**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# CBS DESTEKLİ MOBİL DEPREM BİLGİ SİSTEMİ UYGULAMASI: HAZTURK SON DEPREMLER

**YÜKSEK LİSANS TEZİ**

**Berkay ORUÇ**

**Geomatik Mühendisliği Anabilim Dalı**

**Geomatik Mühendisliği Programı**

**HAZİRAN 2022**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ**

# CBS DESTEKLİ MOBİL DEPREM BİLGİ SİSTEMİ UYGULAMASI: HAZTURK SON DEPREMLER

**YÜKSEK LİSANS TEZİ**

**Berkay ORUÇ**
**(501191610)**

**Geomatik Mühendisliği Anabilim Dalı**

**Geomatik Mühendisliği Programı**

**Tez Danışmanı: Prof. Dr. Himmet KARAMAN**

**HAZİRAN 2022**

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 501191610 numaralı Yüksek Lisans Öğrencisi Berkay ORUÇ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "CBS DESTEKLİ MOBİL DEPREM BİLGİ SİSTEMİ UYGULAMASI: HAZTURK SON DEPREMLER" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

| | | |
|---|---|---|
| **Tez Danışmanı :** | **Prof. Dr. Himmet KARAMAN**<br>İstanbul Teknik Üniversitesi | ..... |
| **Jüri Üyeleri :** | **Doç. Dr. Turan Erden**<br>İstanbul Teknik Üniversitesi | ..... |
| | **Doç. Dr. Bahadır Ergün**<br>Gebze Teknik Üniversitesi | ..... |

**Teslim Tarihi : 03.06.2022**
**Savunma Tarihi : 21.06.2022**

***Eşim ve aileme,***

## ÖNSÖZ

Tez kapsamında Türkiye ve dünyada gerçekleşmiş olan depremleri farklı zaman aralıkları için liste ve harita görünümünde kullanıcılara sunan bir uygulama geliştirilmiştir. Uygulamada Türkiye genelinde gerçekleşen depremler için anlık bildirimler gönderilmektedir. Ayrıca İstanbul ili için toplanma ve geçici barınma alanları bir servis ile kullanıcılara sunulmaktadır.

Projenin fikir aşamasından sonuna kadar tüm süreçte bana desteğini esirgemeyen danışman hocam sayın Prof. Dr. Himmet Karaman'a;

Lisans ve lisansüstü eğitimim boyunca ders aldığım tüm hocalarıma;

Fikir, çalışma ve katkılarıyla her zaman yanımda olan değerli dostum Alihan Keskin'e;

Beni bugünlere kadar getiren, her zaman yanımda ve arkamda olan, eksikliklerini asla hissetmediğim canım annem, babam ve kardeşim olmak üzere tüm aileme;

Son olarak uzun yıllardır desteğini, sevgisini her daim benimle paylaşan, hep yanımda olan sevgili eşime şükranlarımı sunarım.

Haziran 2022

Berkay ORUÇ
Geomatik Mühendisi

**İÇİNDEKİLER**

**Sayfa**

## KISALTMALAR

**AFAD** : Afet ve Acil Durum Yönetim Başkanlığı
**API** : Application Programming Interface
**CBS** : Coğrafi Bilgi Sistemi
**GIS** : Geographic Information Systems
**IDE** : Integrated Development Environment
**NPM** : Node Package Manager
**OOP** : Object Oriented Programming
**USGS** : United States Geological Survey

## ÇİZELGE LİSTESİ

**Sayfa**

## ŞEKİL LİSTESİ

**Sayfa**

# CBS DESTEKLİ MOBİL DEPREM BİLGİ SİSTEMİ UYGULAMASI: HAZTURK SON DEPREMLER

## ÖZET

Depremler, insanlık tarihi boyunca yıkıcı etkileri olmuş, öngörülemez ve engellenemez doğal afetlerdir. Ancak bilim gün geçtikçe depremleri daha iyi anlamakta ve etkilerinin en aza indirgenmesini sağlamaya çalışmaktadır. Şehirlerin doğru planlanıp, depreme dayanıklı bir şekilde inşa edilmesi olumsuz etkileri azaltmakta yardımcı olmaktadır. Yeterli sayıda toplanma ve geçici barınma alanlarının oluşturulması, deprem gerçekleştikten sonra depremzedelerin artçı sarsıntılar ve hasarlı yapılardan daha az etkilenmelerine yardımcı olduğu gibi, yetkililerin duruma müdahale hızlarına katkı sağlayacaktır. Ayrıca toplumun düzenli olarak deprem konusunda eğitilip, bilinçlendirilmesi de depremin etkilerinin azaltılması anlamında büyük önem arz etmektedir. Türkiye gibi deprem bölgesinde yaşayan toplumlar için resmi kaynaklardan aktarılan deprem öncesi, sırası ve sonrasında yapılması gerekenler hakkında bilgiler, toplanma ve geçici barınma alanları bilgilerinin kamuoyuna aktarılması gerekmektedir. Böylelikle toplum, afet durumunda daha az panik yapıp, önlenebilecek olası can ve mal kayıplarının önüne geçebilecektir.

Gelişen teknoloji sayesinde akıllı telefonlar toplumun çoğunluğunda kabul görmekte ve kullanılmaktadır. Akıllı telefonların gün geçtikçe daha güçlü ve verimli olmasıyla mobil uygulamalar günlük hayatın merkezinde bulunmaktadır. Google Play ve Apple Store gibi mağazalardan indirilebilen mobil uygulamalar kullanıcıların günlük ihtiyaçlarını karşılamanın yanı sıra, faydalı kaynakların kullanıcıya ulaştırılmasına da yardımcı olmaktadırlar.

Projenin temel amacı, toplumu deprem hakkında bilinçlendirirken, gerçekleşmiş depremleri en yalın hali ile kullanıcı ile buluşturmaktır. Bu kapsamda Türkiye'de gerçekleşmiş depremler için Boğaziçi Üniversitesi Kandilli Rasathanesi ve Deprem Araştırma Enstitüsü Bölgesel Deprem-Tsunami İzleme ve Değerlendirme Merkezi'nin sunduğu veriler kullanılmaktadır. Ek olarak dünya çapında gerçekleşmiş depremler için U.S. Geological Survey tarafından veriler uygulama içinde yer almaktadır.

Türkiye'de gerçekleşen depremler için akıllı telefonlar üzerinden kullanıcının seçtiği büyüklüğe göre anlık bildirimler gönderilerek, kullanıcılar gerçekleşmiş depremlerden haberdar olmaktadır. Gerçekleşmiş depremler liste görünümde belirli parametrelere göre sıralanıp, filtrelenebilmesinin yanı sıra harita üzerinde de kullanıcı ile paylaşılmaktadır. Depremler gerçekleştiği büyüklüğe göre renklendirilerek daha anlaşılır bir görünüm elde edilmeye çalışılmıştır. Ayrıca harita bileşenine fay hatlarının da eklenmesiyle kullanıcı ilgili bilgilere tek bir uygulama üzerinden ulaşabilmektedir.

Bireylerin afet sonrasında can güvenliklerini sağlayabilmeleri için belirlenen toplanma ve geçici barınma alanları tez kapsamında İstanbul ili özelinde uygulamaya eklenmiştir. Kullanıcı uygulamaya konum izini vererek en yakınındaki toplanma veya geçici barınma alanını konum analizi algoritmaları kullanılarak bulabileceği gibi harita üzerinden de ilgili alanları seçip akıllı telefonuna kaydedebilecektir. Kullanıcı bu alanları istediği kişilerle de paylaşabilecektir. Ek olarak kullanıcı belirlediği bir konumdan toplanma veya geçici barınma alanına yol tarifini offline olarak akıllı telefonunda saklayıp afet anında kullanabilecektir. Uygulamanın bu özelliği ile hem afet öncesi bilinçlenme sağlanacak hem de afet sırasında veya sonrasında internet erişimi olmasa dahi yol yönlendirmelerine akıllı telefonlar üzerinden ulaşılabilecektir.

## GIS SUPPORTED MOBILE EARTHQUAKE INFORMATION SYSTEM APPLICATION: HAZTURK LAST EARTHQUAKES


### SUMMARY

Earthquakes are unpredictable and unpreventable natural disasters that have had devastating effects throughout human history. However, science is trying to better understand earthquakes day by day and try to minimize their effects. The correct planning of the cities and the construction of earthquake-resistant ones helps to reduce the negative effects. Establishing sufficient number of gathering and temporary shelters will not only help earthquake survivors to be less affected by aftershocks and damaged structures after the earthquake has occurred, but also will contribute to the speed of response of the authorities to the situation. In addition, it is of great importance to regularly educate and raise awareness of the society about earthquakes in terms of reducing the effects of the earthquake. For communities living in earthquake zones such as Turkey, information about what to do before, during and after the earthquake, as well as information about assembly and temporary shelters, transferred from official sources, should be shared with the public. Thus, the society will be able to panic less in case of disaster and prevent possible loss of life and property that can be prevented.

Thanks to the developing technology, smart phones are accepted and used by the majority of the society. With smartphones becoming more powerful and efficient day by day, mobile applications are at the center of daily life. Mobile applications that can be downloaded from stores such as Google Play and Apple Store not only meet the daily needs of users, but also help deliver useful resources to the user.

The main purpose of the project is to raise awareness of the society about earthquakes and to bring together the earthquakes that have taken place with the user in their simplest form. In this context, data provided by Boğaziçi University Kandilli Observatory and Earthquake Research Institute Regional Earthquake-Tsunami Monitoring and Evaluation Center are used for earthquakes that took place in Turkey. In addition, data from the US Geological Survey for earthquakes that have occurred worldwide are included in the application.

