

Jetpack Compose ile Android Uygulama Geliştirme Kursu

İnternet Tabanlı İşlemler

Kasım ADALAN

Elektronik ve Haberleşme Mühendisi

Android - IOS Developer and Trainer

Eğitim İçeriği

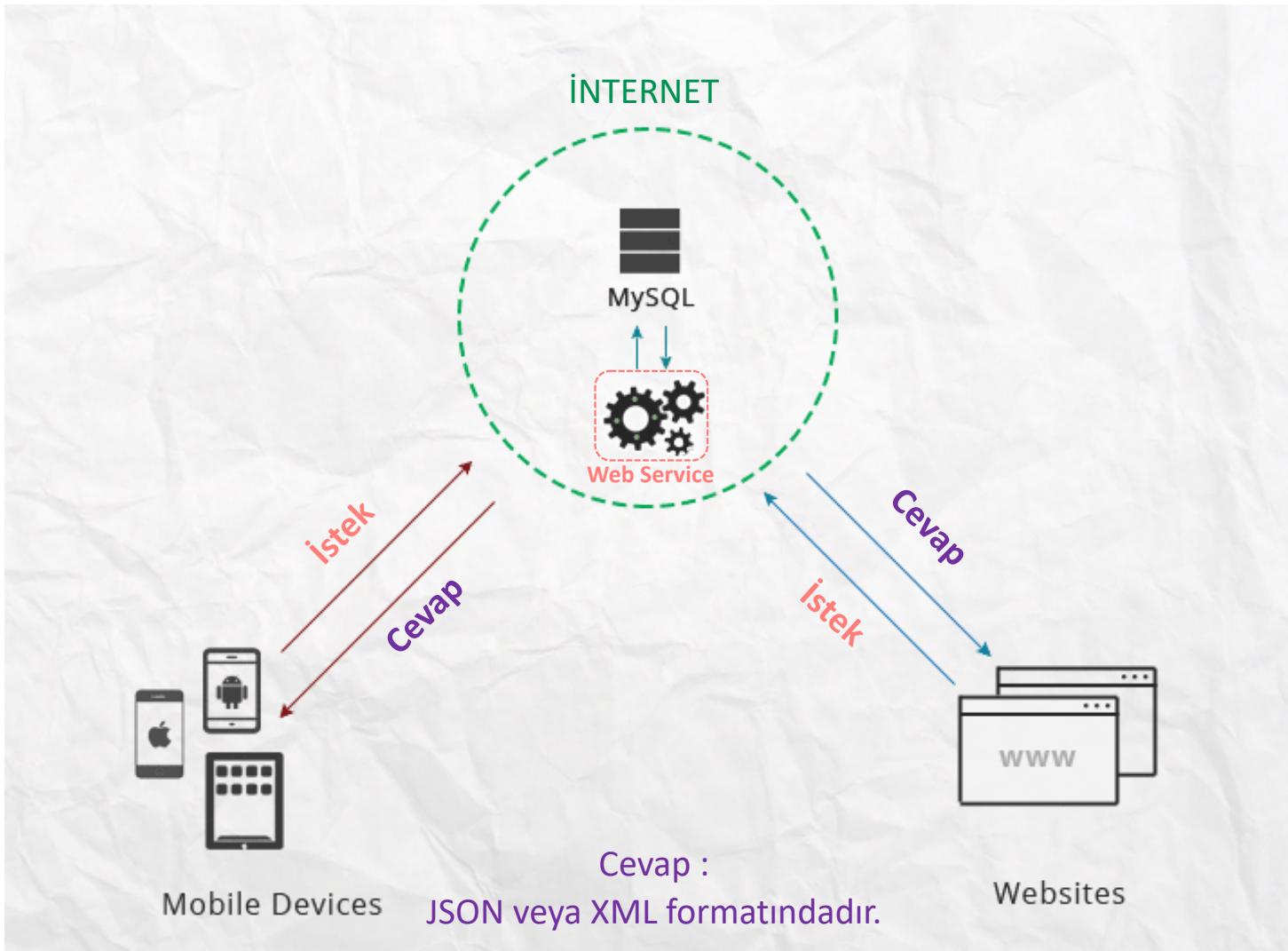
- Restful API
- Retrofit Kütüphanesi
- Firebase Realtime Database
- Coil - Resim İşlemi

RESTful Web Services

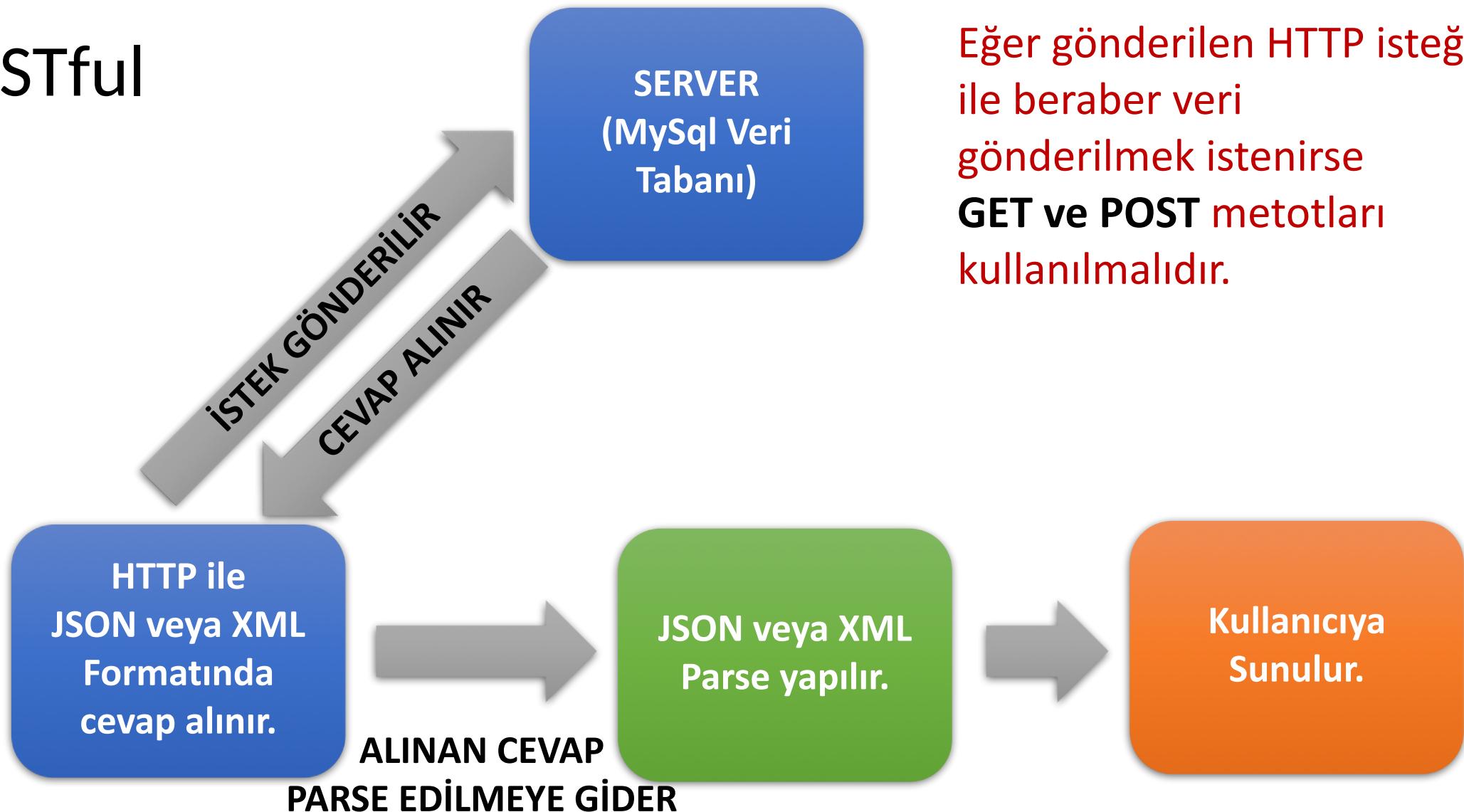
- REST : Representational State Transfer
- Web servise , Bilgileri basit bir formatta transfer etmek için kullanılır.
- XML ve JSON en çok kullanılan formattır.
- Web tabanlı olduğu için internet izni istenmektedir.

`<uses-permission android:name="android.permission.INTERNET"/>`

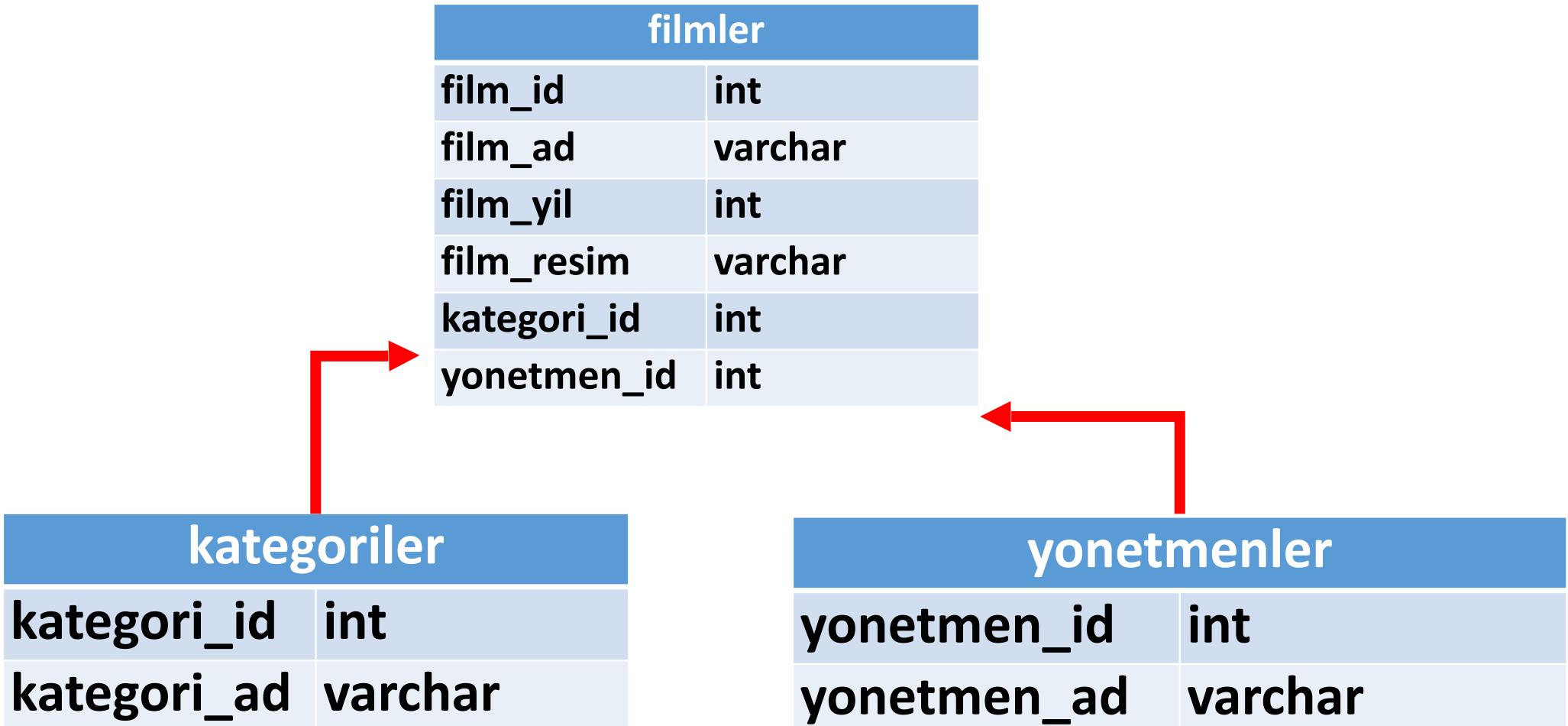
RESTful Çalışma Mimarisi



RESTful



MySQL Veritabanı Yapısı



Filmler

```
CREATE TABLE filmler (
    film_id int(11) primary key auto_increment,
    film_ad varchar(100) not null,
    film_yil int(11) not null,
    film_resim varchar(100) not null,
    kategori_id int(11) not null ,
    yonetmen_id int(11) not null,
    CONSTRAINT fk_yonetmen FOREIGN KEY (yonetmen_id)
        REFERENCES yonetmenler(yonetmen_id),
    CONSTRAINT fk_kategori FOREIGN KEY (kategori_id)
        REFERENCES kategori(kategori_id)
);
```

