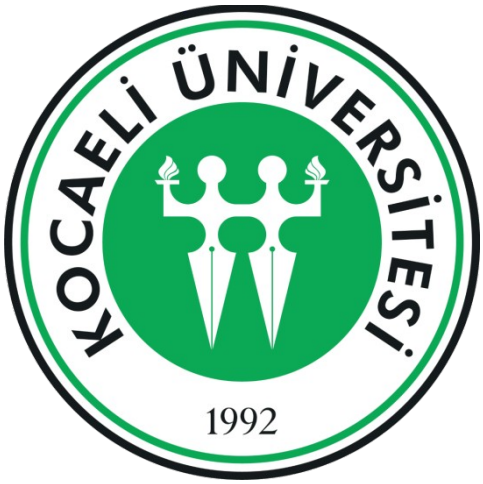


KOCAELİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

YAZILIM LABORATUVARI - II PROJE - II
MOBİL SORGU PROJESİ

CEMRE CAN KAYA
190201137

ENGİN YENİCE
190201133

KOCAELİ 2020

Mobil Sorgu Uygulaması

Cemre Can Kaya
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
190201137

Engin Yenice
Bilgisayar Mühendisliği
Kocaeli Üniversitesi
190201133

***Özet*— Bulut bilişim ve Google Map api kullanarak android platformunda bir uygulama geliştirmeniz beklenmektedir. Taksi gezinge (trajectory) verileri kullanılarak android platformunda farklı sorguların yapılabildiği bir uygulama geliştirmeniz amaçlanmaktadır. ır.**

Anahtar Kelimeler—firebase, google, mobile, taksi, tlc, veri, sorgulama, çizme, api, bulut

GİRİŞ

Programın arka planı (backend) C#, mobil tarafı **React Native** ile geliştirilmiştir. TLC (Trip Record Data) 01-12-2020 – 31-12-2020 aralığında bulunan verileri kullanılarak istenilen sorguların cevaplarına ulaşılması hedeflenmiştir. Bu proje sayesinde bulut platformları ve kullanım yöntemleri hakkında bilgi edinilmiştir.

PROJE MİMARİSİ

Projenin arka planı (backend) kurumsal mimariye uygun bir şekilde geliştirilmiştir. Proje yapısı gereği 5 temel katmana parçalanmıştır.

Projenin ön planı (front-end) parçala yönet mantığı ile geliştirilmiş olup her bir işlem parçacığı ayrı bir bileşende (component) üzerinde yapılmaya çalışılmıştır.

1) Arka Plan (Backend) Yapısı

Proje yapısı gereği 5 temel katmana parçalanmıştır. Bu katmanların detayları bu başlık altında açıklanmıştır.

2) Core Katmanı

Bu katman proje içerisinde bulunması gereken temel bileşenleri bulundurmaktadır.

1. Entities:

Oluşturduğumuz nesnelerin daha somut ve yönetilebilmesi için temel arayüzleri (interface) bu klasör içerisinde tutulmaktadır.

- **IEntity:** Temel nesnelerimizin temel arayüz (interface) sınıfıdır.
- **IDto:** Uzun ismi **Veri iletim nesnesi (Data transfer object)** olarak geçmektedir. Temel nesnelerimizden Kullanıcı arayüzüne (UI (User Interface (Front-end))) göndermek istediğimiz nesne tanımlamalarının temel arayüz (interface) sınıfıdır.

2. Utilities:

Projenin genelinde kullanabileceğimiz araçlar bu klasör altında tutulmaktadır.

- **Results:** İş (Business) katmanında kullanacağımız metotların geriye dönüş değerlerinin daha yönetilebilir ve daha düzenli olması için oluşturduğumuz bir araç sınıfıdır. Bu sınıfı temel olarak özetlenecek olursak. İçerisinde temel olarak 2 adet değişken bulunmaktadır. Bu değişkenler mesaj ve başarı (message, success) durumu olarak isimlendirilmektedir. Veri gönderilmesi durumunda miras verdiği alt sınıfta ise data (veri) değişkeninin bulunduğu ayrı bir dönüş tipi bulunmaktadır. Sınıf içerisinde ki değişkenlerin daha kolay yönetilmesi için başarı (success) durumuna göre alt sınıflar oluşturulmuştur. Bu sınıfların çağrılması durumunda başarı durumu otomatik olarak belirlenmektedir.