Users are informed about earthquakes that have occurred in Turkey by sending instant notifications according to the magnitude selected by the user via smart phones. In

addition to being able to be sorted and filtered according to certain parameters in the list view, the earthquakes that have occurred are also shared with the user on the map. Earthquakes have been colored according to the magnitude of their occurrence, and a more understandable view has been tried to be obtained. In addition, by adding fault lines to the map component, the user can access the relevant information through a single application.

Gathering and temporary accommodation areas determined for individuals to ensure their life safety after the disaster have been added to the application in the province of Istanbul within the scope of the thesis. By giving the location permission to the application, the user will be able to find the nearest gathering or temporary accommodation area using location analysis algorithms, as well as select the relevant areas on the map and save them to her/his smartphone. The user will also be able to share these fields with anyone he wants. In addition, the user will be able to store the directions from a designated location to the assembly or temporary accommodation area offline on his smart phone and use them in case of disaster. With this feature of the application, both pre-disaster awareness will be raised and road directions will be accessible via smart phones even if there is no internet access during or after the disaster.

# 1. GİRİŞ

Deprem, tarih boyunca insanoğlunun anlamaya çalıştığı, büyük yıkımlara neden olmuş bir doğal afet çeşididir. Meydana geleceği zaman ve büyüklüğü kestirilemeyen bu doğal afetin nedenleri görece ortaya konulmuş olsa da tam manasıyla bir önlem alınamamaktadır. Ülkemiz bulunduğu coğrafi konum nedeniyle sık sık deprem gerçeği ile karşılaşmakta, kimi zaman bu gerçeklik büyük can ve mal kayıplarına yol açmaktadır. Kandilli Rasathanesi verilerine göre can kaybı ve hasarlı bina sayısına göre cumhuriyet dönemi Türkiye'sinde birçok yıkıcı deprem meydana gelmiştir (URL-1).

**Çizelge 1.1 :** Cumhuriyet dönemi gerçekleşen yıkıcı bazı depremler.

| TARİH | YER | MAG (Ms) | CAN KAYBI | HASARLI BİNA |
|---|---|---|---|---|
| 27.12.1939 | ERZİNCAN | 7.9 | 32968 | 116720 |
| 20.12.1942 | ERBAA | 7.0 | 3000 | 32000 |
| 27.11.1943 | LADİK | 7.2 | 4000 | 40000 |
| 19.08.1966 | VARTO | 6.9 | 2396 | 20007 |
| 17.08.1999 | GÖLCÜK | 7.8 | 17480 | 73342 |
| 12.11.1999 | DÜZCE | 7.5 | 763 | 35519 |
| 23.10.2011 | VAN | 7.2 | 644 | 17005 |

Ülke insanının deprem konusunda bilgisinin az olması, yeterli dayanıklılığa ve standartlara sahip konutların inşa edilmemesi gibi sebeplerden dolayı bu depremlerin etkisi daha da fazla olmuştur. Özellikle 1999 yılında meydana gelen 17 Ağustos Kocaeli ve 12 Kasım Düzce depremleri, Türkiye'nin depreme ne kadar hazırlıksız olduğunu acı bir şekilde göstermiştir.

1999 yılında karşı karşıya kalınan yıkım sonrası depremlere karşı hazırlıklı olmanın ne denli önemli olduğu anlaşılmış ve bu anlamda adımlar atılmaya başlanmıştır. Yapı Güçlendirme Yönetmeliği hazırlanmış, toplumu deprem konusunda bilinçlendirmek adına devlet ve sivil toplum kuruluşları vasıtası ile etkinlikler ve çalışmalar düzenlenmiştir.

## 1.1 Tezin Amacı

Günümüzde teknolojinin geldiği nokta dikkate alındığında akıllı telefonlar ve kullandığımız mobil uygulamalar günlük hayatımızın ayrılmaz birer parçasıdır. Bankacılık sektöründen, e-ticaret uygulamalarına, sosyal medyadan, dil öğrenme uygulamalarına kadar çok geniş bir yelpazede geliştirilen uygulama bulunmaktadır. Akıllı telefonların sağladığı internet bağlantısı, konum bilgisi gibi özellikler kullanıcıların günlük ihtiyaçlarını karşılayacak kapsamlı uygulamaların geliştirilmesine olanak sağlamaktadır.

Hazılanan tez çalışmasında, kullanıcıların Türkiye'de ve dünyada meydana gelen depremleri mobil bir uygulama vasıtası ile kolaylıkla takip edebilmelerine ek olarak, İstanbul özelinde bulunan toplanma ve geçici barınma alanlarına kolay bir şekilde ulaşabilmeleri hedeflenmiştir. Ayrıca Türkiye'de meydana gelen depremler için kullanıcılara seçtikleri deprem büyüklüğüne göre anlık bildirim gönderilecektir.

Yapılan araştırmalar sonucunda kullanıcıların gerçekleşen depremleri liste görünümü haricinde harita üzerinde de görmek istedikleri tespit edilmiş, geliştirilen mobil uygulamaya harita bileşeninin de eklenmesi hedeflenmiştir.

Tez kapsamında Google Play Store üzerinden Flutter ile geliştirilmiş Android bir uygulama yayınlanmıştır. Harita bileşeni için Flutter frameworkü için özelleştirilmiş açık kaynak "flutter_map" kütüphanesi kullanılmıştır.

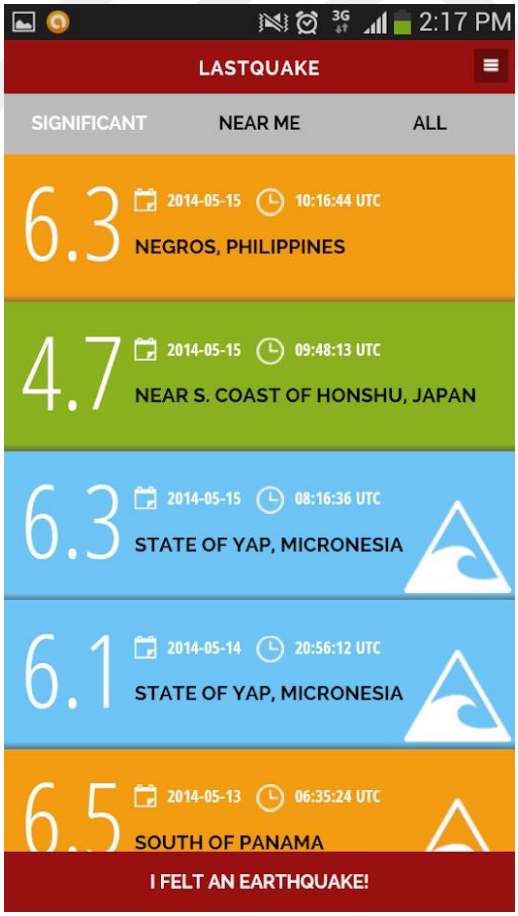
## 1.2 Literatür Araştırması

Literatür araştırması, Google Play Store üzerinde yayınlanmış resmi enstitülere ait veya kişisel olarak geliştirilmiş uygulamalar incelenerek yapılmıştır. Birbirinden farklı birçok uygulama incelenmiş, kullanıcılar tarafından ilgi görmüş veya diğer uygulamalardan bazı özellikleri ile ayrılanlar detaylı bir şekilde ele alınmıştır.

### 1.2.1 LastQuake (EMSC-CSEM)

EMSC-CSEM tarafından geliştirilen uygulama Android işletim sistemine sahip akıllı telefon kullanıcıları için deprem konusunda oldukça popüler bir konuma sahiptir. 1.000.000+ indirme ve 30.000+ yorumdan 4.7/5 puan ortalamasına sahip uygulama, kullanıcılara deprem hakkında uyarı yapmak ve barındırdığı "Depremi Hissetim" özelliği ile gerçek zamanlı veri toplayarak, meydana gelen depremin etkilerini tahmin etmeyi amaçlamaktadır.

Kullanıcıların seçilen deprem ile ilgili yorum yapıp, fotoğraf gönderebildikleri bir "Detaylar" sayfası bulunmaktadır. Ayrıca "Bu Depremi Hissettim" özelliği ile kısa bir anket yapılıp depremin etkisi tahmin edilmeye çalışılmaktadır. Seçilen deprem, "Paylaş" özelliği sayesinde farklı platformlarda paylaşılabilmektedir. Harita görünümü de mevcut olan uygulama, "Güvenlik Önlemleri" seçeneği ile deprem ve tsunami afetleri hakkında bilinmesi ve dikkat edilmesi gereken bilgileri görseller ile kullanıcılara sunmaktadır (URL-2).




