MOBİL SORGULAR PROJESİ

Hüseyin Yılmaz
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi
ylmzhsyn98@gmail.com

Özet

Bu projede bizden, bulut bilişim platformlarından birini ve Google Maps Api kullanarak, Android sistemlerde çalışacak, 2020 Aralık ayının New York taksi gezinge verileri üstünde çeşitli sorgular yapabileceğimiz bir mobil uygulama tasarlamamız istenmiştir.

Giriş

Çalışmada amaç, bulut bilişim platformlarına aşina olmak, Google Maps Api kullanımı konusunda tecrübe edinmek ve Android uygulamalar yazma konusunda deneyim kazanmaktır.

Projede dil ve framework serbest bırakılmıştır. Veritabanı olarak AWS RDS hizmeti üzerinden Postgresql veritabanı kullanılmıştır, backend'de Typescript ile Express.js, frontend'de ise yine Typescript ile React Native tercih edilmiştir. Veritabanı sorgularını kolaylaştırmak için ise bir ORM olan Prisma kullanılmıştır.

Typescript, açık kaynaklı bir programlama dilidir. Tüm Javascript kodları Typescript için de geçerlidir, Javascript'ten farkı ise statik tip tanımlama sayesinde runtime hataları riskini en aza indirmesi ve daha güvenli kodlar yazma imkanı sağlamasıdır.

Projede üç aşama için üçer sorgu seçeneği verilmiştir. Birinci sorgu için "En fazla yolcu taşınan 5 gün ve toplam yolcu sayılarının listelenmesi", ikinci sorgu için "İki tarih arasında belirli bir lokasyondan hareket eden araç sayısının gösterilmesi", üçüncü sorgu için ise "Belirli bir gündeki en uzun seyahatin harita üstünde çizdirilmesi" seçeneği tercih edilmiştir.

Bu sorguların hepsi istendiği şekilde tamamlanmış, gün ve lokasyon seçilmesi gereken yerlerde kullanıcıya istenen seçme şansı sunulmuş ve programın çalıştığı teyit edilmiştir.

Dosyaların Filtrelenmesi

Verilen veri dosyaları çok büyük olduğu için çeşitlilik sağlandığı takdirde dosyadan istendiği sayıda kayıt alınıp, sorguların bu yeni dosya üstünden yapılabileceği söylenmiştir. Bu sebeple dosyalardan rastgele veri alınması için iki adet Python scripti yazıldı.

Bu scriptlerden birincisi, gezinti kayıtları dosyasında gezip, her günden istendiği iki değer arasında rastgele kayıt alıp, oluşan veriyi bir JSON dosyasına yazdırmaktadır. Ayrıca bu script, gereksiz olan sütunları silmekte, sütun adlarını ise Javascript için daha uygun olan "Camel Case" formatına çevirmektedir.

İkinci script ise lokasyon dosyasında gezip, Google Maps Api kullanarak her lokasyonun latitude ve longitude değerlerini de yanlarına ekleyip, JSON formatına çevirmektedir. Böylece frontend kısmında yol tarifi sorguları daha kolay yapılmıştır.

Bu scriptlerin proje notuna dahil olmadığı ve istediğimiz gibi filtreleme yapabileceğimiz belirtildiği için bu scriptlerin başkalarının da kullanabilmesi için paylaşıldığının bilinmesini isterim.

Veritabanı İşlemleri

Veritabanı için, AWS (Amazon Web Services) RDS (Relational Database) hizmeti üzerinden Postgresql kullanılması uygun görüldü. Veritabanı oluşturulduktan sonra public kullanıma açıldı ve dışarıdan erişime izin verildi.

Veritabanı işlemleri için Typescript ORM'lerinden (Object-Relational Mapping) Prisma kullanılmıştır. Prisma, tasarladığımız bir model schema'sı üzerinden veritabanı tablolarını oluşturmakta, veritabanına ilk bilgilerin yüklenmesi için seed işlemlerine izin vermekte, sorguların ve

sonuçlarının yazılan uygulama tarafından tahmin edilebilmesi için ise modeller oluşturmaktadır.