Yönetmenler

```
CREATE TABLE yonetmenler (
    yonetmen_id int(11) primary key auto_increment,
    yonetmen_ad varchar(100) not null
);
```

Kategoriler

```
CREATE TABLE kategoriler (
    kategori_id int(11) primary key auto_increment,
    kategori_ad varchar(100) not null
);
```

Örn:

```
CREATE TABLE `kisiler` (
  `kisi_id` int(11) PRIMARY KEY AUTO_INCREMENT,
  `kisi_ad` varchar(100) NOT NULL,
  `kisi_tel` varchar(100) NOT NULL
) ;
```

PHP

- Android uygulama ile Veritabanının bulunduğu server arasındaki bağlantıyı php ile yapabiliriz.

db_config.php

```
<?php  
define( 'DB_USER' , "u139539474_root" );  
define( 'DB_PASSWORD' , "123456" );  
define( 'DB_DATABASE' , "u139539474_movie" );  
define( 'DB_SERVER' , "mysql.hostinger.web.tr" );  
?>
```

```

<?php
// array for JSON response
$response = array();

//DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
require_once __DIR__ . '/db_config.php';

// Bağlantı oluşturuluyor.
$baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

// Bağlantı kontrolü yapılır.
if (!$baglanti) {
    die("Hatalı bağlantı : " . mysqli_connect_error());
}

$sqlsorgu = "SELECT * FROM kisiler";
$result = mysqli_query($baglanti, $sqlsorgu);

// result kontrolü yap
if (mysqli_num_rows($result) > 0) {

    $response["kisiler"] = array();

    while ($row = mysqli_fetch_assoc($result)) {
        // temp user array
        $kisiler = array();

        $kisiler["kisi_id"] = $row["kisi_id"];
        $kisiler["kisi_ad"] = $row["kisi_ad"];
        $kisiler["kisi_tel"] = $row["kisi_tel"];

        // push single product into final response array
        array_push($response["kisiler"], $kisiler);
    }
    // success
    $response["success"] = 1;
}

// echoing JSON response
echo json_encode($response);

```

tum_kisiler.php

Çalıştırmak istediğimiz PHP kod



```

} else {
    // no products found
    $response["success"] = 0;
    $response["message"] = "No data found";

    // echo no users JSON
    echo json_encode($response);
}

//bağlantı koparılır.
mysqli_close($baglanti);
?>

```

tum_kisiler_arama.php

```
<?php  
  
$response = array();  
  
if (isset($_POST['kisi_ad'])) {  
    $kisi_ad = $_POST['kisi_ad'];  
  
    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.  
    require_once __DIR__ . '/db_config.php';  
  
    // Bağlantı oluşturuluyor.  
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);  
  
    // Bağlantı kontrolü yapılır.  
    if (!$baglanti) {  
        die("Hatalı bağlantı : " . mysqli_connect_error());  
    }  
  
    $sqlsorgu = "SELECT * FROM kisiler WHERE kisiler.kisi_ad like '%$kisi_ad%'";  
    $result = mysqli_query($baglanti, $sqlsorgu);  
  
    if (mysqli_num_rows($result) > 0) {  
  
        $response["kisiler"] = array();  
  
        while ($row = mysqli_fetch_assoc($result)) {  
  
            $kisiler = array();  
  
            $kisiler["kisi_id"] = $row["kisi_id"];  
            $kisiler["kisi_ad"] = $row["kisi_ad"];  
            $kisiler["kisi_tel"] = $row["kisi_tel"];  
  
            array_push($response["kisiler"], $kisiler);  
        }  
  
        $response["success"] = 1;  
        echo json_encode($response);  
    }  
    //bağlantı koparılır.  
    mysqli_close($baglanti);  
}
```

Çalıştırmak
istediğimiz PHP
kod



```
} else {  
    $response["success"] = 0;  
    $response["message"] = "Required field(s) is missing";  
    echo json_encode($response);  
}  
?>
```

```

<?php
$response = array();

if (isset($_POST['kisi_id'])) {
    $kisi_id = $_POST['kisi_id'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }
    $sqlsorgu = "DELETE FROM kisiler WHERE kisiler.kisi_id = $kisi_id";

    if (mysqli_query($baglanti, $sqlsorgu)) {

        $response["success"] = 1;
        $response["message"] = "successfully ";
        echo json_encode($response);
    } else {

        $response["success"] = 0;
        $response["message"] = "No product found";
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>

```

delete_kisiler.php

Çalıştırmak istediğimiz
PHP kod

```
<?php
$response = array();

if (isset($_POST['kisi_ad']) && isset($_POST['kisi_tel'])) {
    $kisi_ad = $_POST['kisi_ad'];
    $kisi_tel = $_POST['kisi_tel'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }

    $sqlsorgu = "INSERT INTO kisiler (kisi_ad,kisi_tel) VALUES ('$kisi_ad','$kisi_tel')";

    if (mysqli_query($baglanti, $sqlsorgu)) {
        $response["success"] = 1;
        $response["message"] = "successfully ";
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "No product found";
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

insert_kisiler.php

Çalıştırmak istediğimiz PHP kod

```

<?php

$response = array();

if (isset($_POST['kisi_id']) && isset($_POST['kisi_ad']) && isset($_POST['kisi_tel'])) {
    $kisi_id = $_POST['kisi_id'];
    $kisi_ad = $_POST['kisi_ad'];
    $kisi_tel = $_POST['kisi_tel'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }

    $sqlsorgu = "UPDATE kisiler SET kisiler.kisi_ad = '$kisi_ad',kisiler.kisi_tel = '$kisi_tel' WHERE
                kisiler.kisi_id = $kisi_id  ";

    if (mysqli_query($baglanti, $sqlsorgu)) {
        $response["success"] = 1;
        $response["message"] = "successfully ";
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "No product found";
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>

```

update_kisiler.php

Çalıştırmak istediğimiz PHP kod

Retrofit

Retrofit Kütüphanesinin kullanımı

Retrofit

A type-safe HTTP client for Android and Java

	One Discussion	Dashboard (7 requests)	25 Discussions
AsyncTask	941 ms	4,539 ms	13,957 ms
Volley	560 ms	2,202 ms	4,275 ms
Retrofit	312 ms	889 ms	1,059 ms

Retrofit Kütüphane kullanım aşaması

1. Gerekli kütüphaneler gradle ile yüklenecek,internet izni alınacak.
2. JSON formatında alınacak bilgi POJO yapısına dönüştürülecek ve Java sınıfları oluşturulacak.
3. JSON verilerini almak için bir adet interface oluşturulacak.
4. Retrofit nesnesi alınacak.
5. Alınan retrofit nesnesi ile oluşturduğumuz interface bağlanacak.
6. Çalıştırmak istediğimiz interface'in içinde bulunan metod çağrıılır ve çalıştırılır.

1.Kütüphane yüklenmesi ve izin alınması

```
<uses-permission android:name="android.permission.INTERNET"/>
```

// Retrofit

```
implementation 'com.squareup.retrofit2:retrofit:2.1.0'
```

// JSON Parse

```
implementation 'com.google.code.gson:gson:2.6.1'
```

```
implementation 'com.squareup.retrofit2:converter-gson:2.1.0'
```

Güncelleme

- **http** ile başlayan web servisler varsayılan olarak çalışmamaktadır.
- **https** ile başlayan web servislerde sorun yaşamamaktadır.
- Bu sorunu çözmek için aşağıdaki ifadeyi manifest dosyasına eklemeniz gereklidir.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.volleykullanimi">

    <uses-permission android:name="android.permission.INTERNET"/>

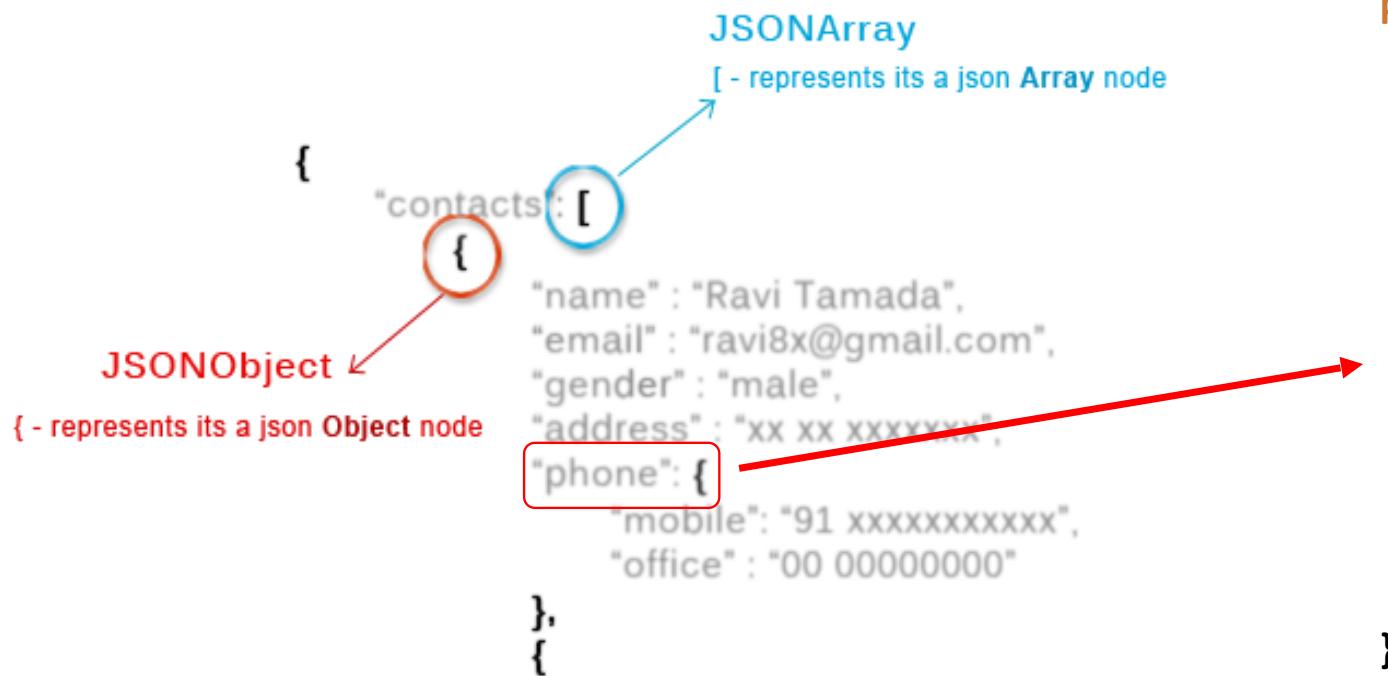
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="VolleyKullanimi"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

JSON Parse İşlemi

- Gelen verilere göre model sınıflar oluşturulmalıdır.



```
public class Contacts {  
  
    private String id;  
    private String name;  
    private String email;  
    private String gender;  
    private String address;  
    private Phone phone;  
  
    public Contacts() {  
    }  
}
```

JSON

```
{  
    "contacts": [  
        {  
            "id": "c200",  
            "name": "Ravi Tamada",  
            "email": "ravi@gmail.com",  
            "address": "xx-xx-xxxx,x - street, x - country",  
            "gender": "male",  
            "phone": {  
                "mobile": "+91 0000000000",  
                "home": "00 000000",  
                "office": "00 000000"  
            }  
        },  
        {  
            "id": "c201",  
            "name": "Johnny Depp",  
            "email": "johnny_depp@gmail.com",  
            "address": "xx-xx-xxxx,x - street, x - country",  
            "gender": "male",  
            "phone": {  
                "mobile": "+91 0000000000",  
                "home": "00 000000",  
                "office": "00 000000"  
            }  
        },  
        .  
        .  
        .  
        .  
    ]  
}
```

JSON CEVAPLARI İÇİN

SINIF DÖNÜŞÜMLERİ

JSON Nesnesi için Sınıf Oluşturma

```
data class Kisiler(@SerializedName( value: "kisi_id")  
    @Expose  
    var kisi_id:Int,  
    @SerializedName( value: "kisi_ad")  
    @Expose  
    var kisi_ad:String,  
    @SerializedName( value: "kisi_tel")  
    @Expose  
    var kisi_tel:String) {  
}
```

```
{  
    "kisiler": [  
        {  
            "kisi_id": "376",  
            "kisi_ad": "Ahmet",  
            "kisi_tel": "5348564412"  
        },  
        {  
            "kisi_id": "379",  
            "kisi_ad": "Zeynep",  
            "kisi_tel": "05375483214"  
        },  
        {  
        },  
        {  
        },  
        {  
        },  
        {  
        }  
    ],  
    "success": 1  
}
```

JSON Cevabı için Sınıf Oluşturma

```
data class KisilerCevap(@SerializedName( value: "kisiler")  
    @Expose  
    var kisiler:List<Kisiler>,  
    @SerializedName( value: "success")  
    @Expose  
    var success:Int) {  
}
```

```
{  
    "kisiler": [  
        {  
            "kisi_id": "376",  
            "kisi_ad": "Ahmet",  
            "kisi_tel": "5348564412"  
        },  
        {  
            "kisi_id": "379",  
            "kisi_ad": "Zeynep",  
            "kisi_tel": "05375483214"  
        },  
        {  
        },  
        {  
        },  
        {  
        },  
        {  
        }  
    ],  
    "success": 1  
}
```

JSON Cevabı için Sınıf Oluşturma

```
|data class CRUDCevap(@SerializedName(value: "success")  
|    @Expose  
|    var success:Int,  
|    @SerializedName(value: "message")  
|    @Expose  
|    var message:String) {  
|  
|}
```

The code defines a data class `CRUDCevap` with two properties: `success` (Int type) and `message` (String type). The `success` property is annotated with `@SerializedName(value: "success")` and `@Expose`. The `message` property is also annotated with `@SerializedName(value: "message")` and `@Expose`. Red arrows point from these annotations to the corresponding keys in the JSON object shown on the right.