**Şekil 1.1 :** LastQuake uygulaması.

### 1.2.2 Deprem bilgi sistemi

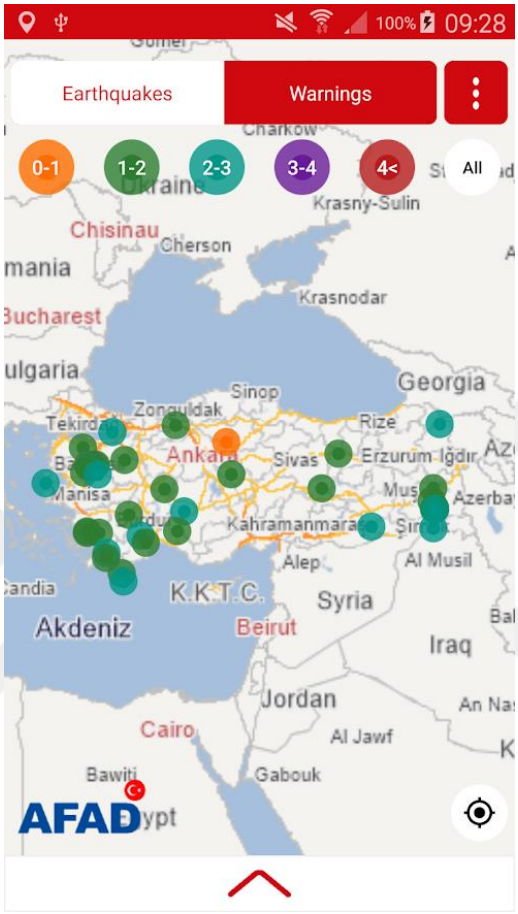
Kandilli Rasathanesi tarafından sunulan uygulama, 2019 yılında Cenk Tarhan tarafından lisansüstü bitirme projesi kapsamında geliştirilmiştir. Uygulama açıldığında "En son depremler", "Deprem Tarihçesi", "Deprem oldu!" ve "Deprem Geçmişi" seçenekleri kullanıcıyı karşılamaktadır. Depremler için hem harita görünümü hem de liste görünümü kullanıcıya sunulmaktadır. "Depremi Hissettim!" seçeneği ile kullanıcıdan 3.0 ve üzeri büyüklükte meydana gelen deprem hakkında anket yapılarak veri toplanmaktadır. Kandilli Rasathanesi ve Deprem Araştırma Enstitüsü tarafından toplanan bu veriler ile depremin nerede hissedildiği ve etkilerinin neler olduğu hakkında çalışmalar yapılabilmektedir (URL-3).


**Şekil 1.2 :** Deprem Bilgi Sistemi 3.0 uygulaması.

### 1.2.3 eAfad

Afet ve Acil Durum Yönetim Başkanlığı tarafından geliştirilen uygulama ile afet ve acil durum bildirimleri ve alınabilecek önlemler kullanıcıya sunulmaktadır. Gerçekleşmiş depremlerin bulunduğu bir harita ile açılan uygulamada, üst menüden depremlerin haricinde diğer afetler için de uyarılara ulaşılabilmektedir. 3.5 ve üzeri

meydana gelen depremler için kullanıcı anlık bildirimler alabilmektedir. Seçilen deprem hakkında ayrıntılı bilgilerin kullanıcılara sunulmasının yanı sıra, ilgili deprem farklı platformlar üzerinde paylaşılabilmektedir. Ayrıca kullanıcı toplanma alanlarına harita üzerinden ulaşabilmektedir (URL-4).




**Şekil 1.3 :** eAfad uygulaması.

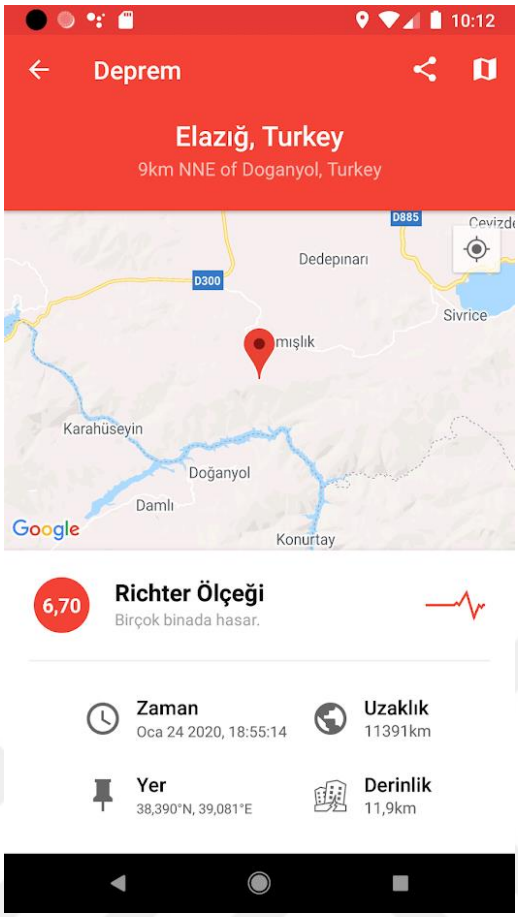
### 1.2.4 Afad acil çağrı

Afet ve Acil Durum Yönetim Başkanlığı tarafından sunulan uygulama, afet durumunda kullanıcının acil çağrı yapabilmesi için geliştirilmiştir. TC kimlik numarası ile kayıt yapılan uygulamada kullanıcının konum izni alınarak harita üzerinde en yakın toplanma alanları gösterilmektedir. Ayarlar ekranından kullanıcının yakınlarının iletişim bilgilerinin eklenebildiği uygulamada, afet farkındalık eğitimi ve diğer eğitim videoları da bulunmaktadır (URL-5).


**Şekil 1.4 :** Afad acil çağrı uygulaması.

### 1.2.5 Deprem uyarılarım

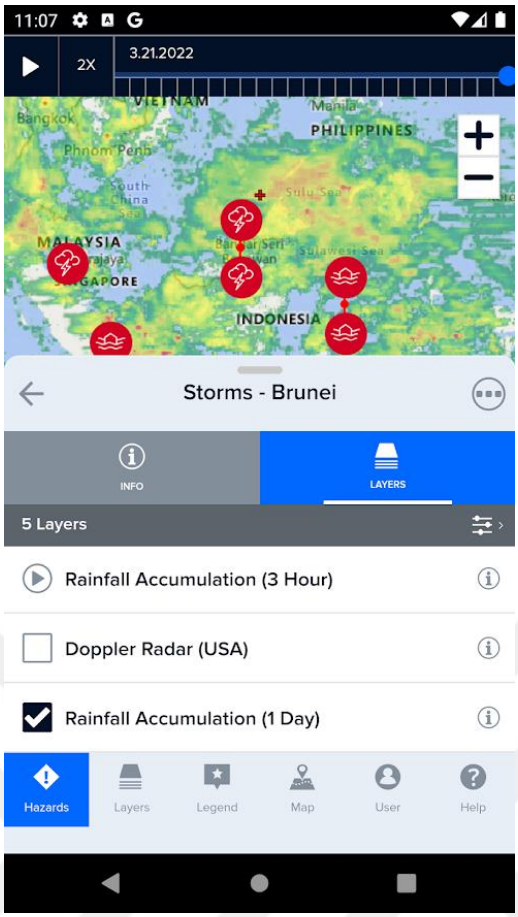
jRustonApps B.V. tarafından bağımsız geliştirilen uygulama, dünyada gerçekleşen depremleri kullanıcı ile buluşturmayı hedeflemektedir. Anasayfada kullanıcıları ikiye ayrılmış bir ekran, harita ve liste görünümü ile karşılamaktadır. Bölge ve zaman aralığı ile gerçekleşmiş depremlerin görüntülenebildiği uygulamada anlık bildirimler ile kullanıcının belirlediği kriterlere göre deprem bilgisi aktarılmaktadır. Ayrıca harita üzerinde dünya çapındaki büyük fay hatları da gösterilmektedir (URL-6).




**Şekil 1.5 :** Deprem Uyarılarım uygulaması.

### 1.2.6 Disaster alert

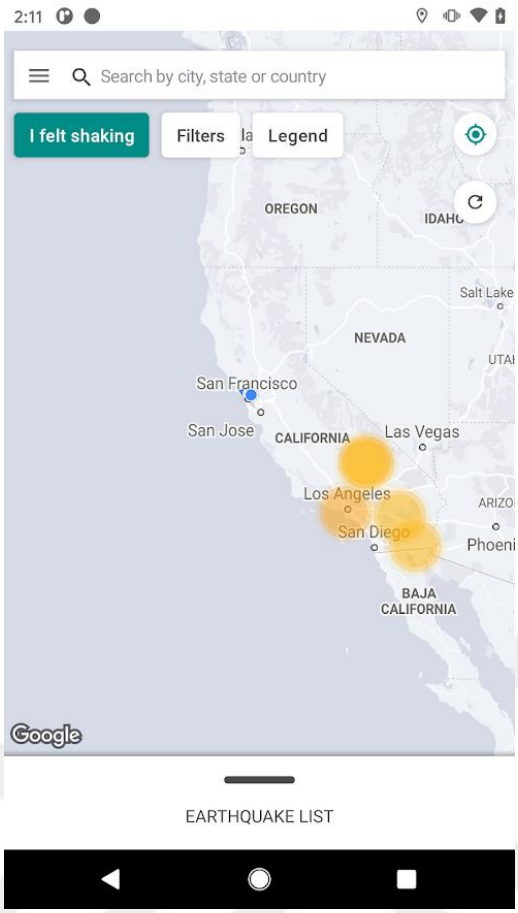
Pacific Disaster Center tarafından yayınlanan uygulama, dünya çapında gerçekleşen onsekiz farklı türdeki kullanıcının yakınında gerçekleşen doğal afet için bildirim göndermek amaçlanarak geliştirilmiştir. Kullanıcı anlık bildirim almak istiyorsa uygulama içinden hesap oluşturması gerekmektedir. Gerçekleşen doğal afetler harita üzerinde gösterildiği gibi liste halinde de kullanıcıya sunulmaktadır (URL-7).




