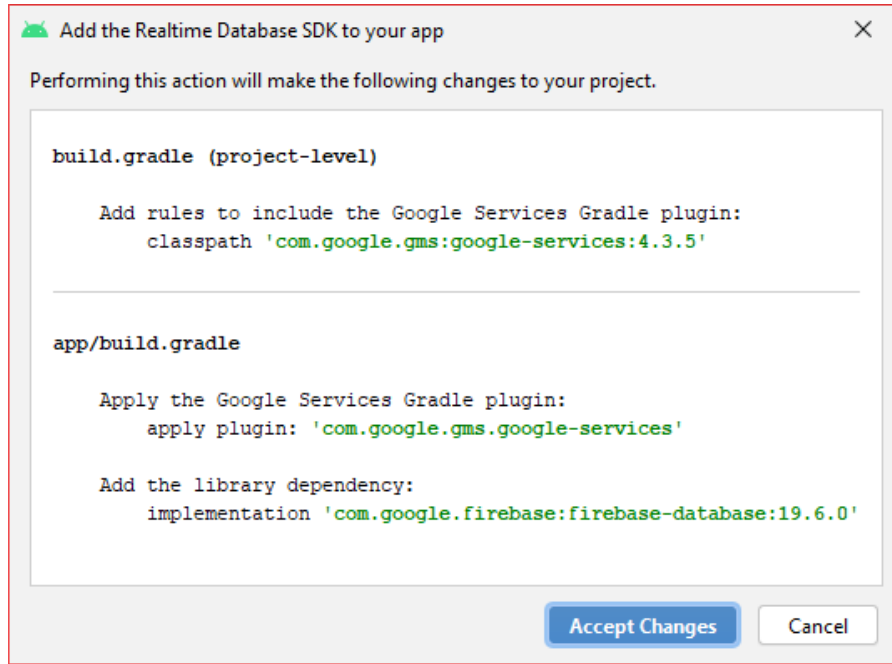
Çıkan menüden Realtime Database seçeneğine tıklıyoruz.



Daha sonra pencerede çıkan seçenekleri uyguluyoruz.

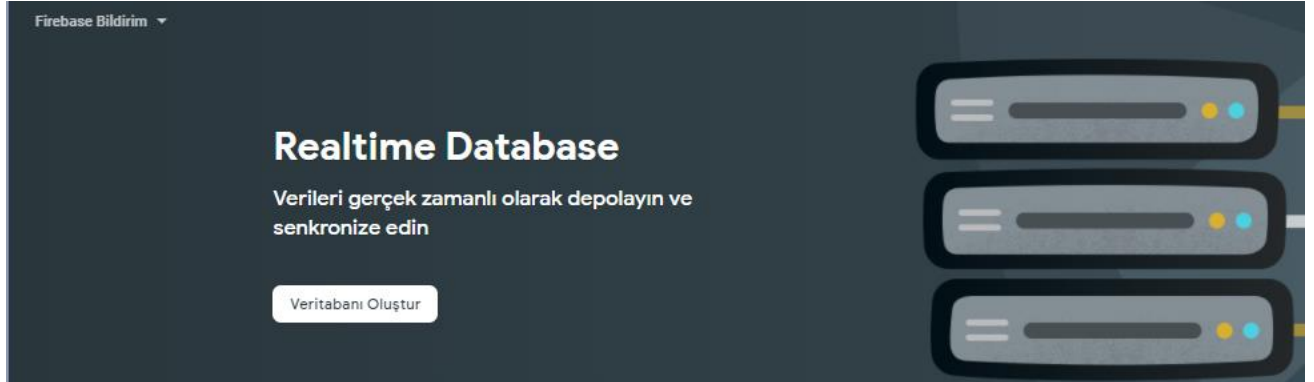


- 1- Connect to Firebase : Bu seçenek ile Firebase ile Android Studio bağlantısını gerçekleştirilir. Daha önceden Firebase konsolda bir hesabımız olması gerekiyor giriş yapılmamış ise giriş yapılması gerektiği şeklinde bir uyarı karşımıza çıkacaktır. Gmail hesabımız ile giriş işlemi yapıldıktan sonra android studio da otomatik olarak gerekli yapılandırmayı yapacaktır.
- 2- Add the Realtime Database SDK to your app)



Accepts Changes(Değişiklikleri kaydet)düğmesine tıklandığında projemize gerekli kodlar otomatik olarak eklenecektir.