```
{ "success": 0,  
  "message": "Required field(s) is missing" }
```

3. Interface oluşturma

```
interface KisilerDaoInterface {  
    @GET( value: "kisiler/tum_kisiler.php")  
    fun tumKisiler() : Call<KisilerCevap>  
  
    @POST( value: "kisiler/tum_kisiler_arama.php")  
    @FormUrlEncoded  
    fun kisiAra(@Field( value: "kisi_ad") kisi_ad:String) : Call<KisilerCevap>  
  
    @POST( value: "kisiler/delete_kisiler.php")  
    @FormUrlEncoded  
    fun kisiSil(@Field( value: "kisi_id") kisi_id:Int) : Call<CRUDCevap>  
  
    @POST( value: "kisiler/insert_kisiler.php")  
    @FormUrlEncoded  
    fun kisiEkle(@Field( value: "kisi_ad") kisi_ad:String  
                ,@Field( value: "kisi_tel") kisi_tel:String) : Call<CRUDCevap>  
  
    @POST( value: "kisiler/update_kisiler.php")  
    @FormUrlEncoded  
    fun kisiGuncelle(@Field( value: "kisi_id") kisi_id:Int  
                    ,@Field( value: "kisi_ad") kisi_ad:String  
                    ,@Field( value: "kisi_tel") kisi_tel:String) : Call<CRUDCevap>  
}
```

Geri dönüş değerini POJO model sınıfı ile aynı olmalıdır.

Yapmak istediğimiz işlemin metodunu yazıyoruz ve muhtemel geri dönüş değerini belirtiyoruz eğer gerekiyorsa veride gönderebiliyoruz.

4. Retrofit Nesnesi Alınır

```
class RetrofitClient {  
    companion object {  
        fun getClient(baseUrl: String): Retrofit {  
            return Retrofit.Builder()  
                .baseUrl(baseUrl)  
                .addConverterFactory(GsonConverterFactory.create())  
                .build()  
        }  
    }  
}
```

Dışarıdan alınan temel url ile retrofit nesnesi alınır.

5. Retrofit nesnesiyle interface'i bağlama

```
class ApiUtils {  
    companion object{  
        val BASE_URL = "http://kasimadalan.pe.hu/"  
  
        fun getKisilerDaoInterface(): KisilerDaoInterface {  
            return RetrofitClient.getClient(BASE_URL).create(KisilerDaoInterface::class.java)  
        }  
    }  
}
```

Interface içerisindeki metodları her kullanmak istediğimizde bu metod çalıştırılacak.

Not : BASE_URL ana path olmalı yani bu yoldan sonra /kisiler/tum_kisiler.php geliyorsa bu yol interface içerisinde belirtilmelidir.Ana domain burda yer alamalı geri kalan uzanti interface içerisinde yer almalıdır.

6. Interface metodunun çağrılmaması

```
@Composable
fun Sayfa() {
    LaunchedEffect(key1: true){ this: CoroutineScope
        tumKisiler()
    }
}

fun tumKisiler(){
    val kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()

    kisilerdaoInterface.tumKisiler().enqueue(object : Callback<KisilerCevap>{
        override fun onResponse(call: Call<KisilerCevap>, response: Response<KisilerCevap>) {
            val liste = response.body().kisiler

            for (k in liste){
                Log.e(tag: "*****", msg: "*****")
                Log.e(tag: "Kişi id",k.kisi_id.toString())
                Log.e(tag: "Kişi ad",k.kisi_ad)
                Log.e(tag: "Kişi tel",k.kisi_tel)
            }
        }
        override fun onFailure(call: Call<KisilerCevap>?, t: Throwable?) {}
    })
}
```

Get Metodu kullanımı (Sadece veri almak için)

```

<?php
// array for JSON response
$response = array();

//DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
require_once __DIR__ . '/db_config.php';

// Bağlantı oluşturuluyor.
$baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

// Bağlantı kontrolü yapılır.
if (!$baglanti) {
    die("Hatalı bağlantı : " . mysqli_connect_error());
}

$sqlsorgu = "SELECT * FROM kisiler";
$result = mysqli_query($baglanti, $sqlsorgu);

// result kontrolü yap
if (mysqli_num_rows($result) > 0) {

    $response["kisiler"] = array();

    while ($row = mysqli_fetch_assoc($result)) {
        // temp user array
        $kisiler = array();

        $kisiler["kisi_id"] = $row["kisi_id"];
        $kisiler["kisi_ad"] = $row["kisi_ad"];
        $kisiler["kisi_tel"] = $row["kisi_tel"];

        // push single product into final response array
        array_push($response["kisiler"], $kisiler);
    }
    // success
    $response["success"] = 1;

    // echoing JSON response
    echo json_encode($response);
}

```

tum_kisiler.php

```

{
    "kisiler": [
        {
            "kisi_id": "376",
            "kisi_ad": "Ahmet",
            "kisi_tel": "5348564412"
        },
        {
            "kisi_id": "379",
            "kisi_ad": "Zeynep",
            "kisi_tel": "05375483214"
        },
        {
            "kisi_id": "380",
            "kisi_ad": "Mehmet",
            "kisi_tel": "05375483215"
        },
        {
            "kisi_id": "381",
            "kisi_ad": "Ayşe",
            "kisi_tel": "05375483216"
        },
        {
            "kisi_id": "382",
            "kisi_ad": "Mustafa",
            "kisi_tel": "05375483217"
        }
    ],
    "success": 1
}

// no products found
$response["success"] = 0;
$response["message"] = "No data found";

// echo no users JSON
echo json_encode($response);
}

//bağlantı koparılır.
mysqli_close($baglanti);
?>

```

```
@GET( value: "kisiler/tum_kisiler.php")
fun tumKisiler(): Call<KisilerCevap>
```

```
fun tumKisiler(){
    val kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()

    kisilerdaoInterface.tumKisiler().enqueue(object : Callback<KisilerCevap>{
        override fun onResponse(call: Call<KisilerCevap>, response: Response<KisilerCevap>) {
            val liste = response.body().kisiler
                Response nesnesinin
                body metoduyla
                KisilerCevap içeresine
                erişebiliyoruz

            for (k in liste){
                Log.e( tag: "*****", msg: "*****")
                Log.e( tag: "Kişi id", k.kisi_id.toString())
                Log.e( tag: "Kişi ad", k.kisi_ad)
                Log.e( tag: "Kişi tel", k.kisi_tel)
            }
        }
        override fun onFailure(call: Call<KisilerCevap>?, t: Throwable?) {}
    })
}
```

Veri Gönderip – Veri Okuma

```

<?php

$response = array();

if (isset($_POST['kisi_ad'])) {
    $kisi_ad = $_POST['kisi_ad'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }

    $sqlsorgu = "SELECT * FROM kisiler WHERE kisiler.kisi_ad like '%$kisi_ad%'";
    $result = mysqli_query($baglanti, $sqlsorgu);

    if (mysqli_num_rows($result) > 0) {

        $response["kisiler"] = array();

        while ($row = mysqli_fetch_assoc($result)) {

            $kisiler = array();

            $kisiler["kisi_id"] = $row["kisi_id"];
            $kisiler["kisi_ad"] = $row["kisi_ad"];
            $kisiler["kisi_tel"] = $row["kisi_tel"];

            array_push($response["kisiler"], $kisiler);
        }

        $response["success"] = 1;
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
}

```

tum_kisiler_arama.php

```

{
    "kisiler": [
        {
            "kisi_id": "376",
            "kisi_ad": "Ahmet",
            "kisi_tel": "5348564412"
        },
        {
            "kisi_id": "379",
            "kisi_ad": "Zeynep",
            "kisi_tel": "05375483214"
        },
        {
            "kisi_id": "380",
            "kisi_ad": "Mehmet",
            "kisi_tel": "05375483215"
        },
        {
            "kisi_id": "381",
            "kisi_ad": "Ayşe",
            "kisi_tel": "05375483216"
        },
        {
            "kisi_id": "382",
            "kisi_ad": "Mustafa",
            "kisi_tel": "05375483217"
        },
        {
            "kisi_id": "383",
            "kisi_ad": "Fatma",
            "kisi_tel": "05375483218"
        }
    ],
    "success": 1
}

```



```

} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>

```

```
@POST( value: "kisiler/tum_kisiler_arama.php")
@FormUrlEncoded
fun kisiAra(@Field( value: "kisi_ad") kisi_ad:String) : Call<KisilerCevap>
```

```
fun ara(){
    val kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()

    kisilerdaoInterface.kisiAra( kisi_ad: "z").enqueue(object : Callback<KisilerCevap>{
        override fun onResponse(call: Call<KisilerCevap>, response: Response<KisilerCevap>) {
            val liste = response.body().kisiler

            for (k in liste){
                Log.e( tag: "*****", msg: "*****")
                Log.e( tag: "Kişi id",k.kisi_id.toString())
                Log.e( tag: "Kişi ad",k.kisi_ad)
                Log.e( tag: "Kişi tel",k.kisi_tel)
            }
        }
        override fun onFailure(call: Call<KisilerCevap>?, t: Throwable?) {}
    })
}
```

Post metodu kullanma (Veri gönderme)

Veri gönderip veri alma

```
@POST( value: "kisiler/delete_kisiler.php")
@FormUrlEncoded
fun kisiSil(@Field( value: "kisi_id") kisi_id: Int): Call<CRUDCevap>
```

@POST Annotation'ı ile hangi url uzantısı kullanılacağı belirleniyor.

@POST Annotation'ı veri gönderme amacıyla kullanılıyor.

@FormUrlEncoded Annotation'ı gönderilen verinin utf-8 vb. encode'lara uyarlıyor.

@Field Annotation'ı url ile gönderilecek veriyi tanımlıyor.

Sonuç : http://kasimadalan.pe.hu/kisiler/delete_kisiler.php/kisi_id=2

Post metodu türevleri

```

<?php
$response = array();

if (isset($_POST['kisi_id'])) {
    $kisi_id = $_POST['kisi_id'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }
    $sqlsorgu = "DELETE FROM kisiler WHERE kisiler.kisi_id = $kisi_id";

    if (mysqli_query($baglanti, $sqlsorgu)) {

        $response["success"] = 1;
        $response["message"] = "successfully ";
        echo json_encode($response);
    } else {

        $response["success"] = 0;
        $response["message"] = "No product found";
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>

```

delete_kisiler.php

Çalıştırmak istediğimiz
PHP kod

Delete (Silme)

```
@POST( value: "kisiler/delete_kisiler.php")
@FormUrlEncoded
fun kisiSil(@Field( value: "kisi_id") kisi_id:Int) : Call<CRUDCevap>
```

```
fun sil(){
    val kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()

    kisilerdaoInterface.kisiSil( kisi_id: 2703).enqueue(object : Callback<CRUDCevap>{
        override fun onResponse(call: Call<CRUDCevap>, response: Response<CRUDCevap>) {
            val mesaj = response.body().message
            val basari = response.body().success
            Log.e(mesaj,basari.toString())
        }
        override fun onFailure(call: Call<CRUDCevap>?, t: Throwable?) {}
    })
}
```

