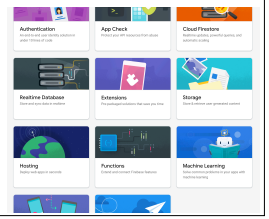
Görsel Programlama

Firebase Bağlantısı

Emir ÖZTÜRK

Firebase
Servisler

- Backend as a service
- Veritabanı
- Bulut saklama
- Doğrulama
- Makine öğrenmesi



Firebase google tarafından sunulan bir platformdur. Platform içerisinde bir çok servis bulunmaktadır.

Firebase
Ön kurulumlar

- FlutterFire
- Firebase CLI
- Google hesabı ile firebase login
- Dart pub global activate flutterfire_cli

Flutter ile Firebase Realtime Database'i kullanmak için belirli kurulumlar gerçekleştirilmelidir. FlutterFire flutter'ın Firebase'e bağlantısını Firebase CLI kullanarak sağlar. Bu iki kurulum gerçekleştirildiğinde google servislerine ulaşabilmek için login olmak gerekmektedir.

Firebase
Konfigürasyon

- flutter pub add firebase_core
- flutterfire configure
- main.dart içerisinde Firebase.initializeApp()
- options: DefaultFirebaseOptions.currentPlatform

Firebase_core paketi eklendikten sonra flutterfire configure ile Firebase tarafında bir proje açılır ya da mevcut bir projenin konfigürasyonu yapılır. Çalıştırılacak platformlar belirlenir ve buna göre anahtarlar oluşturulur.
Firebase servislerini aktifleştirmek için uygulamada bir defaya mahsus (çalıştırma esnasında) Firebase.initializeApp() çağırılmalıdır.

Firebase
Plugin'ler

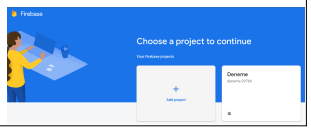
- Uygulamanın ihtiyaç duyduğu hizmetlerin plugin'lerinin eklenmesi
- flutter pub add plugin_adi
- flutterfire configure

Product	Plugin name	iOS	Android	Web	Other Apple (macOS, etc.)
Analytics	firebase_analytics	✓	✓	✓	beta
App Check	firebase_app_check	✓	✓	✓	beta
Authentication	firebase_auth	✓	✓	✓	beta
Cloud Firestore	cloud_firestore	✓	✓	✓	beta
Cloud Functions	cloud_functions	✓	✓	✓	beta
Cloud Messaging	firebase_messaging	✓	✓	✓	beta
Cloud Storage	firebase_storage	✓	✓	✓	beta
Crashlytics	firebase_crashlytics	✓	✓		beta
Dynamic Links	firebase_dynamic_links	✓	✓		
In App Messaging	firebase_in_app_messaging	✓	✓		
Firebase Installations	firebase_app_installations	✓	✓	✓	beta
ML Model Downloader	firebase_ml_model_downloader	✓	✓		beta
Performance Monitoring	firebase_performance	✓	✓	✓	
Realtime Database	firebase_database	✓	✓	✓	beta
Remote Config	firebase_remote_config	✓	✓	✓	beta

Uygulamada bir çok servis plugin'i kullanılabilir. Veritabanı için firebase_database kullanılmalıdır.

Veritabanının oluşturulması
Firebase Console

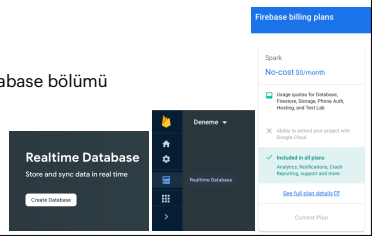
- Yeni bir proje oluşturma
- Proje ismi belirleme



Eğer proje flutterfire configure ile kurulmuyorsa web üzerinden de tanımlanabilir.

Veritabanının oluşturulması
Firebase Console

- Proje seçildikten sonra paket seçimi
- Ücretsiz paket
- Erişim limitleri
- Proje menüsünden realtime database bölümü
- Create database



Projenin kullanacağı paketler yandaki menüden seçilebilmektedir. Bazı servisler ücretli olmakla beraber bazı servisler ise ücretsiz belirli planlar sunmaktadır.

Veritabanının Oluşturulması
Güvenlik kuralları

- Locked mode
- Test mode

Veritabanı oluşturulduğunda locked ve test olmak üzere iki moda sahiptir. Locked modda okuma yazma engeli bulunurken test modunda konfigürasyon ekleme ve silme işlemlerine izin verilecek şekilde değiştirilir.