3) Entities Katmanı

Entity ve Dto arayüzlerinden (interface) örnek alan veri sınıfları bulunmaktadır.

1. Concrete

Entity arayüzünden (interface) örnek alınan veri sınıfları bulunmaktadır.

2. Dto:

Dto arayüzünden (interface) örnek alınan veri sınıfları bulunmaktadır.

4) DataAccess Katmanı

Veri havuzunun yönetilmesinden görevli katmandır. Bu katmanımızı ileride geliştirmeye açık olması için arayüzler (interface) kullanarak geliştirdik. Bu sayede ileride bir gerektiğinde başka yapılara geçmeyi planladık. Şuanda veri havuzumuzu bellekte (In Memory) olarak tutuyoruz.

1. Abstract

Veri sınıflarımızın arayüzlerinin tutulduğu klasör.

2. Concrete

Veri sınıflarımızın tutulduğu klasör.

- Veri sınıflarımızın tutulduğu klasör.
- **FirebaseLocationDal:** Lokasyon bilgilerinin firebase realtime database üzerinden çekildiği temel sınıftır.
- **FirebaseOperationDal:** Taksilerin hareket geçmişlerinin firebase realtime database üzerinden çekildiği temel sınıftır.

  FirebaseOperationLocationDal: Lokasyon bilgilerinin belirli şartlara göre sorgulandığı bir sınıftır. FirebaseOperationDal sınıfından kalıtım almaktadır.
- **FirebaseOperationTypeOneDal:** Tip 1 maddesine ait sorguların yapıldığı veri katmanıdır.
  Yapılan sorgular:
  - En fazla yolcu ta¸sınan 5 günü ve toplam yolcu sayılarını listeleyiniz.
  - Belirli mesafenin altında en çok seyahat yapılan günü ve seyahat uzunluğunu bulunuz (mesafe seçilebilmeli).
  - En uzun mesafeli 5 yolculuktaki günn ve mesafeleri listeleyiniz.
- **FirebaseOperationTypeTwoDal:** Tip 2 maddesine ait sorgularının yapıldığı veri katmanıdır.
  Yapılan sorgular:
  - Iki tarih arasında belirli bir lokasyondan hareket eden araç sayısı kaçtır? (tarihler ve lokasyon seçilebilmeli)
  - Günlük seyahat başına düşen ortalama alınan ücretlere göre; en az ücret alınan iki tarih arasındaki günlük alınan ortalama ücretleri listeleyiniz.
  - Iki tarih arasında seyahat edilen en az mesafeli 5 yolculuk hangisidir? (tarihler seçilebilmeli)
- **FirebaseOperationTypeThreeDal:** Tip 3 maddesine ait sorgularının yapıldığı veri katmanıdır
  Yapılan sorgular:
  - Belirli bir günde en uzun seyahatin harita üstünde yolunu çiziniz (Gün seçilebilmeli). Başlangıç ve varış konumları lokasyonun merkezi kabul edip mesafeye göre bir yol bulunmalıdır.
  - Belirli bir günde aynı konumdan hareket eden araçların rasgele 5'inin yolunu çiziniz (Gün ve konum seçilebilmeli). Başlangıç ve varış konumları lokasyonun merkezi kabul edip mesafeye göre bir yol bulunmalıdır.
  - En az 3 yolcunun bulunduğu seyahatlerden en kısa mesafeli ve en uzun mesafeli yolu çiziniz. Başlangıç ve varış konumları lokasyonun merkezi kabul edip mesafeye göre bir yol bulunmalıdır.
- **GoogleCoordinateDal:** Lokasyonların koordinat bilgilerini google api üzerinden tespit etmektedir.