Öncelikle tablo modellerini içeren schema dosyası yazıldı ve migration çalıştırılarak bu tabloların veritabanında oluşturulması sağlandı. Daha sonra ise seed dosyası yazıldı ve daha önce belirtilen Python scriptleriyle oluşturulan gezinti ve lokasyon dosyalarındaki veriler veritabanına aktarıldı.

Bundan sonra bir Prisma Client oluşturuldu ve veritabanının backend uygulamasına tam olarak entegre edilmesine olanak tanındı. Bu işlemin ardından tablo adları ve hazır sorgu tipleri kullanılarak sorgular çağırılmakta ve veritabanından elde edilen döndülerdeki tüm değişken tipleri program tarafından bilinmektedir.

Sunucu Yapısı

Backend'de Typescript üzerinden Exress.js frameworkü kullanılmıştır. Sunucudaki tüm route'lar istenen veriler sorgulanıp başarıyla döndürüldüğü takdirde statü kodu olarak 200 ve yanında veriyi, bir aksilik olduğun ise statü kodu olarak 400 ve yanında hata mesajını döndürmektedir. Böylece frontend, gelen veriyi nasıl işleyeceğini bilmektedir.

Express Router kullanılarak iki farklı router oluşturuldu. Bunlardan birincisi "/zones" router'ıdır. Bu router'da tek bir alt route vardır ve bu route'a atılan GET istekleri, veritabanındaki lokasyonlardan herhangi bir gezinti verisindeki kalkış noktası olanlarını JSON listesi olarak döndürmektedir. Bu route, ikinci sorgudaki lokasyon seçme ekranı için lokasyon listesini elde etmek için kullanıldı.

İkinci router ise "/trips" yolu içindir. Bu router'da her sorgu için ayrı birer alt route bulunmaktadır. Birinci sorgunun route'u GET isteklerini, ikinci ve üçüncü sorguların route'u ise POST isteklerini kabul etmektedir. Bu route'lar istek geldiğinde, Prisma aracılığıyla veritabanından istenen verileri çekip, kullanıcıya döndürmektedir.

Frontend Yapısı

Frontend'de de yine Typescript üzerinden React Native kullanılmıştır. React Native ile Typescript birleştirildiğinde en önemli artısı, tüm componentlere giden propların tipleri, hangilerinin opsiyonel ve hangilerinin zaruri olduğu önceden belirlenebilmektedir. Bu sayede de her componentin ihtiyaç duyduğu dış veriyi istediği tipte elde etmesi sağlanmakta ve hata riski azaltılmaktadır.

Burada Typescript kullanılmasının diğer bir artısı ise backend'e atılan isteklerden dönen yanıtlardaki veri tiplerinin tanımlanabilmesidir. Gelen verinin içindeki tüm değerlere otomatik tamamlamayla erişilebilmekte ve yine gelen veriyi işlerken hata riski azalmaktadır.

Farklı sayfalar yaratabilmek için "react-navigation" modülünden faydalanılmıştır. Programda üç ana screen vardır. Ekranın altındaki tablardan screenler arasında geçiş yapılabilmektedir.

İlk sorguda her zaman aynı değer döneceği için veri geldikten sonra yeni bir ekran oluşturmak yerine bulunulan ekrana verinin yansıtılması tercih edildi. Burada verileri listelemek için "react-native-table-component" kullanıldı. Bu sayede backend'den gelen veri basit bir şekilde ekrana yansıtıldı.

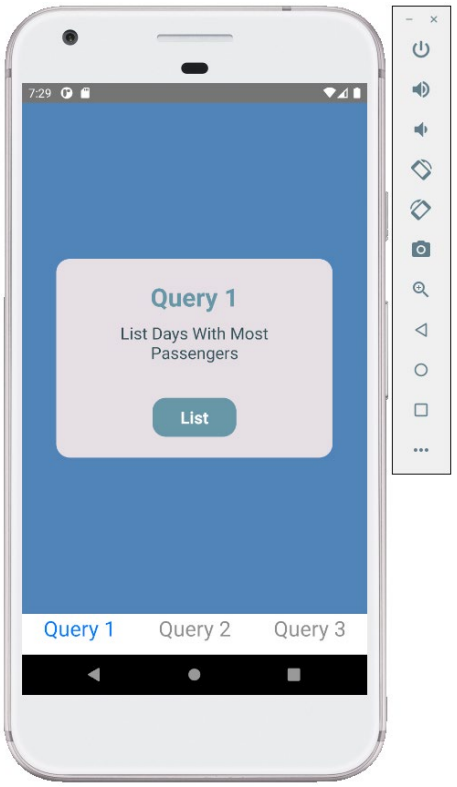
İkinci screen'deki formda ise üç adet input alanı vardır. Birinci alanda lokasyon, ikinci alanda başlangıç tarihi, üçüncü alanda ise bitiş tarihi girilmektedir. Girilen bilgilerle sorgu atıldıktan sonra eğer ki uygun veri bulunduysa yeni bir screene geçilmekte ve yine tabloyla gelen veriler gösterilmektedir. Eğer ki istenen tarih aralığında ve verilen lokasyondan bir veri bulunamadıysa bunun yerine durumu belirten bir uyarı gösterilmektedir.