```
<?php
$response = array();

if (isset($_POST['kisi_ad']) && isset($_POST['kisi_tel'])) {
    $kisi_ad = $_POST['kisi_ad'];
    $kisi_tel = $_POST['kisi_tel'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }

    $sqlsorgu = "INSERT INTO kisiler (kisi_ad,kisi_tel) VALUES ('$kisi_ad','$kisi_tel')";

    if (mysqli_query($baglanti, $sqlsorgu)) {
        $response["success"] = 1;
        $response["message"] = "successfully ";
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "No product found";
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>
```

insert_kisiler.php

Çalıştırmak istediğimiz PHP kod

Insert (Kayıt)

```
@POST( value: "kisiler/insert_kisiler.php")
@FormUrlEncoded
fun kisiEkle(@Field( value: "kisi_ad") kisi_ad:String
            ,@Field( value: "kisi_tel") kisi_tel:String) : Call<CRUDCevap>
```

```
fun ekle(){
    val kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()

    kisilerdaoInterface.kisiEkle( kisi_ad: "jet test ad", kisi_tel: "jet test tel").enqueue(object : Callback<CRUDCevap>{
        override fun onResponse(call: Call<CRUDCevap>, response: Response<CRUDCevap>) {
            val mesaj = response.body().message
            val basari = response.body().success
            Log.e(mesaj,basari.toString())
        }
        override fun onFailure(call: Call<CRUDCevap>?, t: Throwable?) {}
    })
}
```

```

<?php

$response = array();

if (isset($_POST['kisi_id']) && isset($_POST['kisi_ad']) && isset($_POST['kisi_tel'])) {
    $kisi_id = $_POST['kisi_id'];
    $kisi_ad = $_POST['kisi_ad'];
    $kisi_tel = $_POST['kisi_tel'];

    //DB_SERVER,DB_USER,DB_PASSWORD,DB_DATABASE değişkenleri alınır.
    require_once __DIR__ . '/db_config.php';

    // Bağlantı oluşturuluyor.
    $baglanti = mysqli_connect(DB_SERVER, DB_USER, DB_PASSWORD, DB_DATABASE);

    // Bağlantı kontrolü yapılır.
    if (!$baglanti) {
        die("Hatalı bağlantı : " . mysqli_connect_error());
    }

    $sqlsorgu = "UPDATE kisiler SET kisiler.kisi_ad = '$kisi_ad',kisiler.kisi_tel = '$kisi_tel' WHERE
                kisiler.kisi_id = $kisi_id  ";

    if (mysqli_query($baglanti, $sqlsorgu)) {
        $response["success"] = 1;
        $response["message"] = "successfully ";
        echo json_encode($response);
    } else {
        $response["success"] = 0;
        $response["message"] = "No product found";
        echo json_encode($response);
    }
    //bağlantı koparılır.
    mysqli_close($baglanti);
} else {
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
    echo json_encode($response);
}
?>

```

update_kisiler.php

Çalıştırmak istediğimiz PHP kod

Update (Güncelleme)

```
@POST( value: "kisiler/update_kisiler.php")
@FormUrlEncoded
fun kisiGuncelle(@Field( value: "kisi_id") kisi_id:Int
                ,@Field( value: "kisi_ad") kisi_ad:String
                ,@Field( value: "kisi_tel") kisi_tel:String) : Call<CRUDCevap>
```

```
fun guncelle(){
    val kisilerdaoInterface = ApiUtils.getKisilerDaoInterface()

    kisilerdaoInterface.kisiGuncelle( kisi_id: 2709, kisi_ad: "test ad", kisi_tel: "test tel").enqueue(object : Callback<CRUDCevap>{
        override fun onResponse(call: Call<CRUDCevap>, response: Response<CRUDCevap>) {
            val mesaj = response.body().message
            val basari = response.body().success
            Log.e(mesaj,basari.toString())
        }
        override fun onFailure(call: Call<CRUDCevap>?, t: Throwable?) {}
    })
}
```

Firebase Realtime Database



Firebase Realtime Database



- Firebase gerçek zamanlı çalışan bir veri tabanıdır.
- Veri tabanı üzerinde oluşan değişiklikleri anlık olarak projelere yansıtır.
- No sql bir veri tabanıdır. Klasik sql sorguları geçerli değildir.
- Veri kayıt işlemini primary key olmadan yapmaktadır.
- Verileri json yapısında tutmaktadır.
- Web servis yazmanız gereklidir.

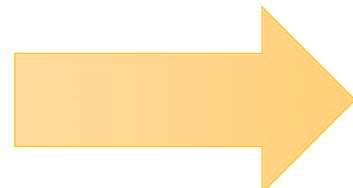
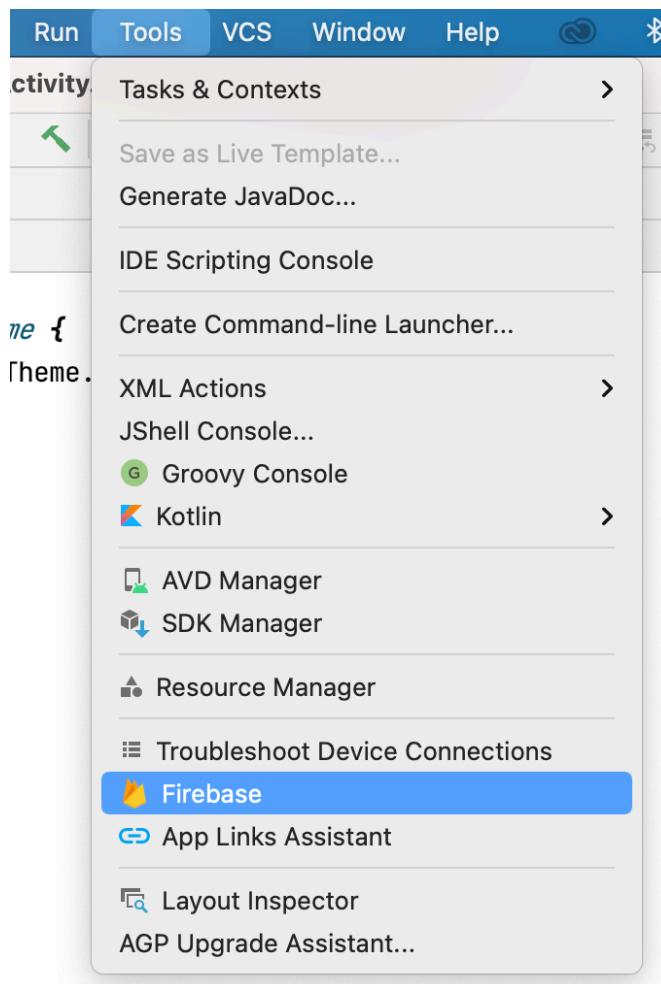
Firebase üzerinde proje oluşturmalıyız.

The screenshot shows the Firebase console homepage. At the top left is the Firebase logo. At the top right are links for "Go to docs", a bell icon for notifications, and a user profile picture. On the left, there's a "Recent projects" section featuring a woman in a yellow sweater interacting with a large screen displaying a blue and white geometric pattern. Below this is a button with a plus sign and the text "Add project". To the right of this are five project cards:

- Egitim** (egitim-d167c) - iOS platform
- Deneme** (deneme-a6a63) - Android platform
- Projem** (projem-eaa40)
- BildirimProjem** (bildirimprojem) - TV platform
- EgitimVeritabani** (egitimveritabani-2b6fc) - TV platform

At the bottom right of the page is the number 48.

Android Studio içerisinde Firebase'a ulaşmak



The right-hand sidebar of the Android Studio interface is shown, with the 'Assistant' tab selected. The 'Firebase' tab is also present and selected. The Firebase section contains the following content:

- Firebase**: A brief introduction stating "Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. Learn more".
- Sign in and manage users**: A section about using popular login providers like Google Sign-In, Facebook, and others, with a "More info" link.
- Realtime Database**: A section about storing and syncing data with a cloud-hosted NoSQL database, with a "More info" link and two "Get started" links: one for Realtime Database and one for Realtime Database [KOTLIN].
- Cloud Firestore**: A section about storing and syncing app data with a flexible, scalable NoSQL cloud-hosted database, with a "More info" link.
- Cloud Storage for Firebase**: A section about storing and retrieving large files like images, audio, and video without writing server-side code, with a "More info" link.
- Cloud Functions for Firebase**: A section about automatically running backend code in response to events triggered by Firebase features and HTTPS requests, with a "More info" link.

On the far right, other tabs like Gradle, Flutter Outline, Flutter Inspector, Flutter Performance, Assistant, and Emulator are visible.

Kasım ADALAN

[← Firebase](#) > Realtime Database

Get started with Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as a tree of JSON objects and synchronized in realtime to every connected client.

[Launch in browser](#)

- ### 1 Connect your app to Firebase

[Connect to Firebase](#)
- ### 2 Add the Realtime Database to your app

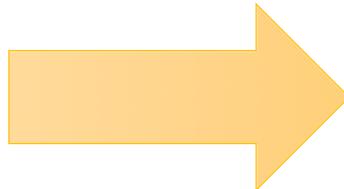
[Add the Realtime Database SDK to your app](#)

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

 - Do you want an easier way to manage library versions?** You can use the [Firebase Android BoM](#) to manage your library versions and ensure that your app is always using compatible versions.

To use the Realtime Database, you need to create the database in the [Realtime Database console](#).
- ### 3 Configure Realtime Database Rules

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be updated, and who has access to it.



Web browser üzerinden bağlanmak istenen proje seçilir.

Welcome to Firebase! To connect your Android app, choose an existing project or create a new one

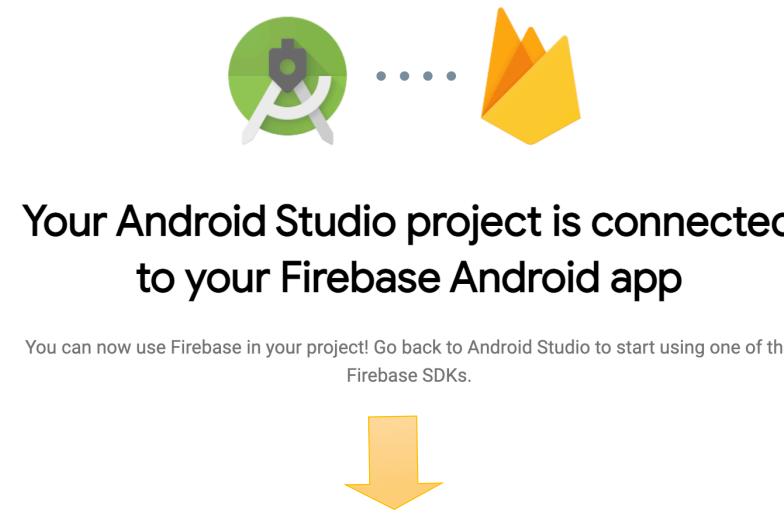
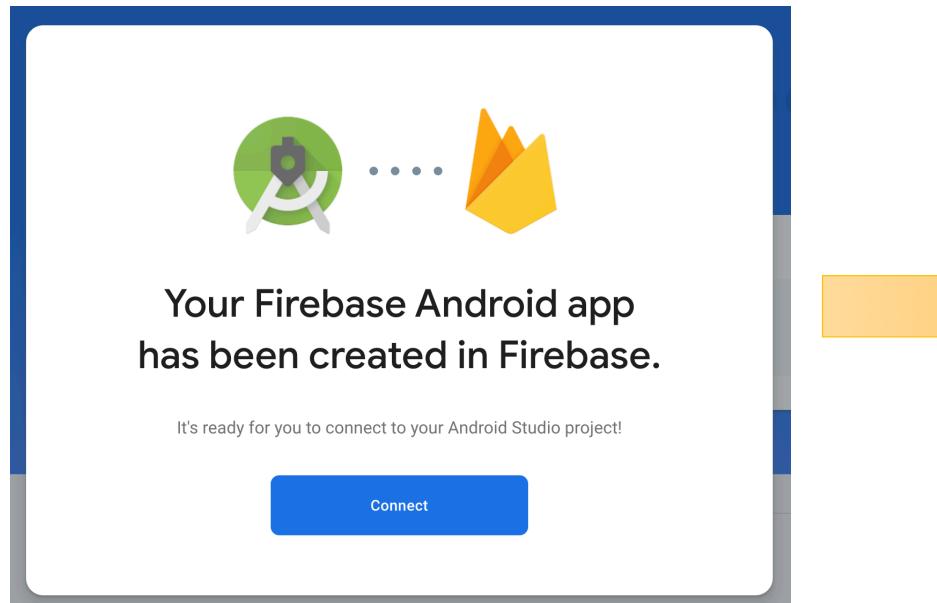
Choose a project to continue

Recent projects

Egitim
egitim-d167c

Deneme
deneme-a6a63

+ Add project



Get started with Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON objects and synchronized in realtime to every connected client.

[Launch in browser](#)

① Connect your app to Firebase

✓ Connected

② Add the Realtime Database to your app

[Add the Realtime Database SDK to your app](#)

Get started with Realtime Database

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON objects and synchronized in realtime to every connected client.

[Launch in browser](#)

① Connect your app to Firebase

✓ Connected

② Add the Realtime Database to your app

[Add the Realtime Database SDK to your app](#)

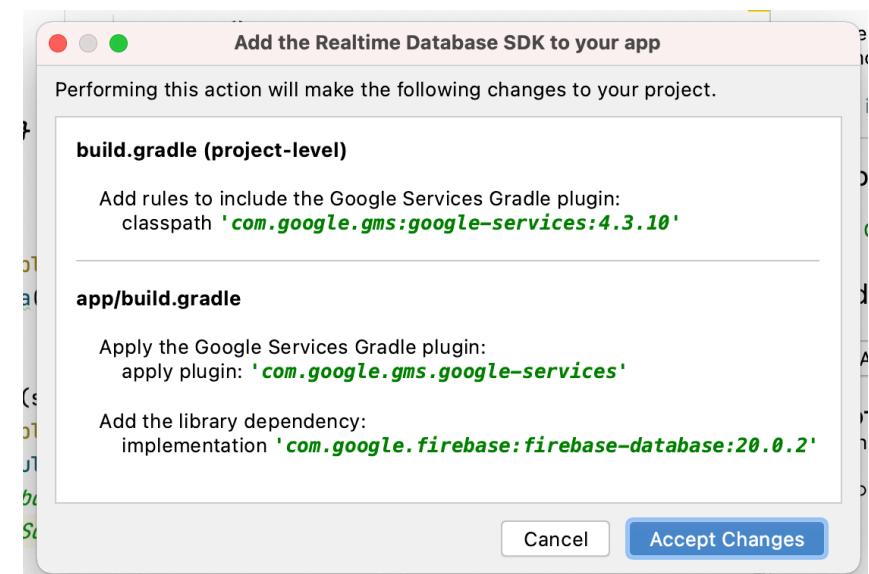
NOTE: After adding the SDK, here are some other helpful considerations to consider:

- **Do you want an easier way to manage library versions?**
You can use the [Firebase Android BoM](#) to manage your library versions and ensure that your app is always using compatible versions.

To use the Realtime Database, you need to create the database in the [Realtime Database console](#).

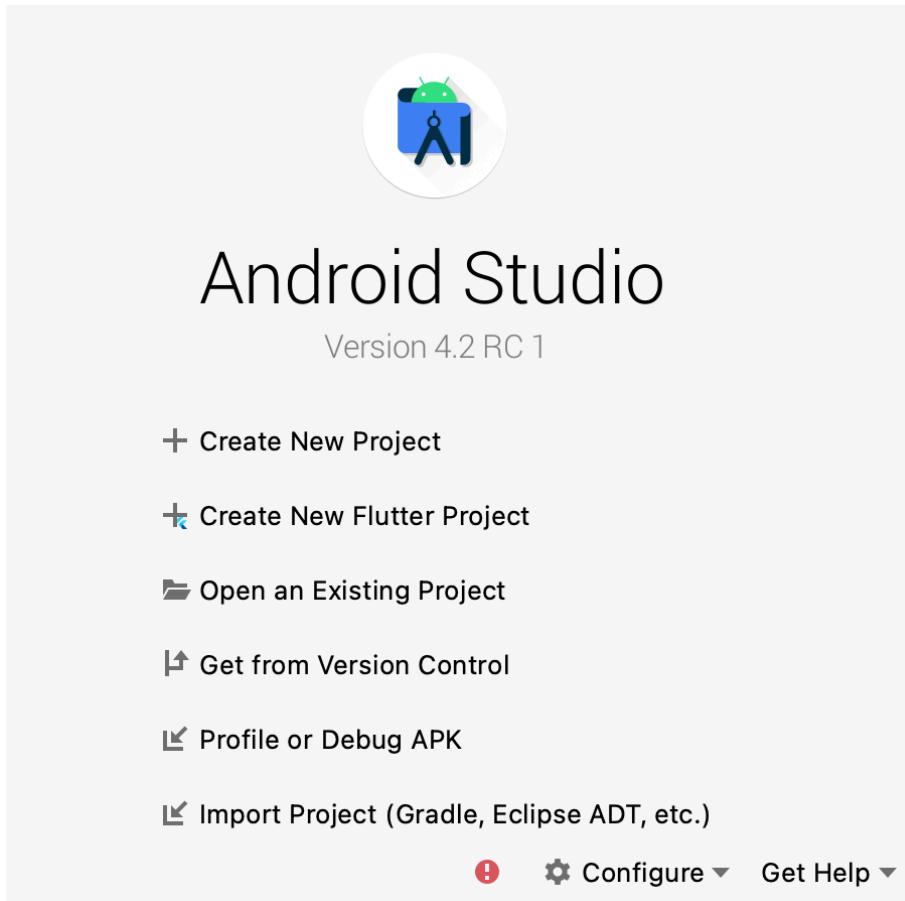
③ Configure Realtime Database Rules

The Realtime Database provides a declarative rules language that lets you define how your data should be structured, how it should be secured, and how it can be read from and written to.



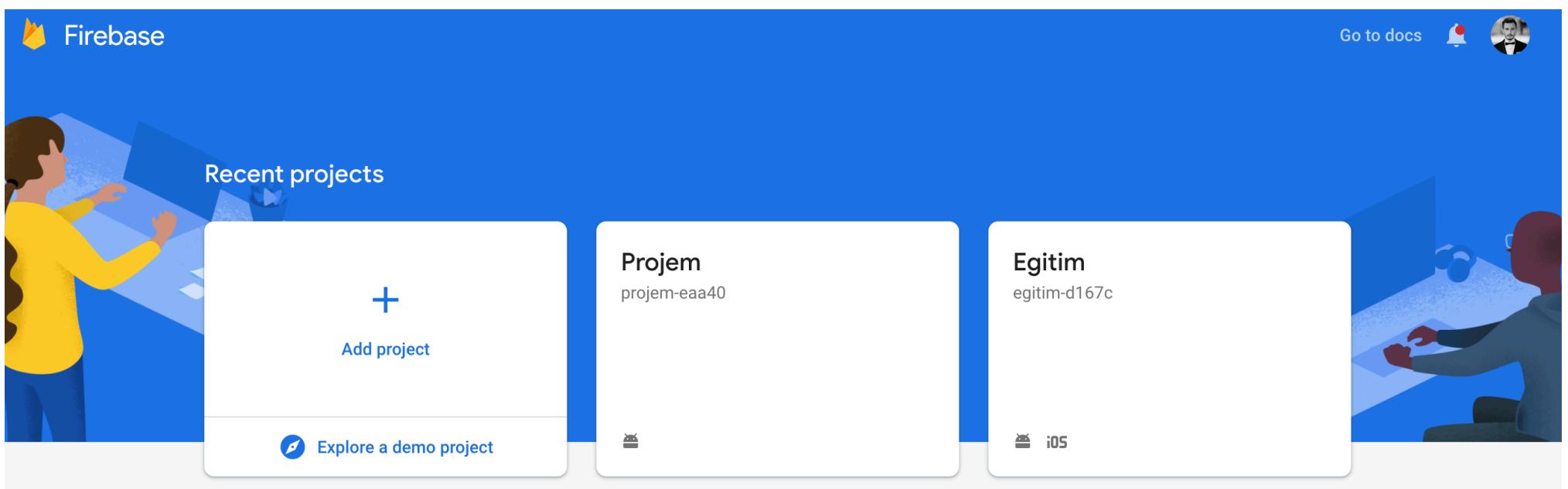
Klasik Firebase Kurulumu

Android Studio Projemiz Oluşturulur



Firebase Projesi Oluşturma

Firebase console üzerinde proje oluşturmalıyız.



Firebase İçinde Android Studio Projesi Oluşturma

The image shows three main components:

- Firebase Project Overview:** On the left, showing navigation tabs for Project Overview, Build, and Authentication. The Build tab is selected.
- Add Firebase to your Android app:** A modal dialog on the right. It has a step 1: Register app section with fields for "Android package name" (containing "com.example.myapplication") and "App nickname (optional)" (containing "My Android App"). It also includes a "Debug signing certificate SHA-1 (optional)" field with placeholder text and a note about its use for Dynamic Links and Google Sign-In.
- Gradle Scripts:** A code editor on the bottom left showing the contents of build.gradle. The "applicationId" line is highlighted with a pink rectangle, and a red arrow points from this highlight to the corresponding "Android package name" field in the Firebase dialog.