3. Helper

- **GetRequestHelper:** Firebase sorguları için bağlantı kurulumu yapan sınıftır.

5) Business Katmanı

Projenin iş kodlarının yazıldığı katmandır. Bu gerekli işlemlerin yönetildiği katmandır. Temel olarak 4 klasöre bölünmüştür.

1. Abstract

Business sınıflarımızın arayüzlerinin tutulduğu klasör.

2. Concrete

Business sınıflarımızın tutulduğu klasör.

- **QueryExampleLocationManager:** API tarafından gelen isteklere DataAccess tarafından aldığı bilgiler doğrultusunda dönüş yapar.
- **QueryExampleOneManager:** API tarafından gelen isteklere DataAccess tarafından aldığı bilgiler doğrultusunda dönüş yapar.
- **QueryExampleTwoManager**: API tarafından gelen isteklere DataAccess tarafından aldığı bilgiler doğrultusunda dönüş yapar.
- **QueryExampleThreeManager**: API tarafından gelen isteklere DataAccess tarafından aldığı bilgiler doğrultusunda dönüş yapar.

3. DependencyResolvers

Bağımlılıkların çözülmesi ve isim havuzuna aktarılması için kullanılmaktadır. (Paket olarak Autofac kullanılmıştır.)

6) WebAPI katmanı

Ön Plan (Front-end) kısmından gelen istekleri karşılayıp gerekli dönüşleri yapmakla görevli olan katmandır. Ön Plan (Front-end) tarafından 10 farklı isteğe karşılık verebilecek 4 adet Controller bulunmaktadır.

1. ExampleOneQueriesController

Tip 1 sorguları ile ilgili iletişim kurmakla görevlidir. (Sorgu 1,2 ve 3 dahil.)

2. ExampleTwoQueriesController

Tip 2 sorguları ile ilgili iletişim kurmakla görevlidir. (Sorgu 1,2 ve 3 dahil.)

3. ExampleThreeQueriesController

Tip 3 sorguları ile ilgili iletişim kurmakla görevlidir. (Sorgu 1,2 ve 3 dahil.)

4. LocationController

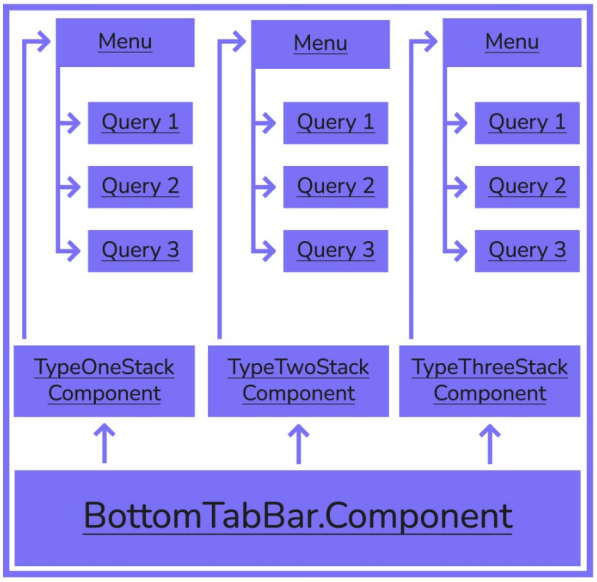
Lokasyon bilgileri ile ilgili iletişim kurmakla görevlidir.

7) Ön Plan (Frontend) Yapısı

Projenin kullanıcı ara yüzü (User Interface) ekranıdır. Arka plan (Backend) tarafında yazılan kodların kullanıcıya görüntüsel olarak aktarıldığı taraftır. React native ile geliştirilmiştir. Native-Base bileşen (component) paketi kullanılmıştır.

1. Ana Bileşen (Component) (app.component)

Projenin çalıştığı ana bileşendir. Bu bileşen içerisinde tab bar bulunmaktadır ve her tab içerisinde bir stack bulunmaktadır.