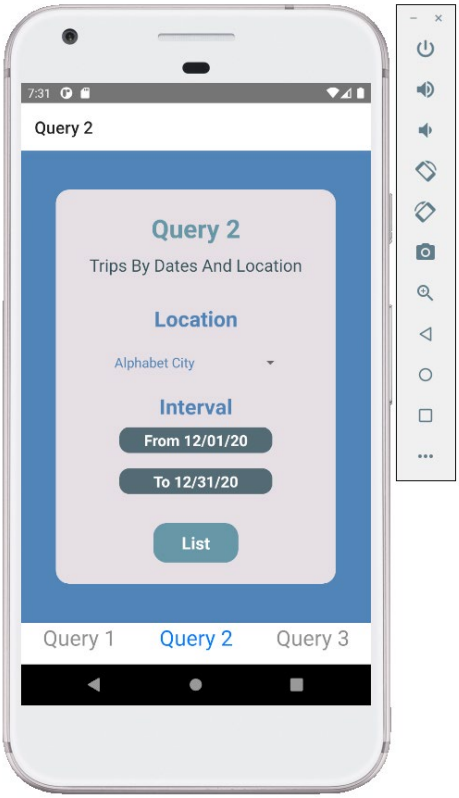
Üçüncü ekranda da bir tarih seçilmekte ve tarih seçildikten sonra belirtilen gün için en uzun yolculuğun bilgisi alınmaktadır. Yolculuğun bilgileri geldikten sonra latitude ve longitude verileriyle harita gösterilmekte ve "react-native-maps-directions" kütüphanesi sayesinde yol çizdirilmektedir. Bu kütüphane, Google Maps Api'daki Directions Api'a konum bilgilerini atıp, gelen verideki Polyline verileri üstünden yol çizimini gerçekleştirmektedir.