Veritabanına Erişim
Konfigürasyonlar

- Flutter pub add firebase_database
- Veritabanı instance'ı almak için database = FirebaseDatabase.instance

Veritabanının bir örneğini almak için FirebaseDatabase.instance kullanılabilir. Döndürülen referans uygulama içerisinde veritabanına erişmek için kullanılmaktadır.

Veritabanına Erişim
Yapı

- İlişkisel veritabanları
  - İlişkiler
- Belge veritabanları ve firebase
  - Json
- JSON
  - Anahtar değer ikilileri
  - Hiyerarşik
  - {}

Firebase Realtime Database bir belge veritabanıdır. İlişkisel veritabanlarının aksine tablolar ve tablolar arasındaki ilişkiler bulunmamaktadır. Bunun yerine Json formatında dokümanlar saklanır. Belirli alanların birden fazla dokümanda geçmesi gerekiyorsa veri tekrarı yapılır. Bu durumda bu tekrarda veri bütünlüğünün uygulama tarafında kontrol edilmesi gerekmektedir. Bunun yerine belirli id'lerin kullanılması ile dokümanlar birbirleri ile ilişkilendirilebilir.

Veritabanına Erişim
Sınıf - Json örneği

```
{
  "name": "John Smith",
  "email": "john@example.com"
}
```



```
class User {
  final String name;
  final String email;

  User(this.name, this.email);

  User.fromJson(Map<String, dynamic> json)
      : name = json['name'],
        email = json['email'];

  Map<String, dynamic> toJson() => {
        'name': name,
        'email': email,
      };
}
```

Bir json dosyasının Dart sınıfı olarak tanımlanması

Veritabanına Erişim
Sınıf - Json arası dönüşüm

- Serialization - Deserialization
- jsonDecode(jsonString)
  - Map<String, dynamic>
- jsonEncode(nesne)
  - String
- Ek metotlar eklenebilir
  - fromJson()
  - toJson()

```
class User {
  final String name;
  final String email;

  User(this.name, this.email);

  User.fromJson(Map<String, dynamic> json)
      : name = json['name'],
        email = json['email'];

  Map<String, dynamic> toJson() => {
        'name': name,
        'email': email,
      };
}
```

Veritabanına bir sınıfın gönderilmesi ya da veritabanından bir sınıf alınması işlemlerinde sınıftan json'a veya json'dan sınıfa dönüşüm işlemlerinin gerçekleştirilmesi gerekmektedir. Bunun için fromJson, toJson metotları sınıf içerisine yazılarak serialization - deserialization işlemi gerçekleştirilebilir.

Veritabanına Erişim
Okuma - Yazma

- Veritabanına okuma yazma yapmak için referans
- DatabaseReference
- Ref = FirebaseDatabase.instance.ref()
- Bir koleksiyonun içerisinde id isteniyorsa ref("koleksiyon/id")
- R = ref.push()
- Yazmak için ref.set(yeni_nesnenin_json_stringi)
  - Üzerine yazma

Veritabanının instance'ı üzerinde ref metodu ile bir referans alınır. Alınan bu ref içerisinde string olarak koleksiyon ve koleksiyon içerisindeki belgenin id'si verilebilir. Id verilmediği zaman tüm koleksiyona erişilir. Eğer yeni bir kayıt eklenmek isteniyorsa ref.push() ile yeni bir kayıt açılarak dönen değer set edilebilir. Set işlemi update de gerçekleştirmektedir.

Veritabanına Erişim
Alan Güncelleme

- Yalnızca bir alt düğümün güncellenmesi isteniyorsa
- ref.update()
- Yalnızca güncellenecek parametrenin anahtarı ve değeri
- Birden fazla parametre olabilir

Eğer belirli bir alanın update edilmesi isteniyorsa ref.update kullanılabilir. Bu metoda parametre olarak bir alan ve güncellenecek değer verilir.

Veritabanına Erişim
Okuma

- Okunacak değerlerin referansının alınması
- Veritabanı instance'ının ref'i
  - get()
- children
- Map<String,dynamic>'e dönüşüm

```
var k = await widget.database!.ref("tweets").get();
var records = k.children.map((e) {
  var x = e.value as Map<String, dynamic>;
  return Record.fromJson(x);
}).toList();
```

Veritabanından alınan referansın children değeri tüm düğümleri döndürür. Bu düğümler Map<String,dynamic>'e dönüştürüldükten sonra istenilen sınıf türüne çevrilebilir.

Veritabanına Erişim

Silme

- Veritabanına eklenmiş bir kaydın getirilmesi
- FirebaseDatabase.instance.ref().remove()

Herhangi bir kaydı silmek için referansa gerekli koleksiyon ve bu koleksiyonun id'si verilir. Daha sonra elde edilen bu değer için remove() metodu çağırılır.