**Şekil 1.6 :** Disaster Alert uygulaması.

### 1.2.7 MyShake

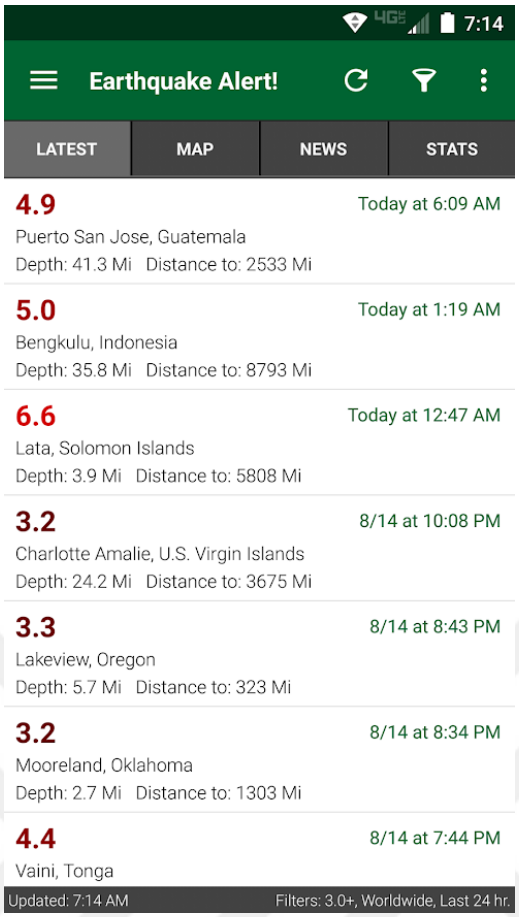
Kaliforniya, Oregon ve Washington eyaletleri gerçekleşmesi beklenen depremler için UC Berkeley Sismoloji Laboratuvarı tarafından erken uyarı sistemi amacı ile geliştirilmiştir. Akıllı telefonların hareket sensörleri sayesinde toplanan verilerin anlamlandırılıp olası 4.5 ve üzeri büyüklüğe sahip depremler önceden tespit edilmeye çalışılmaktadır. Uygulamada gerçekleşmiş depremlerin harita görünümünün dışında, gerçekleşen deprem hakkında kullanıcı tarafından tamamlanması istenen bir de anket özelliği bulunmaktadır. Ayrıca doğal afetler karşısında uygulanması gereken güvenlik ipuçlarına da uygulamadan erişilebilmektedir (URL-8).




**Şekil 1.7 :** MyShake uygulaması.

### 1.2.8 Earthquake alert!

Josh Clemm tarafından yayınlanan uygulama, U.S. Geological Survey tarafından sunulan deprem verilerini kullanıcıya ulaştırmaktadır. Birleşik Devletler'de meydana gelen 1.0+, dünyada ise 4.5+ büyüklüğünde meydana gelen depremler uygulamada bulunmaktadır. Harita ve liste görünümünde sunulan deprem bilgilerinin yanı sıra, dünyada meydana gelen deprem haberleri ve bazı genel deprem istatistikleri uygulamaya entegre edilmiştir. Son olarak meydana gelen depremler hakkında anlık bildirimler gönderilmektedir (URL-9).




**Şekil 1.8 :** Earthquake Alert! uygulaması.

### 1.2.9 ABİS – Afet bilgi sistemi

Beylikdüzü Belediyesi tarafından yayınlanan mobil uygulama ile kullanıcıların meydana gelebilecek depremlere karşı hazırlıklı olmaları amaçlanmıştır. Uygulamanın 1.1.3 sürümünde "Acil Durum" butonu, deprem öncesi hazılanabilecek "Acil Durum Planım", acil durumlarda hızlıca aranabilecek "Aranacak Kişiler", kullanıcının belirlediği adrese göre kullanıcıya en yakın toplanma alanlarını gösteren "Toplanma Alanlarım", "Deprem Çantam" ve depremler ile ilgili bilgilerin bulunduğu "Bunları Unutmayın" seçenekleri bulunmaktadır. Uygulama Beylikdüzü özelinde geliştirildiği için kısıtlı bir very sağlamaktadır. Ayrıca kullanıcılar yalnızca tanımladıkları adrese göre en yakın toplanlanma alanlarına ulaşabilmektedirler (URL-10).


**Şekil 1.9 :** ABİS – Afet Bilgi Sistemi uygulaması.

# 2. ANLIK BİLDİRİM SİSTEMİ

OneSignal üzerine tasarlanan anlık bildirim sistemi tez kapsamında geliştirilen mobil uygulama için önemli özelliklerden biridir. Türkiye'de gerçekleşmiş depremler için Kandilli Rasathanesi tarafından servis edilen verilerin belirli bir otomasyon ile kontrol edilerek akıllı cihazlara ulaştırılması hedeflenmektedir.

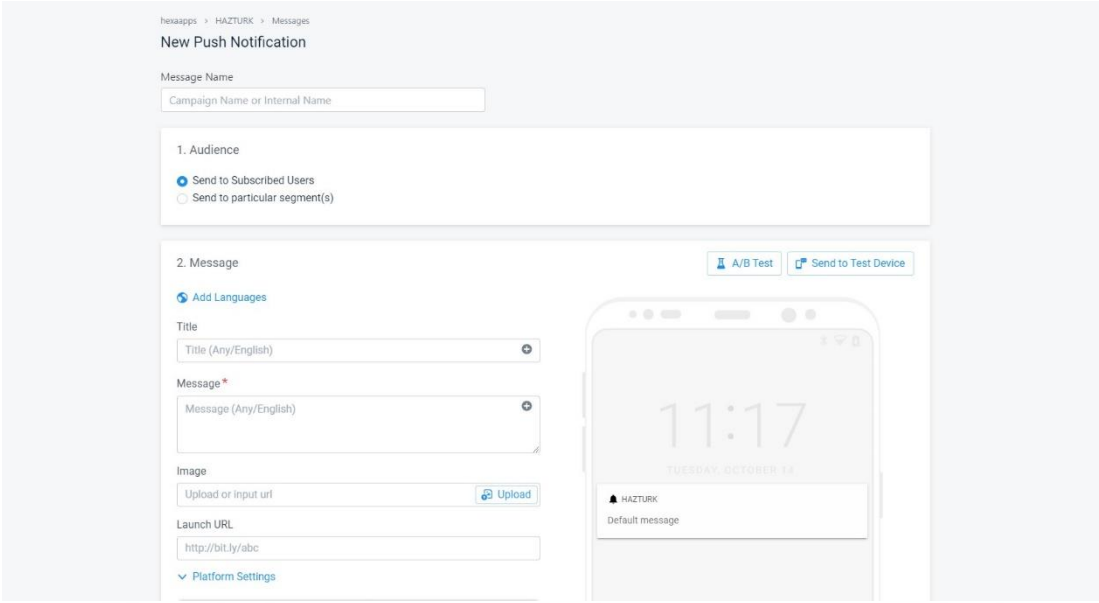
## 2.1 Otomasyon Sistemi

OneSignal üzerinden gönderilecek anlık bildirimler, OneSignal kontrol panelinden insan gözetimi ve kontrolünde gönderilebilse de sürdürülebilir bir yöntem değildir. Meydana gelmiş depremlerin kaynak servisinin devamlı ve stabil belirli aralıklarla kontrol edilmesi mümkün değildir. Bu aşamada anlık bildirimler için bir yazılım projesine ihtiyaç duyulmaktadır. Geliştirilecek otomasyon projesi ile deprem servisinden belirli aralıklar ile yeni deprem meydana gelip gelmediği kontrol edilecektir.

Otomasyon uygulamasına herhangi bir ortamdan istek gönderilmeyeceği için uygulamanın statik bir ipye ihtiyacı bulunmamaktadır. Bu yüzden uygulama javascript dili ile docker ortamı için geliştirilmiştir.

Docker, platformdan bağımsız tek bir sunucudan birden çok uygulamanın birbirini etkilemeden çalışan bir platformdur (URL-11). Docker üzerinden sunulan uygulamalar izole bir şekilde çalıştığı için etkileşime girmesi istenmeyen uygulamalar diğerlerine herhangi bir etki etmeden çalışmaktadırlar.

Otomasyon sistemi geliştirilmeye başlamadan önce anlık bildirim servisi olarak kullanılacak OneSignal üzerinden istek nasıl atılır araştırılmıştır. OneSignal'ın kayıtlı kullanıcılarına özgü kontrol panelinden bu istek atılabilmektedir.




**Şekil 2.1 :** OneSignal kontrol paneli sayfası.

Şekil 3.1'de görüldüğü üzere anlık bildirimin gönderileceği kullanıcı grubu, bildirimin başlığı, içeriği ve birçok farklı seçeneği sağda bulunan illustrasyon üzerinde örneklendirilerek gönderilebilmektedir. Meydana gelme zamanı tahmin edilmesi zor olan depremler için, deprem servisinin sürekli takibi ve bu panel üzerinden bildirim gönderilmesi oldukça güçtür. Bu yöntem yerine yazılım ile süreç otomatik hale getirilmiştir.