Veri Tabanı Oluşturma



Veri tabanını oluştur düğmesine basıp sonrasında konum seçip Test modunda başlayıp devam edebilirsiniz.

Veritabanı kurulumu

1 Veritabanı seçenekleri
2 Güvenlik kuralları

Konum ayarınız, Realtime Database verilerinizin depolanacağı alandır.

Realtime Database konumu

Amerika Birleşik Devletleri (us-central1)

İptal İleri

Veritabanı kurulumu

1 Veritabanı seçenekleri
2 Güvenlik kuralları

Veri yapınızı tanımladıktan sonra verilerinizin güvenliğini sağlayacak kurallar yazmanız gerekir.

[Daha fazla bilgi](#)

☐ **Kilitli modda başla**
Verileriniz varsayılan olarak gizlidir. İstemci okuma/yazma erişimi yalnızca güvenlik kurallarınızda belirtilen şekilde verilir.

☒ **Test modunda başlat**
Hızlı kurulumu etkinleştirmek için verileriniz varsayılan olarak açıktır. Ancak uzun vadeli istemci okuma/yazma erişimini etkinleştirmek için 30 gün içinde güvenlik kurallarınızı güncellemeniz gerekir.

```

{
  "rules": {
    ".read": "now < 1617829200000", // 2021-4-8
    ".write": "now < 1617829200000", // 2021-4-8
  }
}

```

! Test modunun varsayılan güvenlik ayarları, veritabanı referansınıza sahip herkesin önümüzdeki 30 gün boyunca veritabanınızdaki tüm verileri görüntülemesine, düzenlemesine ve silmesine izin verir

İptal Etkinleştir

Uygulamamızı database realtime için hazır hale getirdikten sonra

Realtime Database

Veriler
Kurallar
Yedekler
Kullanım

Kuralları düzenle
İzleme kuralları

Yayıncıdan kaldırılan değişiklikler
Yayınla
Sil
Kurallar Oyun Alanı

```

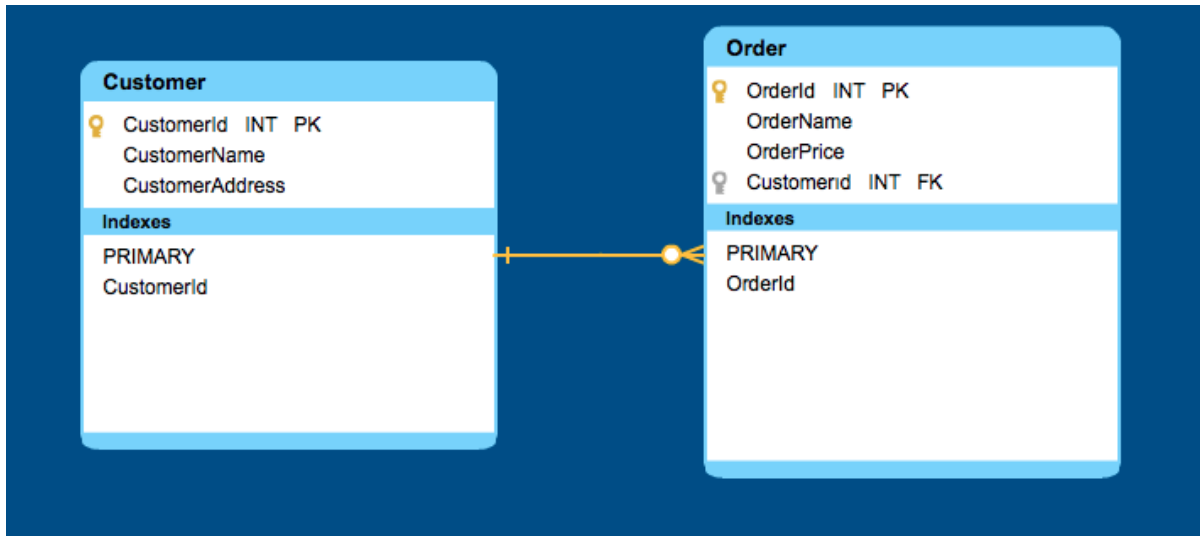
1 {
2   "rules": {
3     ".read": true,
4     ".write": true
5   }
6 }

```

Veri tabanında read(okuma) ve write(yazma) işlemlerini yapabilmemiz için Kurallar (**RULES**) yazan sekmeden **read** ve **write** işlemlerini true olarak değiştirip Yayınla(Publish) işlemi yapılır.