```
defaultConfig {  
    applicationId "com.example.myapplication"  
    minSdkVersion 16  
    targetSdkVersion 30  
    versionCode 1  
    versionName "1.0"  
  
    testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
}
```

Config Dosyasını Projeye Ekleme

Not : Proje düzeni Project formatında olmalıdır.

X Add Firebase to your Android app

1 Register app
Android package name: com.example.myapplication

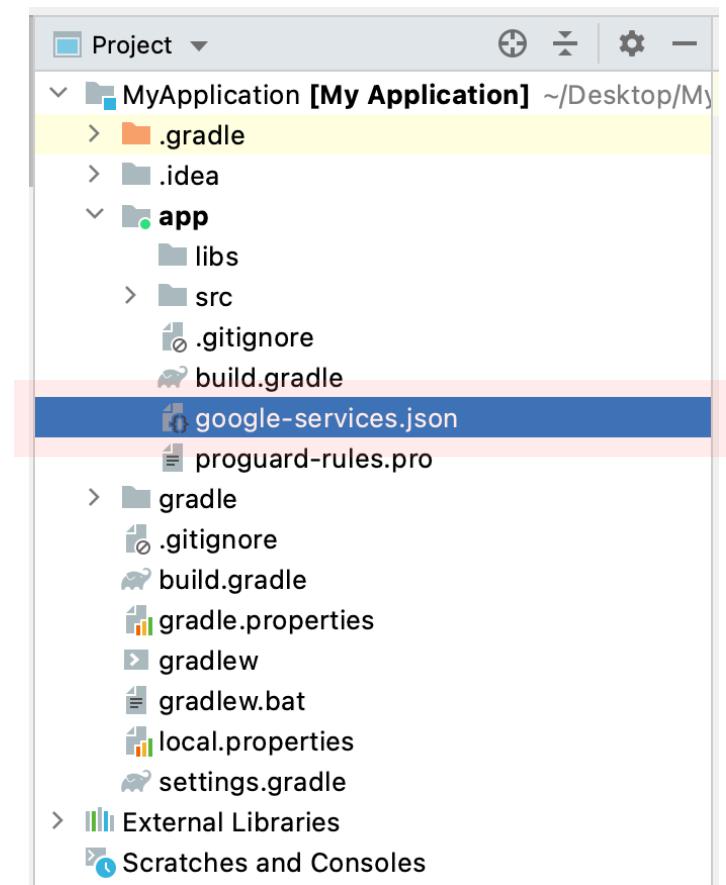
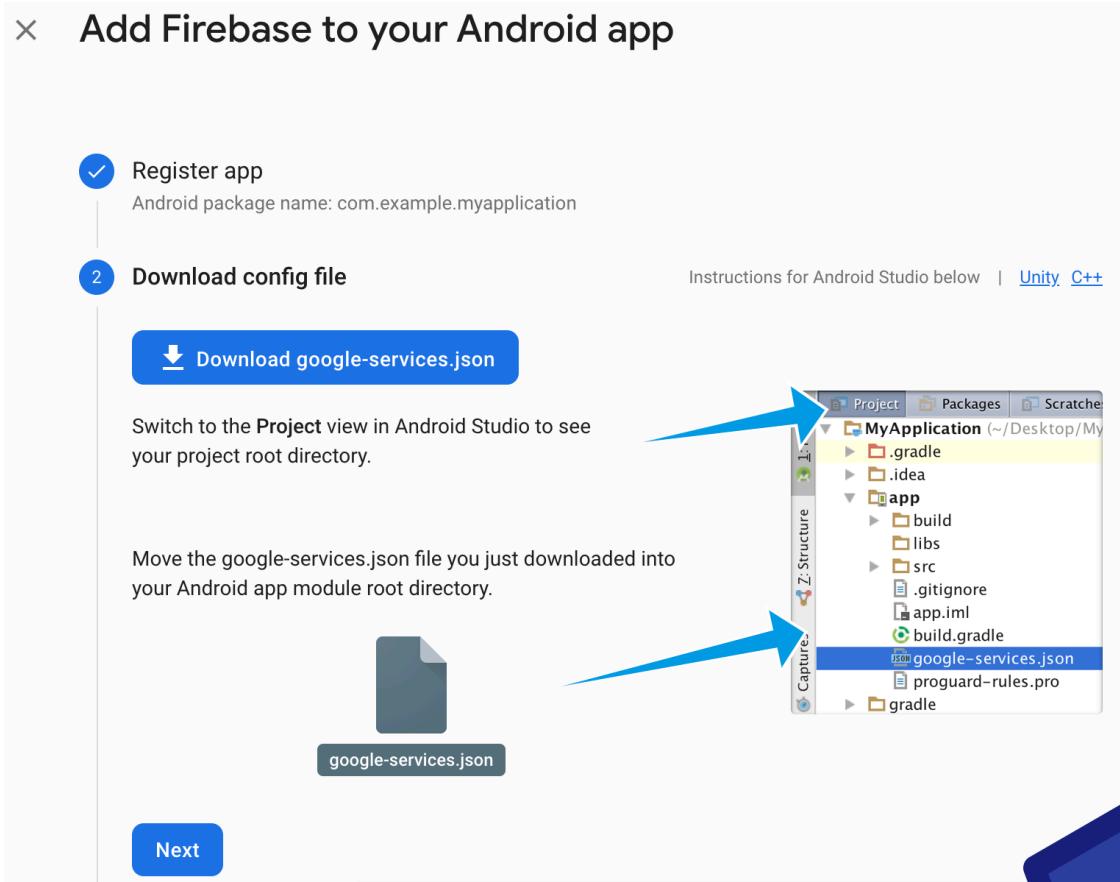
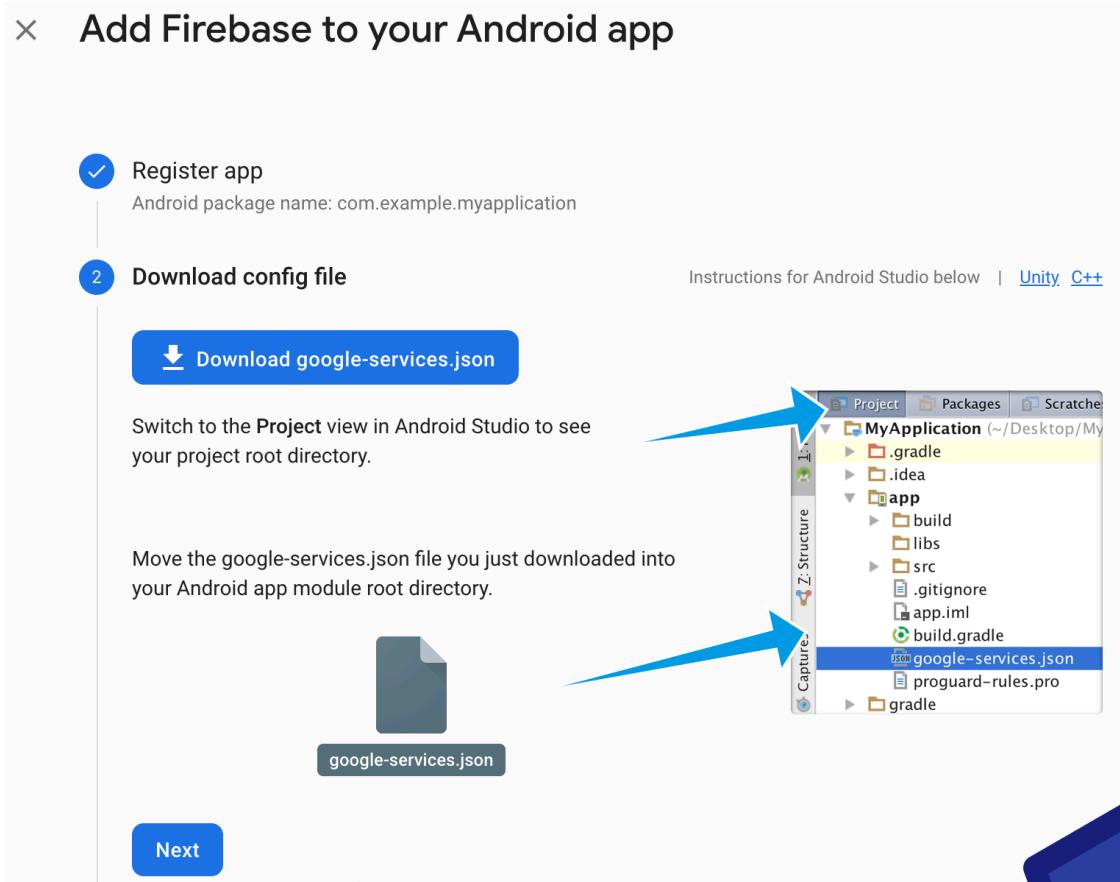
2 Download config file

[Download google-services.json](#)

Switch to the Project view in Android Studio to see your project root directory.

Move the google-services.json file you just downloaded into your Android app module root directory.

Next



Standart Kütüphane Eklenileri

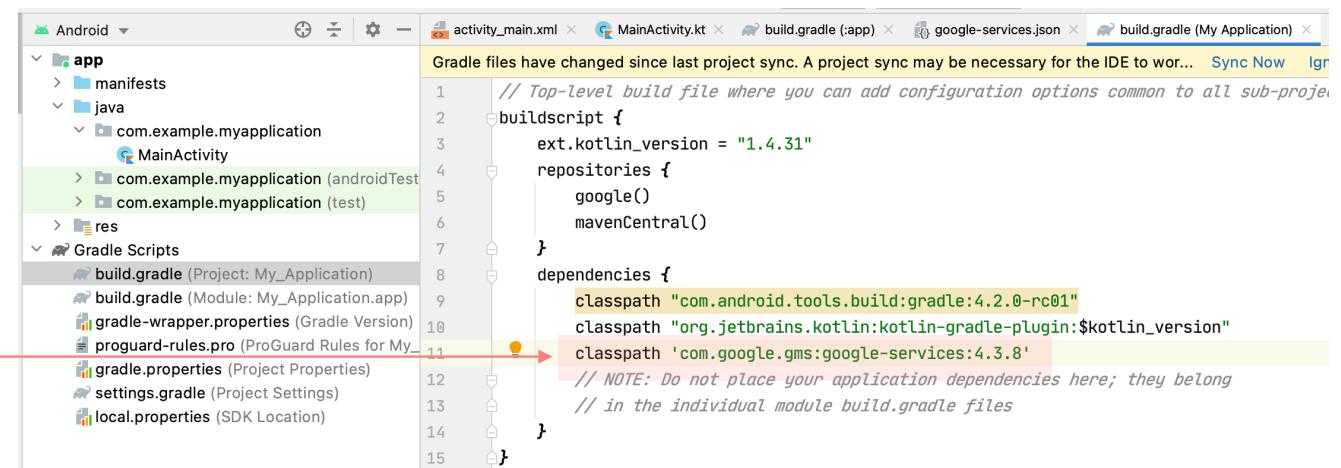
3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

The Google services plugin for [Gradle](#) loads the google-services.json file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (<project>/build.gradle):

```
buildscript {  
    repositories {  
        // Check that you have the following line (if not, add it):  
        google() // Google's Maven repository  
    }  
    dependencies {  
        ...  
        // Add this line  
        classpath 'com.google.gms:google-services:4.3.8'  
    }  
}  
  
allprojects {  
    ...  
    repositories {  
        // Check that you have the following line (if not, add it):  
        google() // Google's Maven repository  
        ...  
    }  
}
```



Android Studio interface showing the project structure and build.gradle file. The build.gradle file contains the configuration for the Google services plugin.

```
// Top-level build file where you can add configuration options common to all sub-projects.  
buildscript {  
    ext.kotlin_version = "1.4.31"  
    repositories {  
        google()  
        mavenCentral()  
    }  
    dependencies {  
        classpath "com.android.tools.build:gradle:4.2.0-rc01"  
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"  
        classpath 'com.google.gms:google-services:4.3.8'  
        // NOTE: Do not place your application dependencies here; they belong  
        // in the individual module build.gradle files  
    }  
}
```

Standart Kütüphane Eklentileri

The screenshot shows the Android Studio interface with the code editor open to the app-level build.gradle file. The file contains the following code:

```
App-level build.gradle (<project>/<app-module>/build.gradle):
```

```
apply plugin: 'com.android.application'  
// Add this line  
apply plugin: 'com.google.gms.google-services'  
  
dependencies {  
    // Import the Firebase BoM  
    implementation platform('com.google.firebase:firebase-bom:28.0.1')  
  
    // Add the dependencies for the desired Firebase products  
    // https://firebase.google.com/docs/android/setup#available-libraries  
}
```

By using the Firebase Android BoM, your app will always use compatible Firebase library versions. [Learn more](#)

Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed since last sync. [Sync now](#)

Previous [Next](#)

Gradle Scripts

- build.gradle (Project: My_Application)
- build.gradle (Module: My_Application.app) *
- gradle-wrapper.properties (Gradle Version)
- proguard-rules.pro (ProGuard Rules for My_Application)
- gradle.properties (Project Properties)
- settings.gradle (Project Settings)
- local.properties (SDK Location)

```
35 apply plugin: 'com.android.application'  
36 apply plugin: 'com.google.gms.google-services'  
37  
38 dependencies {  
39     implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"  
40     implementation 'androidx.core:core-ktx:1.3.1'  
41     implementation 'androidx.appcompat:appcompat:1.2.0'  
42     implementation 'com.google.android.material:material:1.2.1'  
43     implementation 'androidx.constraintlayout:constraintlayout:2.0.1'  
44     testImplementation 'junit:junit:4.+'  
45     androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
46     androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
47  
48 }  
49  
50 implementation platform('com.google.firebase:firebase-bom:28.0.1')
```

Firebase Konsoldan Projelin Kontrolü

Proje kurulumu yapıldıktan sonra android studio projesi firebase konsolda görünmeliidir.

4 Next steps

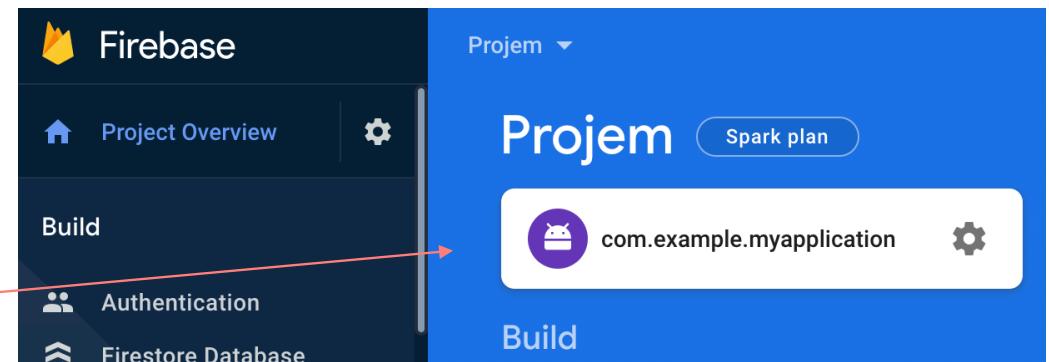
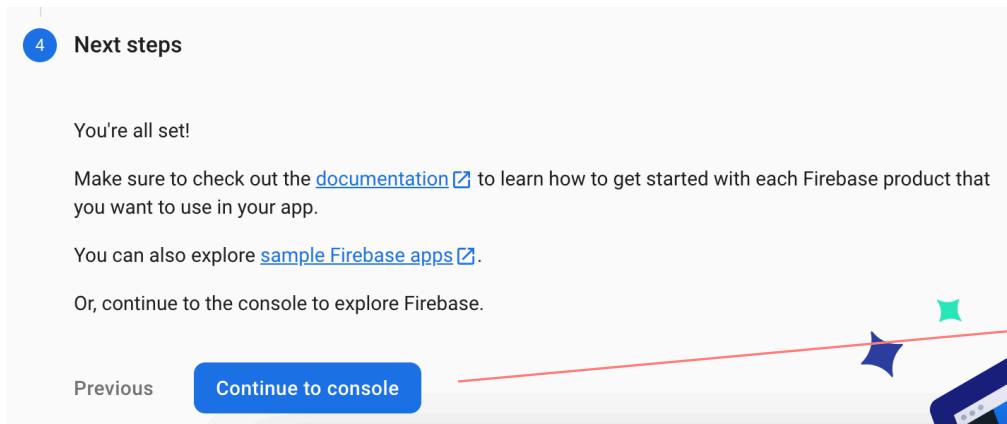
You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

Previous [Continue to console](#)



Firebase Üzerinde Çalışmak İstediğimiz Kütüphaneyi Ekleme

Firebase birçok teknoloji içermektedir , projemizde hangisini kullanmak istiyorsak ilgili kütüphaneyi eklemeliyiz.

Bu kütüphane linkleri aşağıdaki linkten ulaşılabilir.

<https://firebase.google.com/docs/android/setup#available-libraries>

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

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.1'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.1'
    implementation 'com.google.firebase:firebase-firebase:23.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    implementation platform('com.google.firebaseio.firebaseio-bom:28.0.1')
    implementation 'com.google.firebaseio.firebaseio-database-ktx:20.0.0'
}
```