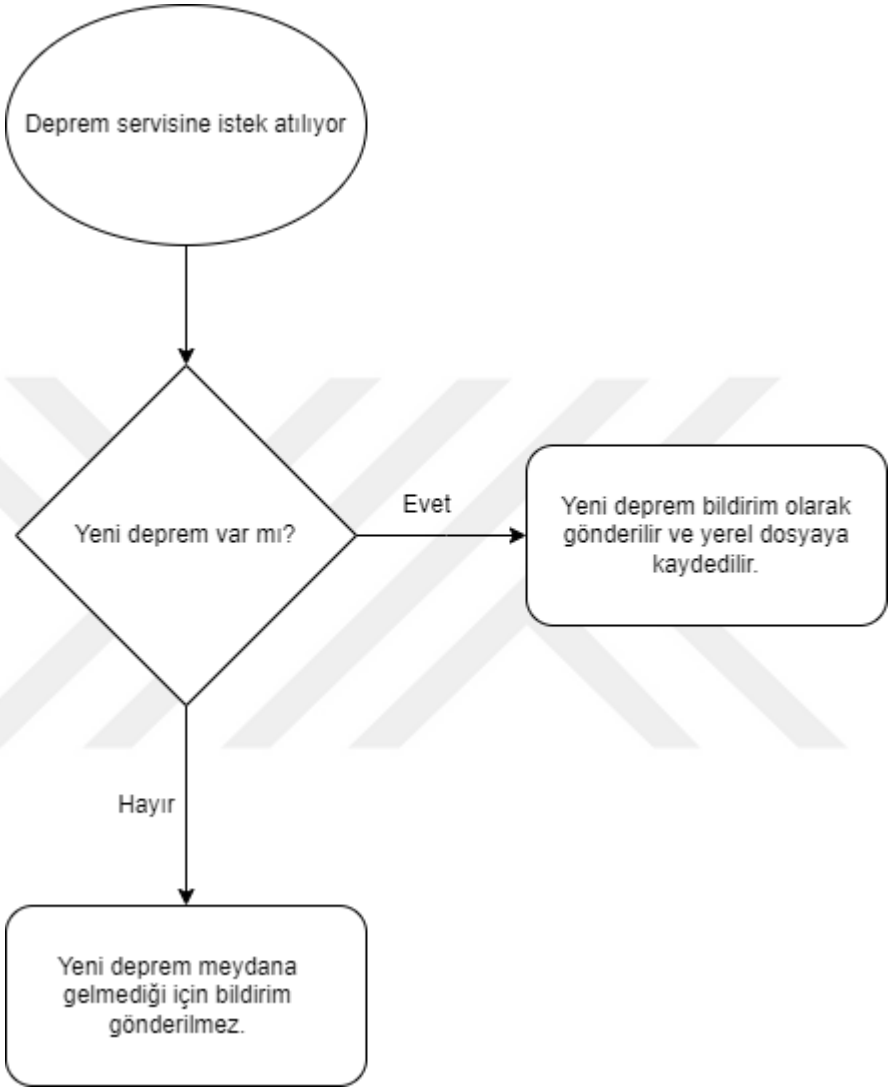
## 2.2 Geliştirme

Uygulama platform bağımsız bir docker containerı üzerinden çalışacağı için konfigürasyon dosyası olan "Dockerfile" oluşturulmuştur. Node ortamında yazılacak olan uygulama için konfigürasyon dosyasında sırasıyla; node ortamı aktifleştirilmiş, "app" klasörü çalışma klasörü olarak belirlenmiş, projede bulunan dosyalar bu klasöre aktarılmış, npm paketleri yüklenmiş ve son olarak node projesinde "index.js" dosyası çalıştırılmıştır. (Ek – A.1)

Node.js projelerinde uygulamada kullanılacak paketler, komutlar, proje metadatası gibi bilgiler "package.json" dosyasında bulunmaktadır. Proje kapsamında servisten gelen verinin işlenmesi için cheerio, ağ üzerinden yapılacak istekler için ise node-fetch paketi kullanılmıştır. (Ek – A.2)

Her iki dakikada bir ilgili deprem servisini kontrol eden sistem, gerçekleşmiş yeni depremleri "previousEarthquake.json" dosyasında daha önce kaydedilmiş depremler

ile karşılaştırmakta ve ilgili deprem bu dosyada yoksa bu dosyaya yazılmakta ardından Ek – A.3'de tanımlanan fonksiyon ile OneSignal servisine "POST" metodu ile istek gönderilmektedir. İstek ile RestKey, uygulama id gibi hassas veriler gönderildiği için daha güvenli olan "POST" metodu "GET" metoduna tercih edilmiştir.




**Şekil 2.2 :** Anlık bildirim çalışma prensibi.

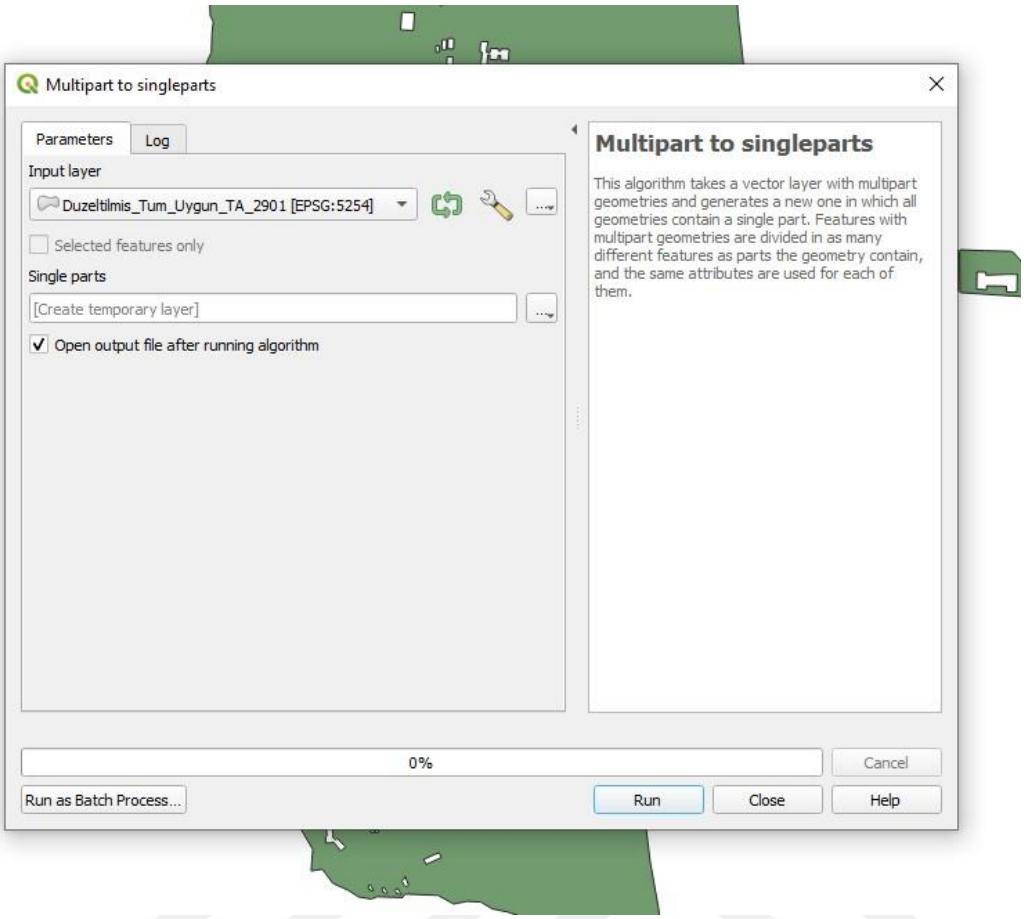
# 3. TOPLANMA VE GEÇİCİ BARINMA ALANLARI SERVİSİ

Deprem sırası veya sonrasında insanlar hayatta kalma içgüdüsüyle bulundukları kapalı alanlardan, daha güvenli olduğunu düşündükleri açık alanlara doğru yönelmektedirler. Ancak bilinçsiz olarak herhangi bir alanda toplanmak hem afet sonrası çalışma ekiplerinin işlerini zorlaştırabilir hem de güvenli olarak düşünülen konuma giden vatandaşların can güvenliklerini tehlikeye atabilir. Bu kapsamda afet sonrası çalışma ekiplerinin afetzedelere hızlı bir şekilde ulaşabilecekleri ve afetzedelerin de güvenli bir şekilde afetten korunabilecekleri alanlara ihtiyaç duyulmaktadır. Yeterli kapasitede, afetzedelerin güvenliğini sağlayacak ve en önemlisi ilgililerin afetzedelere en hızlı şekilde ulaşabilecekleri alanlar toplanma alanları olarak tanımlanmaktadır. (Özkılıç, 2020)

Tez kapsamında geliştirilen mobil uygulamada kullanılmak üzere, İstanbul ilinde bulunan toplanma ve geçici barınma alanlarını sunan bir veri servisine ihtiyaç duyulmaktadır. Bu ihtiyacı karşılamaya yönelik toplanma ve geçici barınma alanlarına ait veriler bir PostgreSQL veritabanında depolanıp, bir servis yazılım uygulaması ile mobil uygulamaya güvenli bir şekilde sunulmaktadır.

## 3.1 Verinin İşlenmesi

İstanbul ili özelinde yapılan çalışma için bulunan toplanma ve geçici barınma alanı verileri oluşturulurken bir CBS geometri türü olan multipoligon olarak üretilmiştir. Multipoligon geometri türünde olan veriler bir grup poligon geometrisinin tek bir geometri ile gösterilmesini sağlamaktadır. Burdaki temel amaç birbirinden bağımsız geometrilere sahip ancak tek bir veriyi simgeleyen objeler için doğru geometriyi gösterebilmektir. Farklı birçok adadan oluşan ülke geometrisi multipoligon geometri türüne örnek olarak gösterilebilir (URL-12). Ancak multipoligon türündeki geometriler ile konumsal analizler yapıldığında istenilen sonuca ulaşmakta problemler ile karşılaşılabilmektedir. İncelenen veride multipoligon türündeki geometri yapısına tez kapsamında ihtiyaç olmadığı görülmüştür. QGIS masaüstü uygulaması yardımı ile Esri firmasının sunduğu dosya yapısı olan Shapefile türündeki veri ilgili konumsal analiz aracı kullanılarak multipoligondan poligon geometri türüne dönüştürülmüştür.