(Firebase de bulunan bu kurallar sayesinde kullanıcıların yetkilerini kısıtlayabilir, sadece belirli verilere erişmesini sağlayabilir ve kendi kurduğunuz yapıya göre olmasını istediğiniz kontrolleri ekleyebilirsiniz.)

Şimdi uygulamamızda kullanacağımız veritabanı yapısını oluşturalım örnek olarak Müşteri -> Sipariş ilişkisini kullanılacak. Her bir müşteri farklı farklı siparişler verebilir. Örneğin A müşterisinin; Mont aldığını düşünelim. B müşterisi de Pantolon alabilir. Yani bir müşteri birden fazla ürün sipariş verebilir. Veritabanı şeması da aşağıdaki gibidir.



Bir sonraki adımda **Customer** ve **Order** Modellerimizin kodları bulunmaktadır.

Customer.java

```
public class Customer {
    private String customerId;
    private String customerName;
    private String customerAddress;

    public Customer(){

    }

    public String getCustomerId() {
        return customerId;
    }

    public void setCustomerId(String customerId) {
        this.customerId = customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getCustomerAddress() {
        return customerAddress;
    }

    public void setCustomerAddress(String customerAddress) {
        this.customerAddress = customerAddress;
    }

    public Customer(String customerId, String customerName, String customerAddress){

        this.customerId = customerId;
        this.customerName = customerName;
        this.customerAddress = customerAddress;
    }
}
```

Order.Java

```
public class Order {  
  
    private String orderId;  
    private String orderName;  
    private int orderPrice;  
  
    public Order(){  
  
    }  
  
    public Order(String orderId, String orderName, int orderPrice) {  
        this.orderId = orderId;  
        this.orderName = orderName;  
        this.orderPrice = orderPrice;  
    }  
  
    public String getOrderId() {  
        return orderId;  
    }  
  
    public void setOrderId(String orderId) {  
        this.orderId = orderId;  
    }  
  
    public String getOrderName() {  
        return orderName;  
    }  
  
    public void setOrderName(String orderName) {  
        this.orderName = orderName;  
    }  
  
    public int getOrderPrice() {  
        return orderPrice;  
    }  
  
    public void setOrderPrice(int orderPrice) {  
        this.orderPrice = orderPrice;  
    }  
  
}
```

Modellerimizi tanımladıktan sonra MainActivity sınıfımızı inceleyecek olursak uygulama ilk açıldığında onStart methodunda Database reference üzerinden DataSnapshot ile child lara ulaşıyoruz ve ilgili kayıtlar var ise Customer modeline atarak RecyclerView in adapterına set ediyoruz. Böylece realtime database de Customer ile ilgili kaç kayıt var ise hepsini getirip ekranda göstermiş oluyoruz.

Veritabanında herhangi bir değişikliği dinlemek için `addValueEventListener()` yapısını kullanıyoruz. Uygulamamızda göreceğiniz gibi veritabanında bir değişiklik yaptığımızda örneğin `update`, `delete` anında o verinin değiştiğini göreceksiniz. Firebase console dan da girerek örneğin; müşterinin adını değiştirdiğimizde android uygulamamızda yer alan müşterinin adında otomatik olarak değiştiğini göreceksiniz.