Bottom tab bar içerisinde 3 tab bulunur. Her tab içerisinde 1 stack bulunur. Bu stackler içerisinde 4 sayfa bulunur, 1 menu sayfası ve sorgu sayfaları ancak sorgu sayfalarına erişim menu sayfasından gerçekleşir. Her tab içerisinde de varsayılan olarak menu sayfası çalıştırılır.

YÖNTEM

1. *Verilerin saklanması*

Taksi gezinge verileri Firestore Cloud platformunda Realtime Database içerisinde JSON formatında tutulmaktadır. 1.3 M veri içerisinden filtreleme sonucu 155000 (günlük 5000) veri kullanılmıştır.

2. *Sorgular*

Sorgular Backend üzerinden çalıştırılmaktadır. Veriler Firestore Cloud üzerinden Backend üzerine çekilir ve sorgular sonucu geri dönen taksi verileri Web API üzerinden Frontend tarafına gönderilir. Front end tarafına gelen veriler ekranda gösterilir.

3. *Harita üzerinde gösterim*

Backend'ten gelen Tip 3 sorgularının verilerini Frontend tarafında harita üzerinde çizdirebilmek için MapView ve MapViewDirections Kütüphaneleri kullanılmaktadır.

- MapView

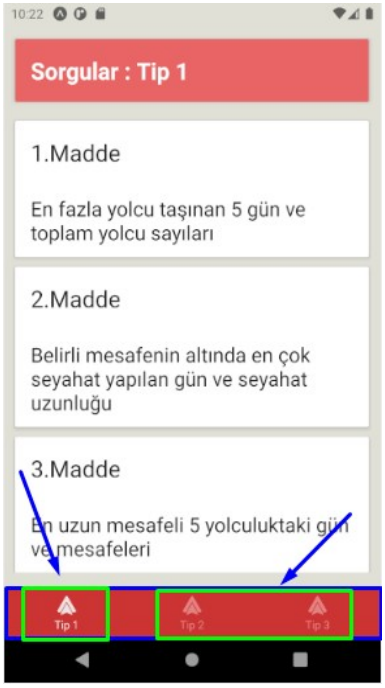
MapView bileşeni ile mobil uygulamanın bulunduğu işletim sistemine göre harita servis sağlayıcısı kullanılarak harita oluşturulur.

- MapViewDirections

MapViewDirections bileşeni için Google Cloud Api üzerinden alınan api key girilir. MapViewDirection bileşenine girilen iki konum bilgileri ile Api üzerinden iki konum arası yol bilgisi getirilir ve harita üzerinde çizdirilir.

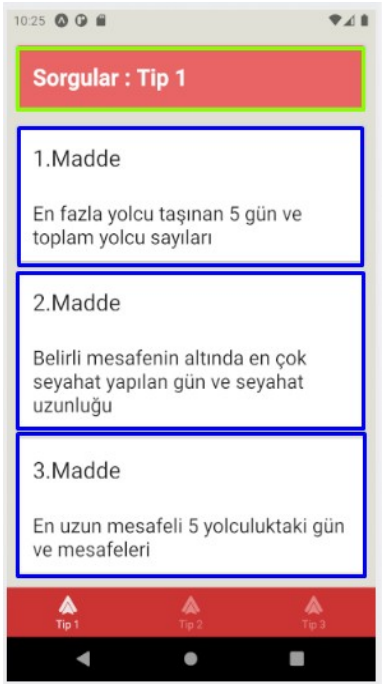
NASIL KULLANILIR

1) Yönlendirme Menüsü (Buttom Tab Bar)