In-App Messaging	com.google.firebaseio.firebaseio-inappmessaging-ktx	20.0.0	✓ (required)
In-App Messaging Display	com.google.firebaseio.firebaseio-inappmessaging-display-ktx	20.0.0	✓ (required)
Firebase installations	com.google.firebaseio.firebaseio-installations-ktx	17.0.0	
Firebase ML Model Downloader API	com.google.firebaseio.firebaseio-ml-modeldownloader-ktx	24.0.0	
Performance Monitoring	com.google.firebaseio.firebaseio-perf-ktx	20.0.0	
Performance Monitoring plugin	com.google.firebaseio.firebaseio-perf-plugin	1.4.0	
Realtime Database	com.google.firebaseio.firebaseio-database-ktx	20.0.0	
Remote Config	com.google.firebaseio.firebaseio-config-ktx	21.0.0	✓
Google Play services plugin	com.google.gms.google-services	4.3.8	

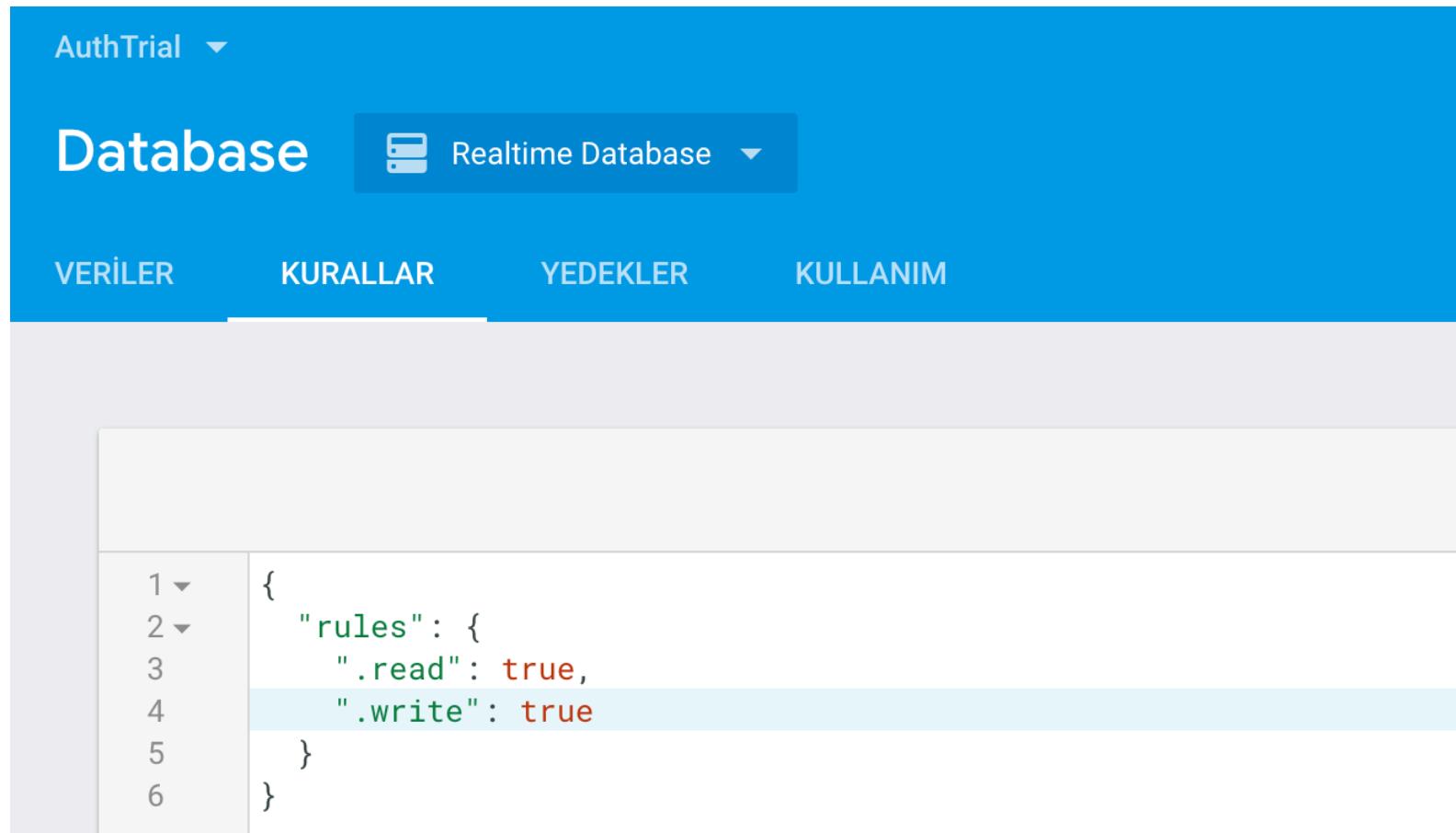
Veri tabanı Kuralları

AuthTrial ▾

Database  Realtime Database ▾

VERİLER KURALLAR YEDEKLER KULLANIM

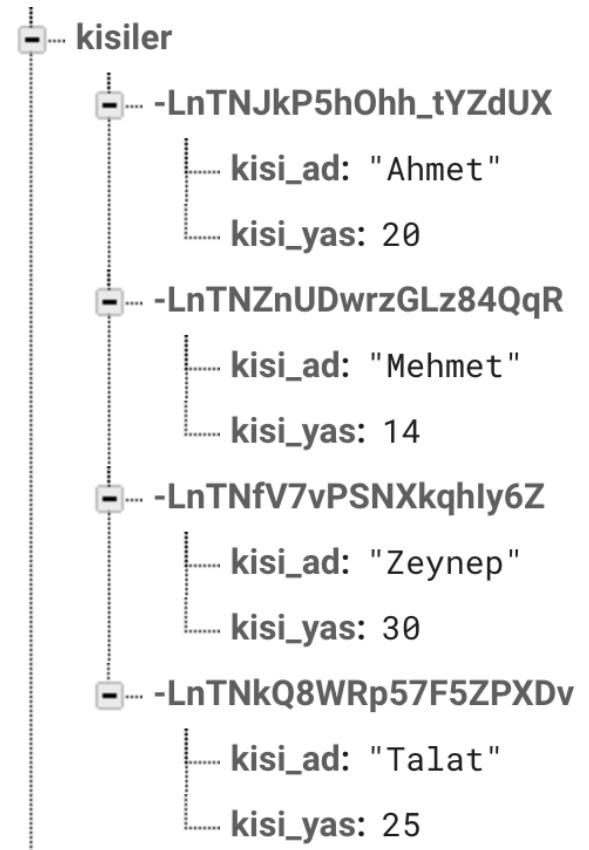
```
1 ▾ {  
2 ▾   "rules": {  
3     ".read": true,  
4     ".write": true  
5   }  
6 }
```



Veri yapısı

```
@IgnoreExtraProperties  
data class Kisiler(var kisi_ad:String? = "",  
                    var kisi_yas:Int? = 0) {  
}
```

Not : Firebase sınıf özelliklerinin varsayılan değeri olan nullable değişken olmasını istemektedir.

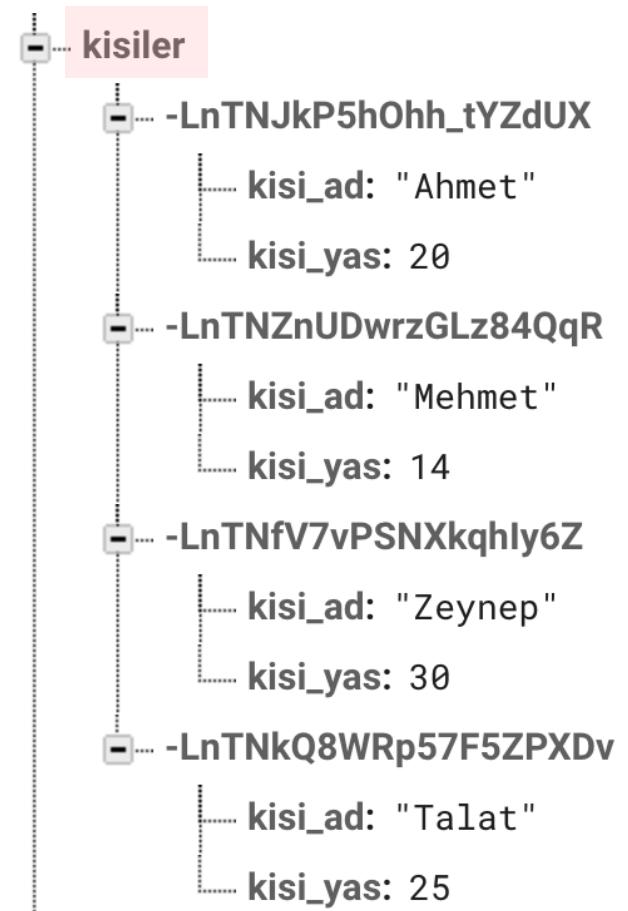


Veri yazılacak tablonun seçilmesi

```
@Composable
```

```
fun Sayfa() {  
    ekle()  
}
```

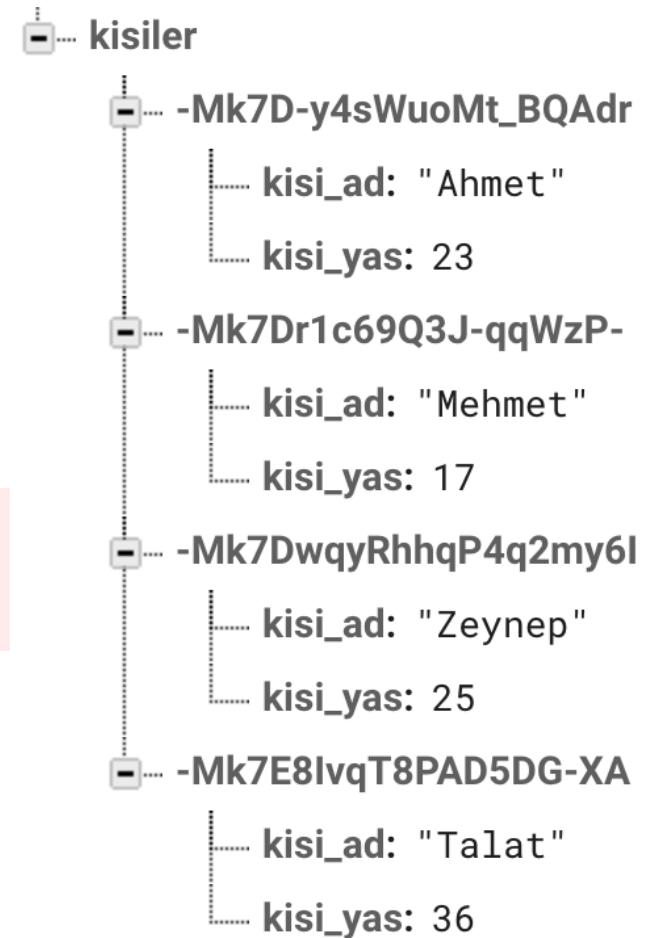
```
fun ekle(){  
    val db = FirebaseDatabase.getInstance()  
    val refKisiler = db.getReference( path: "kisiler")  
  
    val yeniKisi = Kisiler( kisi_ad: "Ahmet", kisi_yas: 23)  
    refKisiler.push().setValue(yeniKisi)  
}
```