MainActivity de customer yapısını göreceğimiz için `onCreate` methodunda reference ulaşıyoruz.

```
DatabaseReference databaseReferenceCustomers = FirebaseDatabase.getInstance().getReference("customers");
```

Kayıt işlemi içinde aşağıdaki yapıyı oluşturuyoruz. FloatingButtona basıldığı anda ;her kayıt birbirinden farklı olması için bir id alıyoruz (`customerId`) kullanıcı uygulamada müşteri adını soyadını girdiğinde ve adres spinnerdan ilgili adresi seçtiğinde customer objesini oluşturuyoruz. Database reference in `child` methoduna id yi atıp `setValue` ile customer objemizi kaydediyoruz. Uygulama içinden kayıt işlemini bu şekilde yapıyoruz ayrıca firebase platformundan kendi oluşturduğumuz bir json dosyasını import ederek de kayıtları oluşturabiliriz.

```
String id = databaseReferenceCustomers.push().getKey();
BreakIterator editText;
Customer customer = new Customer(id, editText.getText().toString(), spinner.getSelectedItem().toString());
databaseReferenceCustomers.child(id).setValue(customer);
```

NOT: Bu kısımda bulut yapısı ile Android studio arasında bağlantıların oluşturulması ve Android studio platformunda veri tabanı oluşturma işlemlerinden bahsedildi. Android studio üzerinde işlemler yapmak için gerekli adımlar Mobil Uygulama Geliştirme modülünde açıklanmıştır.

Uçandroid Android Programı Veri Tabanı işlemleri

Burada bulut yapısının uygulama üzerinde tanımlanması ve kullanılması ile ilgili temel aşamalardan bahsedilecektir. Diğer açıklamaları **Mobil Uygulama Geliştirme** modülünde bulabilirsiniz.

Basla.Java üzerinde firebase kütüphanelerinin yüklenmesi için aşağıdaki kütüphaneleri kullanırız.

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
```

Veri tabanı işlemlerini yapmak için kullanacağımız referans değişkenleri tanımlanıyor.

```
FirebaseDatabase database;  
DatabaseReference dbref;  
private Uyeler uyeler = new Uyeler();
```

Veritabanını kullanarak bir görev kaydı yapmak için hazırlanan metot ile bulut yapısına kayıt nasıl yapıldığı görülmektedir. `myRef2 = database.getReference("Gorevler").child(key)`; ile veritabanına referans eden `myRef2` değişkeni ile yeni bir Key push ediliyor. Oluşan veri alanına `gorev` classındaki bilgiler `myRef2.setValue(gorev)`; komutu ile gönderiliyor.

```
private void gorevKaydet() {  
    Gorev gorev = new Gorev();  
    FirebaseDatabase database = FirebaseDatabase.getInstance();  
    DatabaseReference myRef = database.getReference("Gorevler");  
    String key = myRef.push().getKey();  
    DatabaseReference myRef2 = database.getReference("Gorevler").child(key);  
    gorev.setId("1");  
    gorev.setTarih("11.08.2019");  
    gorev.setKim("Web-Admin");  
    gorev.setKime("ucanarkeolog");  
    gorev.setKomut("FotoCek");  
    gorev.setYapildimi("0");  
    gorev.setFotograf("/upload/resim1");  
    gorev.setYTarih("11.08.2019");  
    gorev.setKoordinatX("40.0282748");  
    gorev.setKoordinatY("32.4678937");  
    //myRef.child(key).child("Id").setValue("1");  
  
    myRef2.setValue(gorev);  
  
    //Toast.makeText(MainActivity.this, "Gorev kaydedildi",  
    Toast.LENGTH_LONG).show();  
}
```

Okunmuş olan bir görevi class içerisinde saklayarak ilgili classı bulutta kaydetmek için aynı işlemler yapılarak aynı isimde iki fonksiyon oluşturuyoruz. Bu da daha esnek kullanım imkanı sağlamaktadır. Nesneye dayalı programlamanın özelliklerinden overloading kullanmış oluyoruz.

```
private void gorevKaydet(Gorev grv) {  
    FirebaseDatabase database = FirebaseDatabase.getInstance();  
    DatabaseReference myRef = database.getReference("Gorevler");  
    String key = grv.getId();  
    DatabaseReference myRef2 = database.getReference("Gorevler").child(key);  
  
    myRef2.setValue(grv);  
}
```

```
//Toast.makeText(MainActivity.this, "Gorev nesnesi kaydedildi",
Toast.LENGTH_LONG).show();
}
```

Bulut Veritabanından veri alma ve görev listesine eklemek için **dbRef** ile referans edilen veritabanı sunucusundan gelen bir **Listener** olayı ile gelen veriyi almış oluyoruz. Daha sonra alınan verinin anlamlı bir şekilde saklanması için **gorevAl** metodunu kullanıyoruz.