**Şekil 3.1 :** Konumsal analiz QGIS.

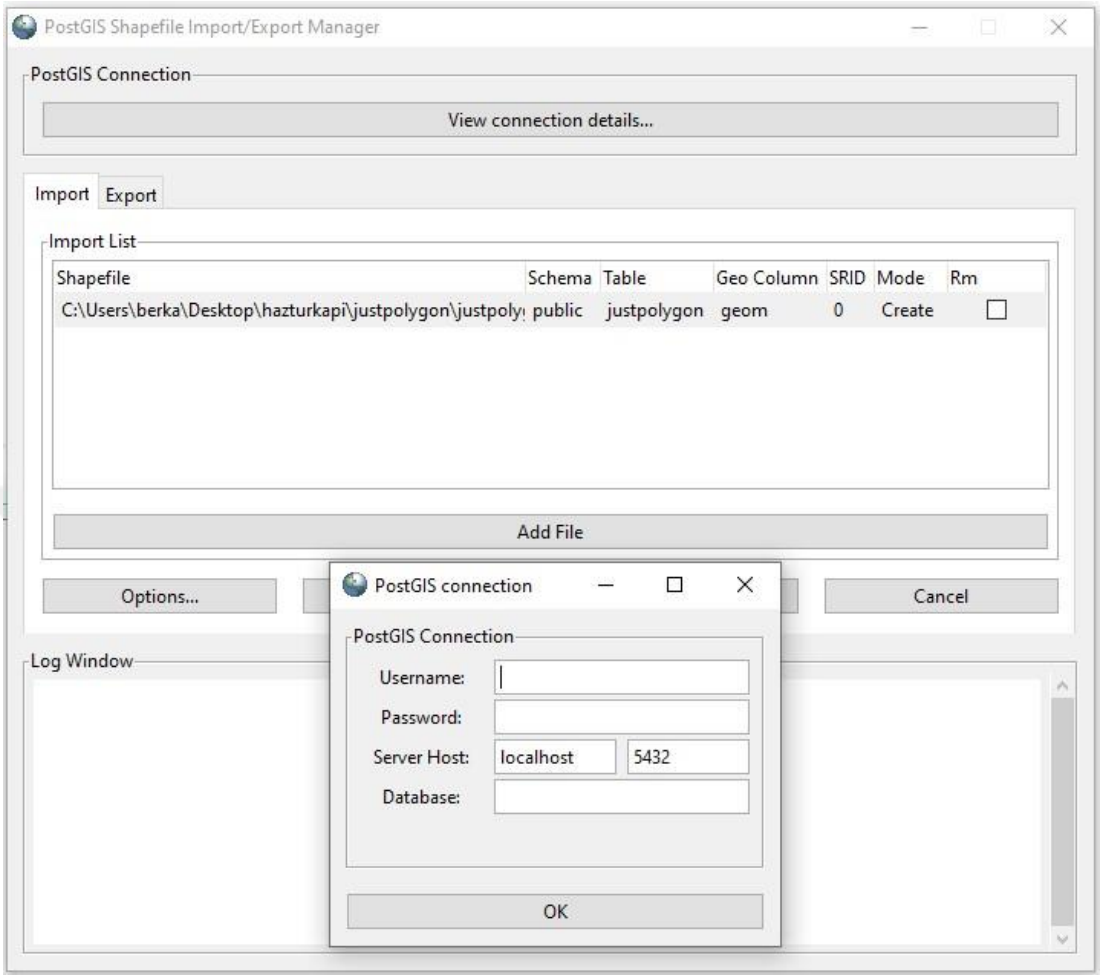
Geometri dönüşümü sağlanan veri daha sonra kullanılmak üzere Shapefile olarak kaydedilmiştir.

## 3.2 Veritabanı

Toplanma ve geçici barınma alanları verisinin bir servis ile mobil uygulamaya sunulabilmesi için verinin bir veritabanından saklanması gerekmektedir. Gelişen teknoloji ekseninde birçok farklı veritabanı ortamı geliştirilmiştir. Tez kapsamında güvenli, hızlı ve en önemlisi konumsal analizlerin yapılabildiği bir veritabanı ortamı seçilmesi gerekliliği ortaya çıkmıştır. PostgreSQL, açık kaynaklı bir proje olması, güvenli, hızlı ve en önemlisi PostGIS gibi konumsal analizlere imkan sağlayan eklentileri ile tez projesinde kullanılmak üzere tercih edilmiştir. PostgreSQL için onüçüncü versiyon, PostGIS için ise üçüncü versiyon kullanılmıştır.

Multipoligon geometri türünden poligon türüne dönüştürülen toplanma ve geçici barınma alanları verisinin bir PostgreSQL veritabanına aktarılması gerekmektedir. Bu

amaçlar PostGIS eklentisi ile beraber yüklenen “PostGIS Shapfile Import/Export Manager” isimli uygulama kullanılmıştır.




**Şekil 3.2 :** PostGIS Shapefile Import/Export Manager uygulaması.

Açılan arayüzde oluşturulan veritabanının bağlantı bilgileri tanımlandıktan sonra “Add file” butonu ile veritabanına aktarılmak istenilen shapefile dosyası seçilmiştir. Veride bulunan öznitelikler otomatik olarak ilgili tabloda oluşmakta ve tüm veriler seçilen tabloya aktarılmaktadır.

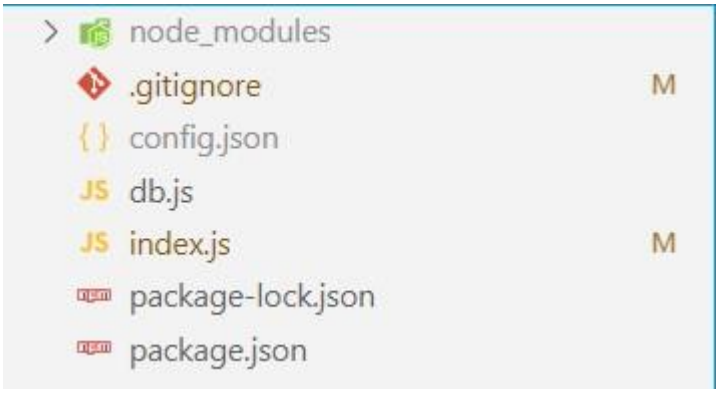

**Şekil 3.3 :** Katman özniteliklerinin veritabanına aktarılmış hali.

Proje kapsamında İstanbul ili için üretilmiş veride toplam 974 toplanma ve geçici barınma alanı bulunmaktadır.

### 3.3 Servis Katmanı

Yazılım projelerinde veritabanında saklanan verilerin direkt olarak mobil veya web uygulama üzerinden ulaşılması güvenlik açıklarına sebep olmaktadır. Veritabanı erişim bilgilerinin herkese açık olması, verilerin kaybolmasına veya manipüle edilmesine neden olabilmektedir. Bu nedenle verilerin herkese açık olarak sunulması için veritabanı ile kullanıcının kullandığı uygulama arasında bir servis katmanına ihtiyaç bulunmaktadır. Verilerin güvenli bir şekilde mobil uygulamaya aktarılması için proje kapmasında bir servis katmanı geliştirilmiştir.

Node.js ile geliştirilen proje konfigürasyonların tanımlandığı “package.json”, veritabanı bağlantısı ve veri sorgularının bulunduğu “db.js”, servise gelen isteklerin cevaplandığı “index.js”, önemli bilgilerin saklandığı “config.json” dosylarından oluşmaktadır.



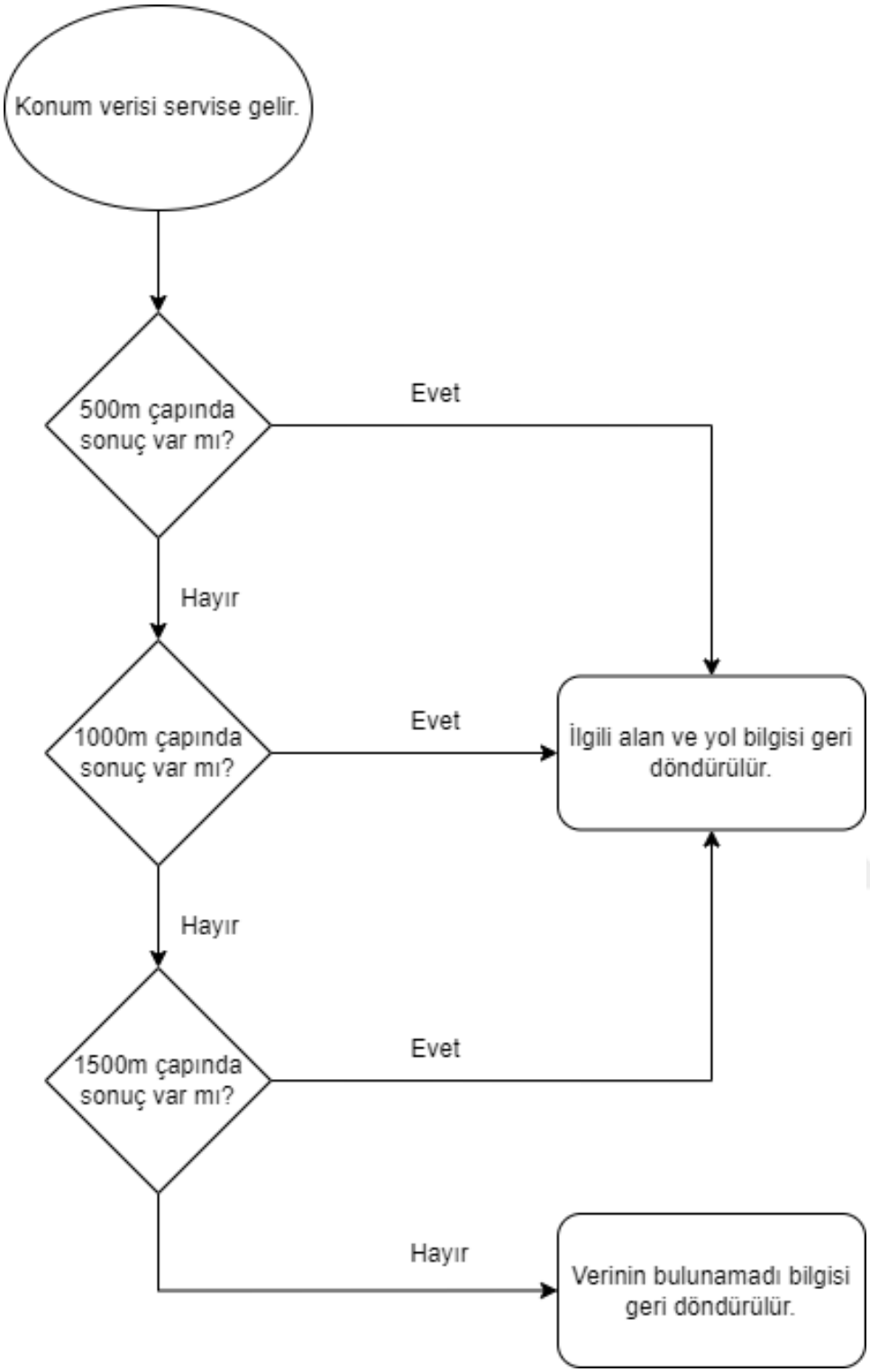

**Şekil 3.4 :** Servis katmanı dosya yapısı.

Veritabanı erişim bilgileri ve serviste kullanılacak ücretli uygulamaların APIKEY değerlerinin tek bir yerden yönetimi ve direkt olarak kod içinde kullanılmaması için "config.js" dosyası oluşturulmuştur. (Ek – A.4)