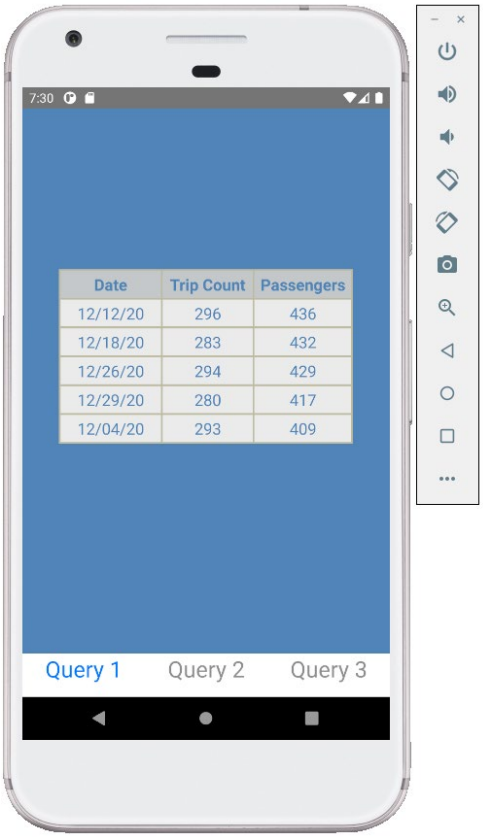
Sonuç

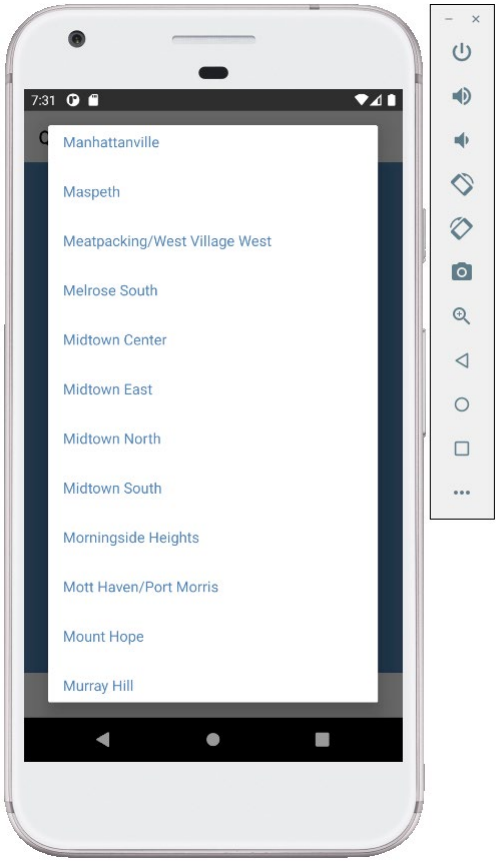
İstenen tüm isterler gerçekleştirilerek tüm sorguları başarıyla gerçekleştiren bir uygulama yazılmış ve çeşitli farklı girdilerle düzgün çalıştığı teyit edilmiştir.

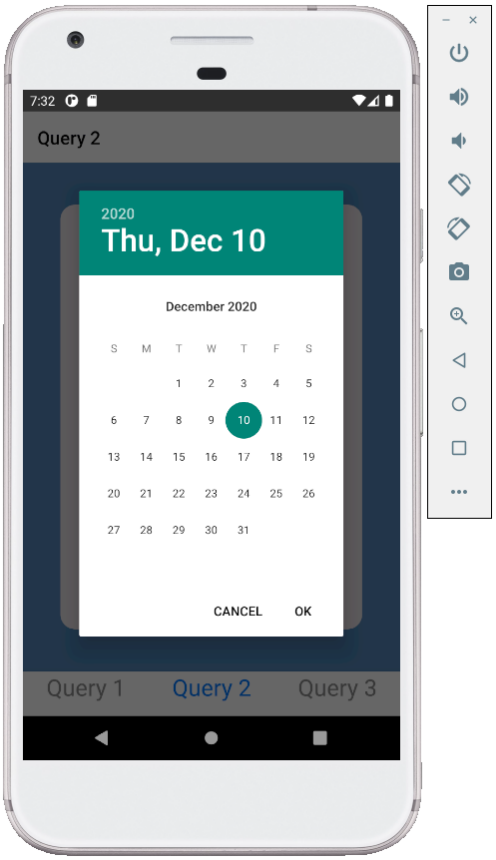
Çıktılar


7:29
Query 1
List Days With Most Passengers
List
Query 1
Query 2
Query 3

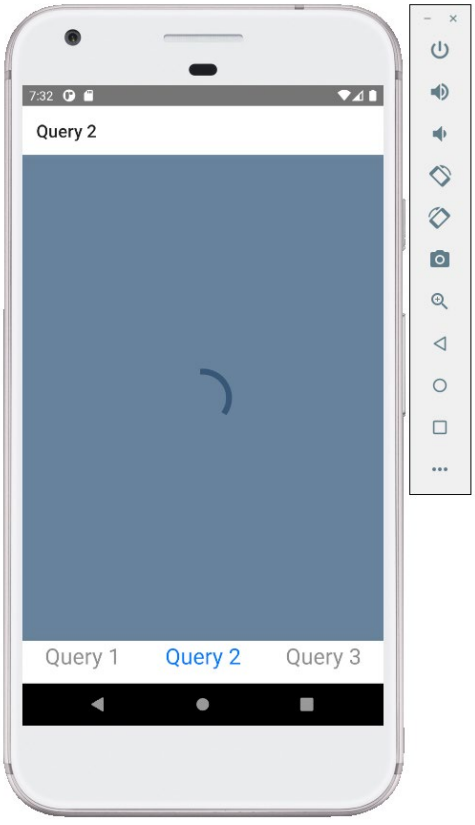

7:31
Query 2
Query 2
Trips By Dates And Location
Location
Alphabet City
Interval
From 12/01/20
To 12/31/20
List
Query 1
Query 2
Query 3