```
dbref.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        try {
            gorevAl(dataSnapshot);
        } catch (CameraAccessException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});
```

```
private void gorevAl(DataSnapshot dataSnapshot) throws CameraAccessException {

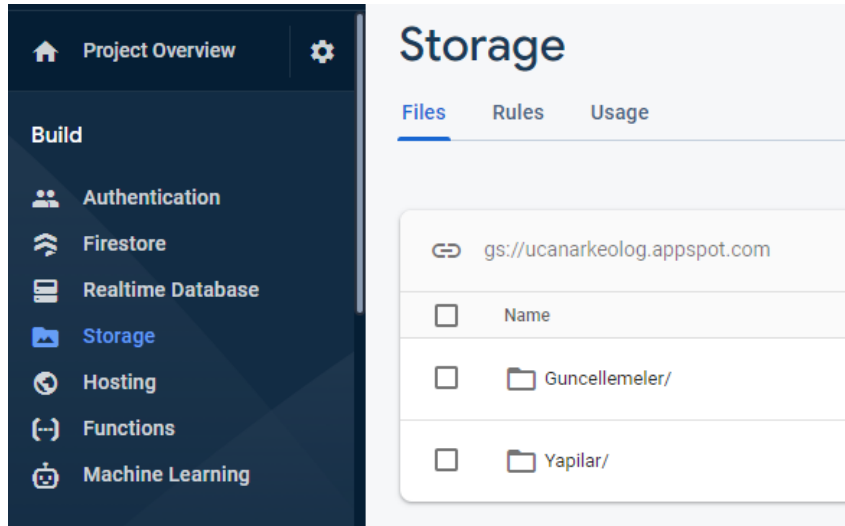
    gorevler.clear(); /// gorevleri temizlemek icin
    for(DataSnapshot ds : dataSnapshot.getChildren())
    {
        String grvId=ds.getKey();
        Gorev grv = new Gorev();
        //grv.setId(ds.getValue(Gorev.class).getId());
        grv.setId(ds.getKey());
        grv.setTarih(ds.getValue(Gorev.class).getTarih());
        grv.setKim(ds.getValue(Gorev.class).getKim());
        grv.setKime(ds.getValue(Gorev.class).getKime());
        grv.setKomut(ds.getValue(Gorev.class).getKomut());
        grv.setYapildimi(ds.getValue(Gorev.class).getYapildimi());
        grv.setFotograf(ds.getValue(Gorev.class).getFotograf());
        grv.setYTarih(ds.getValue(Gorev.class).getYTarih()); //Bugunki tarihi
        kaydet
        grv.setKoordinatX(ds.getValue(Gorev.class).getKoordinatX());
        grv.setKoordinatY(ds.getValue(Gorev.class).getKoordinatY());
        grv.setRakim(ds.getValue(Gorev.class).getRakim());

        if(grv.getYapildimi().matches("0"))
        {
            if(grv.getKomut().matches(getString(R.string.kmtFotoCek)) ||
            grv.getKomut().matches(getString(R.string.kmtGitFotoCek)) ||
            grv.getKomut().matches(getString(R.string.kmtBaglanSunucu)) ||
            grv.getKomut().matches(getString(R.string.kmtBaglanDron)))
            {
                gorevler.add(grv);
                Toast.makeText(Basla.this, grv.getKomut()+" "+gorevler.size(),
```