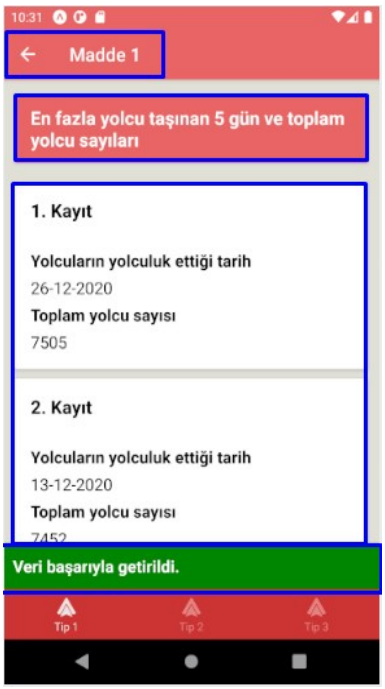
Sayfanın alt kısmında sabit olarak bulunur. Sorgu tipleri arasında geçiş yapmak için kullanılır. Bulunduğunuz menü daha parlak gözükürken aktif olmayan menüler ise daha soluk bir renkte gözükmektedir.

2) Ana Sayfa Kullanımı



Sayfanın en üst kısmında bulunan başlık alanı üzerinden hangi sorgu menüsü üzerinde gezdiğinizi görüntüleyebilirsiniz. Sorgulamak istediğiniz madde üzerine tıklayarak sorgulama işlemini gerçekleştirebilirsiniz.

3) Sorgu Sonucu Ekranı (Normal)



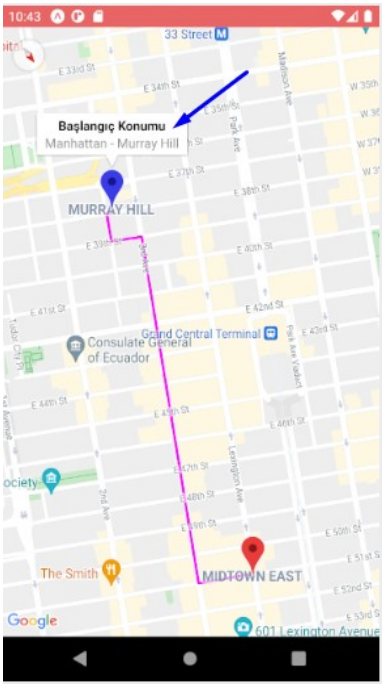
Sorgu sonucu ekranı üzerinde en üst kısımda seçilen madde gözükmektedir. Maddenin solunda bulunan ok işareti ile bir sayfa geriye gidebilirsiniz. (Ana sayfa veya veri giriş sayfası).

Sayfanın üst kısmında bulunan açıklama üzerinden sorgu hakkında detaylı sözel bilgiye ulaşabilirsiniz.

Sayfanın ortasında ki alanda sorgunun sonuçları ile ilgili bilgiler verilmektedir.

Sayfanın alt kısmında bulunan bilgilendirme mesajı alanı ile işlem hakkında kısa bir bilgi alabilirsiniz.

4) Sorgu Sonucu Ekranı (Harita)



Harita üzerinde verilen sonuçlarda

- Mavi olarak gösterilen işaretçi (marker) başlangıç konumunu göstermektedir.
- Varış konumları rastgele herhangi bir renkte işaretçi (marker) ile gösterilmektedir. (Resimde kırmızı)

Başlangıç ve varış noktası hakkında bilgiye ulaşmak için işaretçi (marker) üzerine tıklanması yeterlidir.

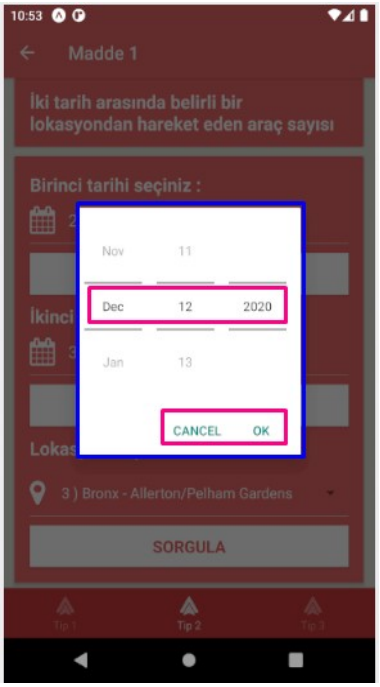
İşaretçi içerisinde bulunan bilgiler:

- Konum durumu (başlangıç veya bitiş)
- Konumun adres bilgisi

Sorgu ekranından geri çıkmak için telefonunuzda bulunan geri tuşuna basabilirsiniz. Bu işlem sizi bir önceki sayfaya yönlendirecektir.