Veritabanı bağlantısı oluşturacak konfigürasyon bilgileri projeye dahil edildikten sonra "db.js" dosyasında bulunan ve PostgreSQL veritabanına bağlantı kurmayı sağlayan pg paketi ile veritabanı bağlantısı oluşturulur. (Ek – A.5)

Mobil uygulama üzerinden kullanıcının servise gönderdiği konuma karşılık etrafında bulunan toplanma ve geçici barınma alanlarına ait bilgiler, servisin çalıştığı sunucuya istek olarak gönderilir. Bu istek sonucunda çalışan kodlar, sırasıyla 500, 1000, 1500 metre çapında toplanma ve geçici barınma alanlarını veritabanından, konumsal SQL kod parçacıkları ile edinir. (Ek – A.6)

Çalıştırılan SQL kod parçacıkları sonucunda bir toplanma veya geçici barınma alanı bulunduğunda, kullanıcının servise gönderdiği konum ile veritabanının döndüğü konum arasındaki yürüyüş olarak en kısa zamana sahip yol Mapbox Directions API kullanılarak kullanıcya ilgili alanın ve yolun konum bilgisi geri döndürülür. (Ek – A.7)




**Şekil 3.5 :** Servis katmanının çalışma prensibi.

# 4. UYGULAMA

HAZTURK Son Depremler uygulaması artan akıllı telefonu kullanımı göz önünde bulundurularak mobil uygulama şeklinde geliştirilmesi amaçlanmıştır. Bu kapsamda öncelikle mobil uygulamanın geliştirileceği ortamın belirlenmesi gerekmektedir. No-code platformları hariç tutulduğunda mobil uygulamalar native (yerel) veya cross-platform (çapraz platform) araçlar ile ortaya çıkarılmaktadırlar. Native geliştirme araçları, Android için Kotlin ve Java, iOS içinse Swift ve Objective-C olarak tanımlanabilir. Native mobil uygulama geliştirme araçlarının aksine birçok farklı cross-platform uygulama geliştirme aracı bulunmaktadır. Cross-platform geliştirme araçları ile tek bir yazılım dili ile geliştirilmiş uygulamlar ilgili konfigürasyonlar yapıldığında birden fazla ortamda çalışabilmektedir. Örnek vermek gerekirse Android işletim sistemi için tasarlanıp, geliştirilen bir uygulama konfigürasyonlar yapıldıktan sonra iOS veya web ortamlarında da çalışabilmektedir. Farklı platformlar için tek bir kod yapısı ile uygulama geliştirilebildiği için bu araçların geliştiriciler arasındaki popülerliği gün geçtikçe artmaktadır. Statista tarafından yayınlanan ve otuz binden fazla yazılımcının katıldığı ankette 2021 yılında en çok tercih edilen üç cross-platform uygulama geliştirme aracı sırasıyla; Flutter, React Native ve Cordova olarak ortaya çıkmıştır (URL-13).

## 4.1 Mobil Uygulama

Mobil uygulama geliştirilirken tasarım, yazılım geliştirme ve yayına alma adımları uygulanmıştır. Uygulama geliştirme için cross-platform aracı olan Flutter seçilmiş ve Android işletim sistemi kullanım oranları göz önünde bulundurularak ana hedef olarak belirlenmiştir.

### 4.1.1 Tasarım

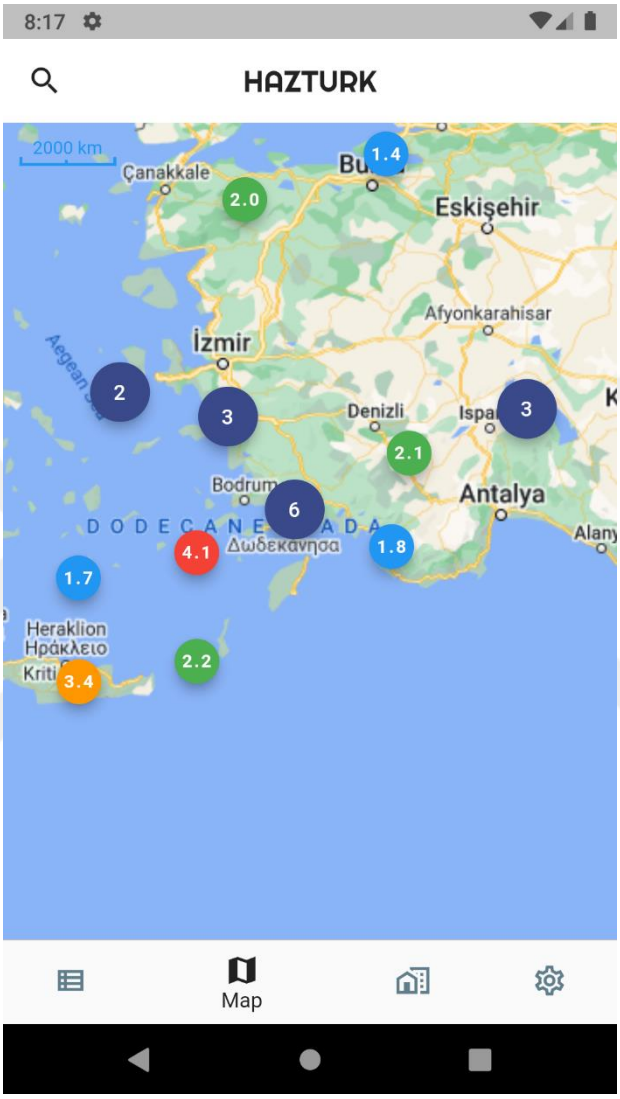
Mobil uygulama geliştirirmenin en önemli adımlarından biri olan tasarım adımı, kullanıcı arayüzü ve kullanıcı deneyimi olarak temel anlamda ikiye ayrılmaktadır. Kullanıcı arayüzünün kullanıcının ihtiyaçlarını sade bir şekilde ele alması, uygulamanın kullanıcı sayısını arttırmasına ek olarak verimli çalışmasına da olanak sağlamaktadır. (Atalar, 2020) Kullanıcı arayüzü, uygulamanın görünümüne

odaklanırken, kullanıcı deneyimi, kullanıcının uygulamada geçirdiği vakit süresince deneyimlediği olumlu ve olumsuz görüşlere odaklanmaktadır.

Google tarafından geliştirilen "Material Design" prensiplerine göre tasarlanan uygulamada kullanıcının fonksiyonlara en az efor ile ulaşması hedeflenmiştir. Akıllı telefon ekran boyutlarının giderek büyüdüğü dikkate alınarak ana menü ekranın altına konumlandırılmıştır. Uygulama ana menü üzerinden ulaşılana dört ana ekran olarak tasarlanmıştır. Meydana gelen depremlerin liste halinde kullancıya sunulduğu ekran, uygulama ilk açıldığında gösterilmektedir.

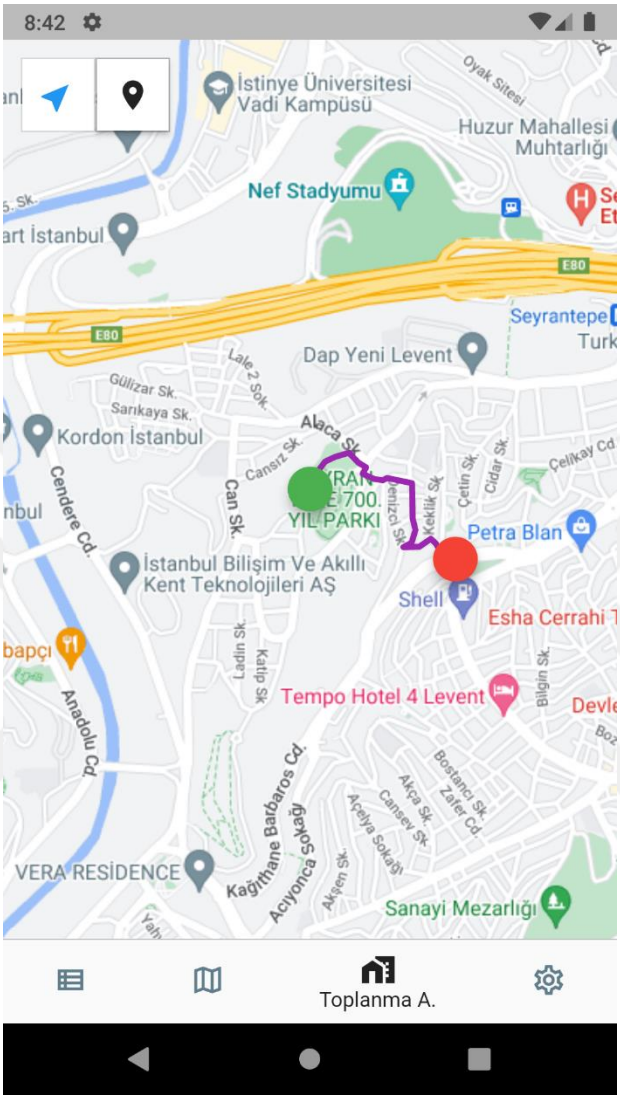

**Şekil 4.1 :** Meydana gelen depremler için liste görünümü.

Kullanıcının ana menüden ulaşabildiği ikinci ekran meydana gelen depremler için harita görünümünü içermektedir. Kümeleme algoritmasının kullandığı bu ekranda depremler, her bir küme için pasta grafiği veya deprem sayısı şeklinde gösterilmektedir.