```
Toast.LENGTH_LONG).show();  
    }  
}  
  
gorevCalistir();  
}
```

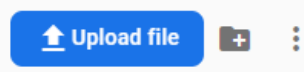
FİREBASE STORAGE DOSYA SAKLAMA SERVİSİ

Firestore storage hizmeti bir sanal disk olarak çalışmaktadır. Firestore Cloud Storage diğer platformlardan erişim imkanlarını sağlamak için componentler sağlar. Dosyaları dizin yapıları içerisinde saklamaya, okuma, yazma, değişiklik yapma ve silme işlemlerinin yapılmasına imkan tanır. Klasörler üzerinde işlemler yapmaya olanak sağlar.






Firestore dosya sunucusu hizmetini açtığımızda Files sekmesinde dosyaların tutulduğu alan ve dosya ve klasörler görünmektedir. Dosya üzerindeki işlemleri buradan yapabildiğimiz gibi otantikasyonu gerçekleştiren tüm cihazlardan yapılabiliriz.

Upload File butonu ile yüklemeleri yapabiliriz.



Rules sekmesi içerisinde sunucuda dosya işlemlerine yönelik kuralların ve koşulların yazılabildiği alan bulunmaktadır. Usage ise tüm kullanıcıların yaptığı işlemlerin kullanım oranlarını izleyebildiğimiz bir alan sunmaktadır.

Storage			
Files Rules Usage			
gs://ucanarkeolog.appspot.com > Yapılar			
<input type="checkbox"/>	Name	Size	Type
<input type="checkbox"/>	 Ucanark1566152581.jpg	291.81 KB	image/jpeg
<input type="checkbox"/>	 Ucanark1566152875.jpg	293.78 KB	image/jpeg
<input type="checkbox"/>	 Ucanark_20190818_224905.jpg	289.38 KB	image/jpeg

Dosya servisini açtıktan sonra kullanımı için bazı tanımlamalar yapılmaktadır. Bu tanımlamalardan sonra sunucuya referans olarak kullanılan değişkenler ile tek komutla işlemlerin yapılması sağlanabilmektedir.

Uçandroid Android Programı Dosya işlemleri

Assistant

Firebase > Storage

Upload and download a file with Firebase Storage

Firebase Storage provides secure file uploads and downloads for your Firebase apps, regardless of network quality. You can use it to store images, audio, video, or other user-generated content.

[Launch in browser](#)

- 1 Connect your app to Firebase**

on Connected
- 2 Add Firebase Storage to your app**

on Dependencies set up correctly
- 3 Create a reference**

Declare a [StorageReference](#) and initialize it in the `onCreate` method.

```
private StorageReference mStorageRef;

mStorageRef = FirebaseStorage.getInstance().getReference();
```
- 4 Upload a file**

The simplest way to upload to your storage bucket is by uploading a local file, such as photos and videos from the camera, using the `putFile()` method. You can also upload raw data using `putBytes()` or from an `InputStream` using `putStream()`.

```
Uri file = Uri.fromFile(new File("path/to/images/rivers.jpg"));
StorageReference riversRef = storageRef.child("images/rivers.jpg");

riversRef.putFile(file)
    .addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            // Get a URL to the uploaded content
            Uri downloadUrl = taskSnapshot.getDownloadUrl();
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
            // Handle unsuccessful uploads
            // ...
        }
    });
```

2 Event Log
Gradle Console

Android Studio da **Firestore->Storage->Upload and download a file with Firestore Storage** seçeneğini tıklayıp ; gerekli adımları izledikten sonra ilgili kütüphaneleri indirmesini bekliyoruz. Projemizde bu işlemleri aşağıdaki kodları kullanarak gerçekleştiriyoruz.

Uçandroid üzerinde Basla.java içerisinde

```
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
```

Dosya sistemi değişkenleri tanımlanıyor

```
FirebaseStorage storage = FirebaseStorage.getInstance();
StorageReference storageRef;
```

Void init içerisinde bulut depolama pointer i hazırlanıyor. storageRef ile bulut dosya sistemine erişim sağlanacaktır.

```
public void init() {
    storageRef = storage.getReference();
}
```

Fotocek metodu içerisine baktığımızda kamera nesnesinden alınan görüntü işlendikten sonra fotoDosyasi değişkenine **ucandroid'e** ilgili görevin yapıldığı tarih bilgisi dosya ismine eklenerek saklanmak üzere hazırlanır. Çekilmiş olan fotoğraf telefon üzerine değil Firestore tarafında çalışmakta olan storage bulut yapısı içinde sakalanacaktır.