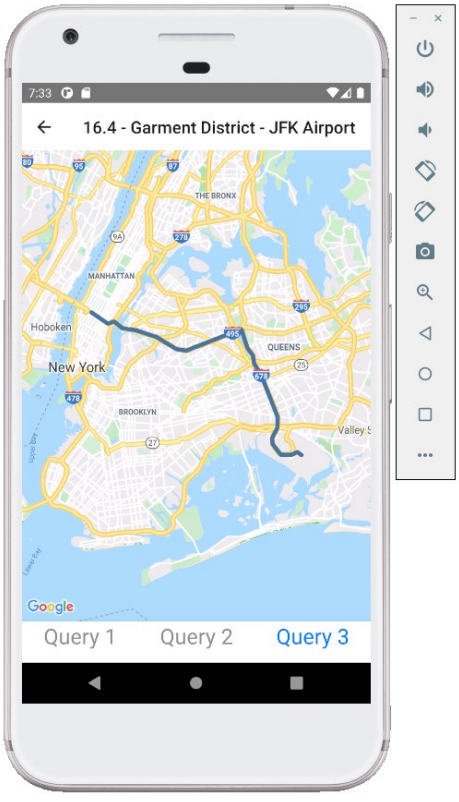

7:30
Date
Trip Count
Passengers
12/12/20
296
436
12/18/20
283
432
12/26/20
294
429
12/29/20
280
417
12/04/20
293
409
Query 1
Query 2
Query 3


7:31
Manhattanville
Maspeth
Meatpacking/West Village West
Melrose South
Midtown Center
Midtown East
Midtown North
Midtown South
Morningside Heights
Mott Haven/Port Morris
Mount Hope
Murray Hill


7:32
Query 2
2020
Thu, Dec 10
December 2020
S M T W T F S
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
CANCEL
OK
Query 1
Query 2
Query 3


7:32
Trips (199)
Date
Pick Up Zone
Drop Off Zone
Distance
Total Amount
12/10/20 Midtown Center Lenox Hill West 1.4 15.95
12/10/20 Midtown Center Upper East Side South 1.4 14.3
12/10/20 Midtown Center Williamsburg (North Side) 5.1 27.8
12/10/20 Midtown Center Upper East Side North 1.76 18.74
12/10/20 Midtown Center Penn Station/ Madison Sq West 1.56 13.92
12/10/20 Midtown Center Steinway 5.7 26.8
12/10/20 Midtown Center Upper West Side South 2.43 22.56
12/11/20 Midtown Center Clinton East 0.86 9.3
Query 1
Query 2
Query 3


7:32
Query 2
Query 1
Query 2
Query 3


7:33
16.4 - Garment District - JFK Airport
THE BRONX
MANHATTAN
Hoboken
New York
QUEENS
BROOKLYN
Google
Query 1
Query 2
Query 3

Kaynaklar

https://www.typescriptlang.org/docs/

http://expressjs.com/en/api.html

https://expressjs.com/en/guide/routing.html

https://stackoverflow.com/questions/18304436/in-express-and-node-js-is-it-possible-to-extend-or-override-methods-of-the-resp

https://christiangiacomi.com/posts/extend-express-typescript-custom-types/

https://www.prisma.io/docs/

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

https://reactnavigation.org/docs/getting-started/

https://reactnavigation.org/docs/tab-based-navigation

https://reactnative.dev/docs/stylesheet

https://reactnative.dev/docs/pressable

https://www.carlrippon.com/react-children-with-typescript/

https://www.typescriptlang.org/docs/handbook/jsx.html

https://reactjs.org/docs/hooks-intro.html

https://reactnative.dev/docs/activityindicator

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

https://javascript.info/async-await

https://reactnative.dev/docs/alert

https://github.com/react-native-maps/react-native-maps

https://github.com/bramus/react-native-maps-directions

https://github.com/react-native-datetimepicker/datetimepicker

https://github.com/react-native-community/react-native-maps/issues/1761

https://www.instamobile.io/react-native-tutorials/react-native-maps/