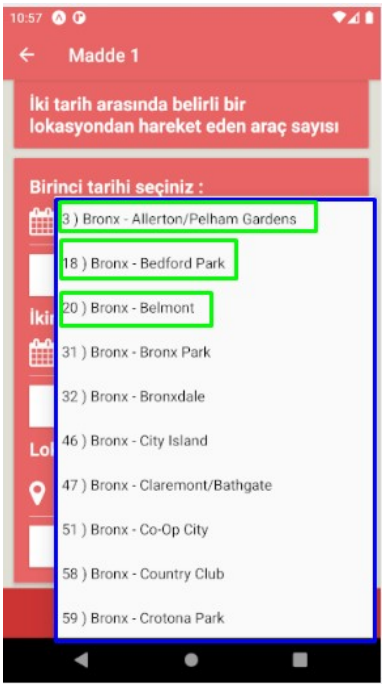
5) Giriş Nesnesi (İnput) Kullanımı

1. Tarih Seçimi



Tarih seçimi ekranı üzerinden Aralık 2020 tarihine ait herhangi bir seçebilirsiniz. Tarihler arası geçiş yapabilmek için değiştireceğiniz tarih türünün üzerine basıp elinizle yukarı veya aşağı hareket yaparak tarih üzerinde değişiklik yapabilirsiniz.

2. Lokasyon seçimi



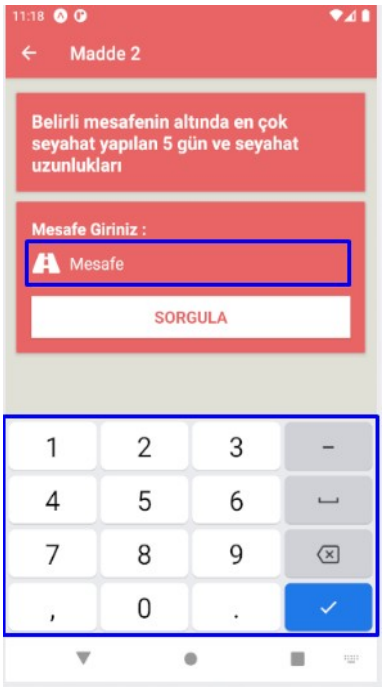
Açılan lokasyon listesi üzerinden istediğinizi herhangi bir lokasyonu seçebilirsiniz. Liste ilçe (borough) ve bölge (zone) ye göre alfabetik olarak sıralanmaktadır.

Lokasyon listesi içeriğini örnek bir veri üzerinden inceleyelim:

3) Bronx – Allerton/Pelham Gardens

1. 3 ile gösterilen kısım kayıt numarası (LocationID) temsil edilmektedir. 3)
2. İlk tire (-) işaretine kadar olan kısım ilçe (borough) olarak temsil edilmektedir. Bronx
3. İlk tire (-) işaretinden sonra ki kısım bölge (zone) olarak temsil edilmektedir. Allerton/Pelham Gardens

3. Mesafe seçimi



Mesafe girinizin yazısının altında bulunan giriş nesnesi (input) tıklanması durumunda ekranın altında ekran klavyesi açılacaktır. Sorgulamak istediğiniz mesafe değerini açılan klavye yardımıyla gerçekleştirebilirsiniz.

KAYNAKÇA

1) https://www.npmjs.com/package/react-native-maps-directions
2) https://www.npmjs.com/package/react-native-maps
3) https://enginyenice.com/
4) https://nativebase.io/
5) https://expo.io/
6) https://cloud.google.com/
7) https://firebase.google.com/
8) https://developers.google.com/maps