Insert - Veri Kaydı

```
fun ekle(){
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

    val yeniKisi = Kisiler( kisi_ad: "Ahmet", kisi_yas: 23)
    refKisiler.push().setValue(yeniKisi)
}
```



Veri Okuma

```
fun tumKisiler(){
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

    refKisiler.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {

            for(d in snapshot.children){
                val kisi = d.getValue(Kisiler::class.java)

                if(kisi != null){
                    Log.e( tag: "*****", msg: "*****")
                    Log.e( tag: "Kişi key",d.key!!)
                    Log.e( tag: "Kişi ad",kisi.kisi_ad!!)
                    Log.e( tag: "Kişi yaşı",kisi.kisi_yas.toString())
                }
            }
        }

        override fun onCancelled(error: DatabaseError) {}
    })
}
```

```
@IgnoreExtraProperties
data class Kisiler(var kisi_ad:String? = "",  
                  var kisi_yas:Int? = 0) {  
}
```



Delete : Veri silme

```
fun sil(){
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

    refKisiler.child( pathString: "-Mk7DwqyRhhqP4q2my6I").removeValue()
}
```



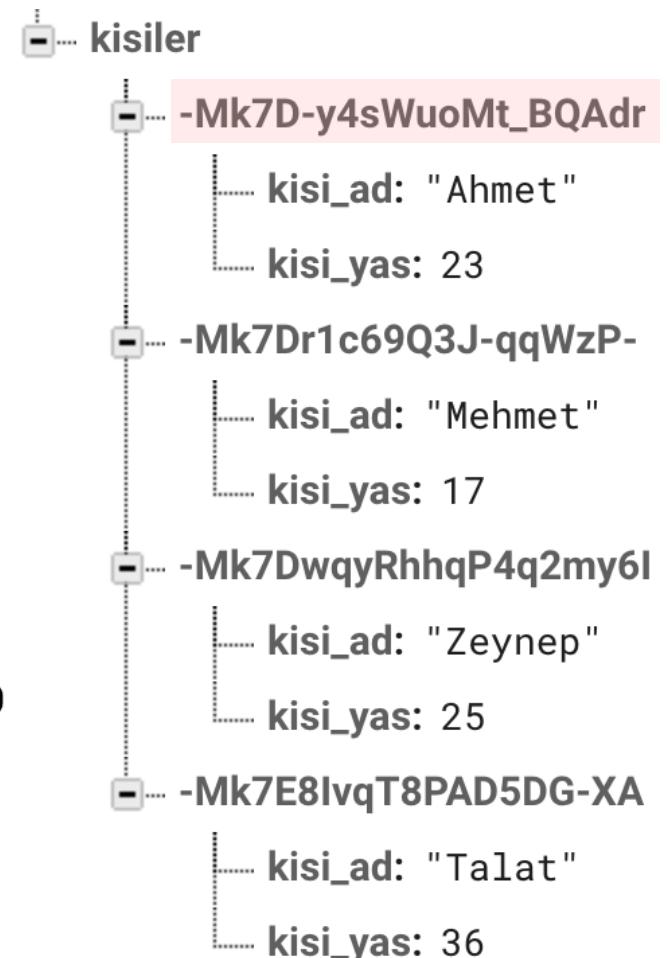
Update : Veri güncelleme

```
fun guncelle(){
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

    val bilgiler = HashMap<String,Any>()

    bilgiler["kisi_ad"] = "Yeni Ahmet"
    bilgiler["kisi_yas"] = 99

    refKisiler.child( pathString: "-Mk7D-y4sWuoMt_BQAdr").updateChildren(bilgiler)
}
```



Sorgu çeşitleri

EqualsTo : Eşitlik Arama

```
fun esitlikArama() {
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

    val sorgu = refKisiler.orderByChild( path: "kisi_ad").equalTo( value: "Mehmet")

    sorgu.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {

            for(d in snapshot.children){
                val kisi = d.getValue(Kisiler::class.java)

                if(kisi != null){
                    Log.e( tag: "*****", msg: "*****")
                    Log.e( tag: "Kişi key",d.key!!)
                    Log.e( tag: "Kişi ad",kisi.kisi_ad!!)
                    Log.e( tag: "Kişi yaşı",kisi.kisi_yas.toString())
                }
            }
        }

        override fun onCancelled(error: DatabaseError) {}
    })
}
```



LimitToFirst veya LimitToLast

```
fun limit() {
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

    val sorgu = refKisiler.limitToFirst( limit: 1)

    sorgu.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {

            for(d in snapshot.children){
                val kisi = d.getValue(Kisiler::class.java)

                if(kisi != null){
                    Log.e( tag: "*****", msg: "*****")
                    Log.e( tag: "Kişi key",d.key!!)
                    Log.e( tag: "Kişi ad",kisi.kisi_ad!!)
                    Log.e( tag: "Kişi yaşı",kisi.kisi_yas.toString())
                }
            }
        }

        override fun onCancelled(error: DatabaseError) {}
    })
}
```



Değer Aralığında Veri Alma

```
fun degerAraligi() {
    val db = FirebaseDatabase.getInstance()
    val refKisiler = db.getReference( path: "kisiler")

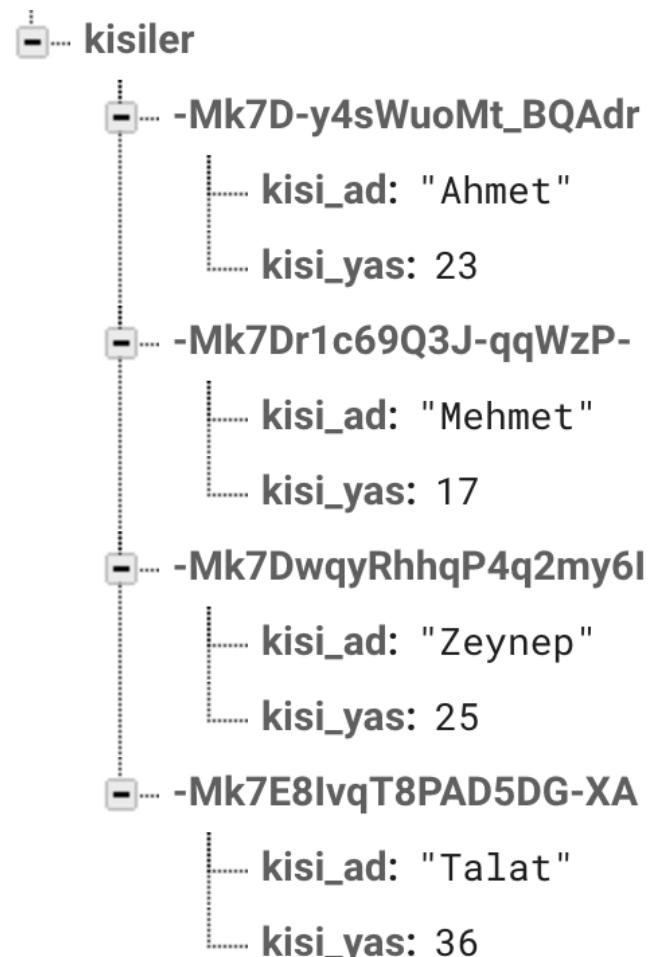
    val sorgu = refKisiler.orderByChild( path: "kisi_yas").startAt( value: 30.0).endAt( value: 40.0)

    sorgu.addValueEventListener(object : ValueEventListener {
        override fun onDataChange(snapshot: DataSnapshot) {

            for(d in snapshot.children){
                val kisi = d.getValue(Kisiler::class.java)

                if(kisi != null){
                    Log.e( tag: "*****", msg: "*****")
                    Log.e( tag: "Kişi key",d.key!!)
                    Log.e( tag: "Kişi ad",kisi.kisi_ad!!)
                    Log.e( tag: "Kişi yaşı",kisi.kisi_yas.toString())
                }
            }
        }

        override fun onCancelled(error: DatabaseError) {}
    })
}
```



Glide

Glide Kütüphanesi

- Glide ile url üzerinden resimleri uygulama içinde görüntüleyebiliriz.
- Jetpack compose desteği vadır.
- Hafıza yönetimi başarılıdır.

```
GlideImage(  
    imageModel = "http://kasimadalan.pe.hu/yemekler/resimler/${yemek.yemek_resim_adi}",  
    modifier = Modifier.size(100.dp)  
)
```

Kurulum

```
<uses-permission android:name="android.permission.INTERNET" />
```

```
dependencies {

    implementation 'androidx.core:core-ktx:1.8.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.5.1'
    implementation platform('androidx.compose:compose-bom:2022.10.00')
    implementation 'androidx.compose.ui:ui'
    implementation 'androidx.compose.ui:ui-graphics'
    implementation 'androidx.compose.ui:ui-tooling-preview'
    implementation 'androidx.compose.material3:material3'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation platform('androidx.compose:compose-bom:2022.10.00')
    androidTestImplementation 'androidx.compose.ui:ui-test-junit4'
    debugImplementation 'androidx.compose.ui:ui-tooling'
    debugImplementation 'androidx.compose.ui:ui-test-manifest'

    implementation "com.github.skydoves:landscapist-glide:1.4.4"
}

implementation "com.github.skydoves:landscapist-glide:1.4.4"
```

Güncelleme

- **http** ile başlayan web servisler varsayılan olarak çalışmamaktadır.
- **https** ile başlayan web servislerde sorun yaşamamaktadır.
- Bu sorunu çözmek için aşağıdaki ifadeyi manifest dosyasına eklemeniz gereklidir.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.volleykullanimi">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="VolleyKullanimi"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Resim Görüntüleme

```
@Composable
fun Sayfa() {
    val resimAdi = remember { mutableStateOf( value: "django.png" ) }

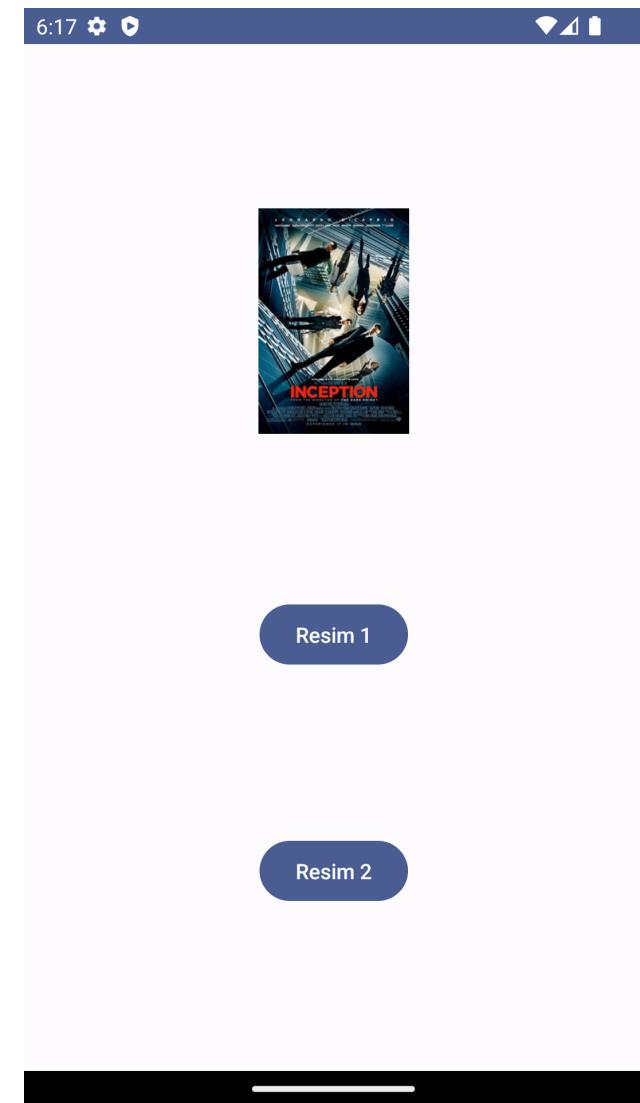
    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally
    ) { this: ColumnScope
        GlideImage(imageModel = "http://kasimadalan.pe.hu/filmller/resimler/${resimAdi.value}",
            modifier = Modifier.size(100.dp,150.dp))
        Button(onClick = { resimAdi.value = "inception.png" }) { this: RowScope
            Text(text = "Resim 1")
        }
        Button(onClick = { resimAdi.value = "interstellar.png" }) { this: RowScope
            Text(text = "Resim 2")
        }
    }
}
```

Boyutlandırma olmaz ise
bulunduğu alan kadar
büyük görünür



Resim 1

Resim 2



Teşekkürler...



kasım-adalan



kasimadalan@gmail.com



kasimadalan