```
public void FotoCek() throws CameraAccessException {
    Log.d("FOTO", "FotoCek: 1");
    if (kameraDevice==null){
        return;
    }
    CameraManager kmrMan = (CameraManager)
    getSystemService(Context.CAMERA_SERVICE);
    CameraCharacteristics kmrChar =
    kmrMan.getCameraCharacteristics(kameraDevice.getId());
    Size[] jpegSizes=null;
    jpegSizes=kmrChar.get(CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP).getOutputSizes(ImageFormat.JPEG);
    int width=640;
    int height=480;
    if (jpegSizes!=null && jpegSizes.length>0)
    {
        width=jpegSizes[0].getWidth();
    }
}
```

```

        height=jpegSizes[0].getHeight();
    }
    final ImageReader fotoReader = ImageReader.newInstance(width, height,
ImageFormat.JPEG, 1);
    List<Surface> outputSurfaces = new ArrayList<>(2);
    outputSurfaces.add(fotoReader.getSurface());
    outputSurfaces.add(new Surface(kameraView.getSurfaceTexture()));
    final CaptureRequest.Builder captureBuilder =
kameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_STILL_CAPTURE);
    captureBuilder.addTarget(fotoReader.getSurface());
    captureBuilder.set(CaptureRequest.CONTROL_MODE,
CameraMetadata.CONTROL_MODE_AUTO);

    int rotation = getWindowManager().getDefaultDisplay().getRotation();
    captureBuilder.set(CaptureRequest.JPEG_ORIENTATION,
ORIENTATIONS.get(rotation));
    // Dosyaya kaydet
    Long tsLong = System.currentTimeMillis()/1000;
    String ts=tsLong.toString();

    fotoDosyasi = new File (Environment.getExternalStorageDirectory()+
"/Ucandroid" + ts + ".jpg");
    ImageReader.OnImageAvailableListener fotoReaderListener = new
ImageReader.OnImageAvailableListener() {
        @Override
        public void onImageAvailable(ImageReader imageReader) {
            Image image=null;
            image = fotoReader.acquireLatestImage();
            ByteBuffer buffer = image.getPlanes()[0].getBuffer();
            byte[] bytes = new byte[buffer.capacity()];
            buffer.get(bytes);
            try {
                FotoKaydet(bytes);
                Toast.makeText(Basla.this, fotoDosyasi.getAbsolutePath(),
Toast.LENGTH_SHORT).show();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                if (image!=null)
                {
                    image.close();
                }
            }
        }
    };

    fotoReader.setOnImageAvailableListener(fotoReaderListener, kmrBgHandler);

    final CameraCaptureSession.CaptureCallback captureListener = new
CameraCaptureSession.CaptureCallback() {
        @Override
        public void onCaptureCompleted(CameraCaptureSession session,
CaptureRequest request, TotalCaptureResult result) {
            super.onCaptureCompleted(session, request, result);
            Konus("Fotoğraf çekildi");
            Log.d("FOTO", "FotoCek: Fotoğraf Kaydedildi");
            try {
                createCameraPreview();
            } catch (CameraAccessException e) {

```

```
        e.printStackTrace();
    }
}
};
```

Foto Kaydet metodu ile byte dizidi olarak bilgileri alınan fotoğraf dosya olarak aktarılır. OutputStream içerisine fotoDosyasi bilgileri ile alınan veriler write() metodu ile gönderilir. İşlem tamamlandığında outputStream bağlantısı kapatılmalıdır.

```
private void FotoKaydet(byte[] bytes) throws IOException {
    OutputStream outputStream = null;
    outputStream = new FileOutputStream(fotoDosyasi);
    outputStream.write(bytes);
    outputStream.close();
}
```

Kaynaklar

1. <https://www.salesforce.com/ca/cloud-computing/#:~:text=more%20about%20SaaS-,PaaS,servers%20or%20special%20testing%20environments>
2. <https://www.logosoft360.com/Post2.aspx?t=1>
3. <https://gelecegiyazanlar.turkcell.com.tr/blog/firebase-nedir-avantajlari-nelerdir>
4. <https://devnot.com/2017/web-ve-mobil-uygulamalar-icin-firebase/>
5. <https://benimyazilimblogum.blogspot.com/2019/02/hafta-2-firebase-nedir-ve-ne-ise-yarar.html>
6. <https://ceaksan.com/tr/firebase-nedir-nasil-kullanilir>
7. <https://www.mobilhanem.com/android-firebase-crud-islemleri/>



**UÇANDROİD,
MAKİNE ÖĞRENMESİ İLE OTONOM İHA SİSTEMİ
ÜRETİMİ VE PROJE TABANLI ÜRETİM EĞİTİM
MODÜLLERİNİN GELİŞTİRİLMESİ**

Bu proje T.C. Ankara Kalkınma Ajansı tarafından finanse edilmektedir.