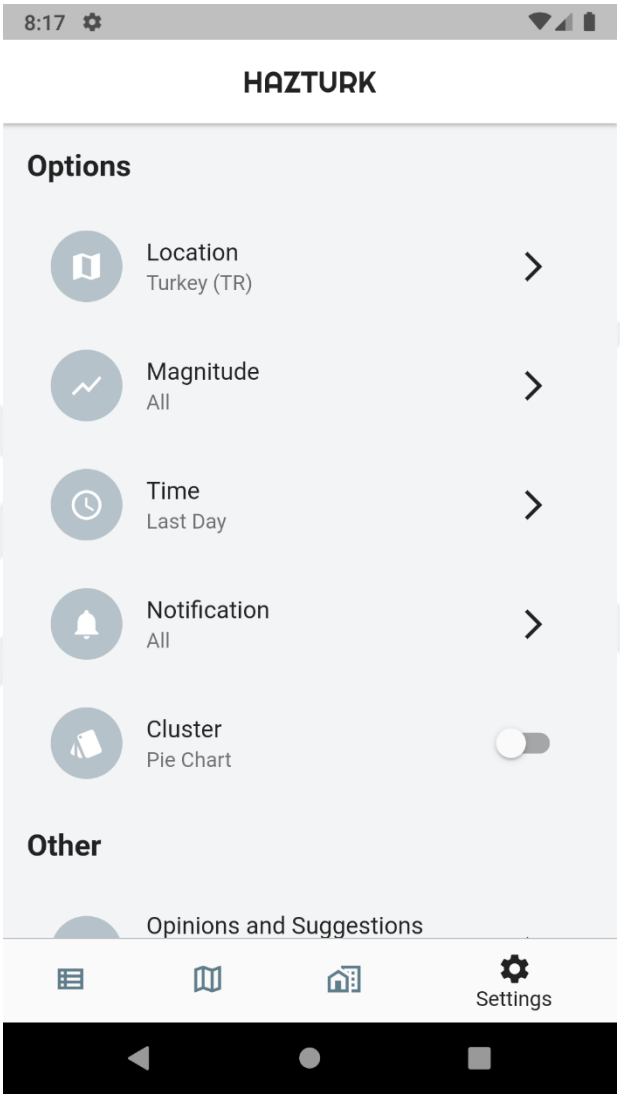

**Şekil 4.2 :** Harita görünümü.

Toplanma ve geçici barınma alanları harita üzerinde üçüncü ana ekranda kullanıcıya sunulmaktadır. Kulllanıcı tarafından toplanma ve geçici barınma alanları ile ilgili işlemlerin bu ekranda gerçekleştirmesi hedeflenmiştir.




**Şekil 4.3 :** Toplanma ve geçici barınma alanları ekranı.

Dördüncü ekran olarak uygulama ile ilgili ayarlar ekranı tasarlanmıştır. Kullanıcının almak istediği anlık bildirim filtresi, uygulamada sunulan deprem verisinin kaynağı, meydana gelen depremler için zaman ve büyüklük filtresi, harita görünümü için pasta grafiği seçeneği, son olarak da uygulamanın değerlendirilebileceği, paylaşılabileceği seçenekler sunulmuştur.



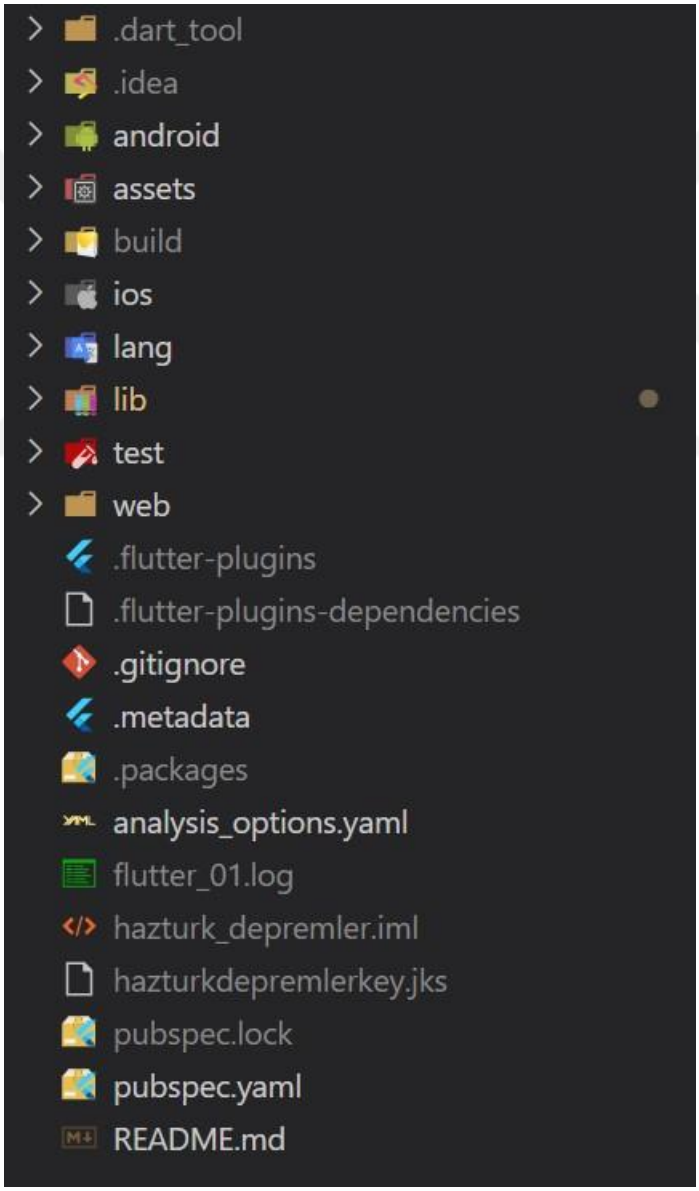

**Şekil 4.4 :** Ayarlar ekranı.

### 4.1.2 Uygulama yapısı

Mobil uygulama geliştirebilmek için öncelikle bir geliştirme aracı ve kodların yazılacağı bir IDE ihtiyacı bulunmaktadır. Mobil uygulamanın farklı platformlarda da çalışabilmesi için cross-platform geliştirme aracı olan Flutter tercih edilmiştir. Hafif, hızlı, geniş eklenti desteği ve açık kaynak bir proje olup, Microsoft tarafından geliştirilen Visual Studio Code yapılan çalışma kapsamında IDE olarak belirlenmiştir.

Flutter, Google tarafından 2017 yılında ilk kez sunulan, Android, iOS, masaüstü, web ve gömülü sistemler için uygulamalar geliştirilebilen bir araçtır. Tek bir kod yapısı ve ilgili konfigürasyonlar ile bu işletim sistemleri için hızlı bir şekilde uygulama geliştirmeyi hedeflemektedir. Bu nedenle öncelikli olarak Android, daha sonra iOS işletim sistemi için geliştirilen proje kapsamında Flutter geliştirme aracı olarak seçilmiştir.

Uygulama projesi temel klasör yapısına sahiptir. Şekil 2.5'te görüldüğü üzere Flutter ile geliştirilen uygulamalarda bulunması gerekli olan bu klasör ve dosya yapısı, projenin doğru bir şekilde derlenmesi için önem arz etmektedir.




**Şekil 4.5 :** Mobil proje klasör ve dosya yapısı.

Lib klasöründe uygulamanın işlevlerinin gerçekleştiği Dart yazılım dili ile yazılmış dart uzantılı dosyalar bulunmaktadır. Android klasöründe Android işletim sistemi, ios klasöründe iOS işletim sistemi ve web klasöründe web tabanlı uygulama için bulunması gerekli konfigürasyon dosyaları bulunmaktadır.

Tez kapsamında Android işletim sistemine yönelik uygulama çıktısı alınacağı için gerekli izinlerin tanımlanması ve konfigürasyonların yapılması gerekmektedir. Uygulamada internet üzerinden veri çekileceği için INTERNET, kullanıcının izin vermesi durumunda konum bilgisi kullanılacağı için ACCESS FINE LOCATION ve ACCESS COARSE LOCATION izinleri eklenmiştir (URL-14) (URL-15).

Uygulamada bulunması hedeflenen özellikler ve bu özelliklere göre ilgili Flutter paketleri belirlenmiştir. Kullanılacak paketler Flutter ve Dart için resmi paket merkezi olan “pub.dev” sitesi üzerinden bulunmuştur. Flutter için paket yönetim dosyası olan “pubspec.yaml” içinde bu paketler ve versiyonları tanımlanmıştır.

```
17    cupertino_icons: ^1.0.0
18    http: ^0.12.0+4
19    xml: ^5.0.2
20    shimmer: ^1.1.1
21    wc_flutter_share: ^0.2.0
22    flutter_map: ^0.10.1+1
23    flutter_map_marker_cluster: ^0.3.1
24    onesignal_flutter: ^3.2.7
25    lottie: ^0.7.0+1
26    customtogglebuttons: ^0.0.5
27    hive: ^1.4.4+1
28    hive_flutter: ^0.3.1
29    location: ^3.2.4
30    get_it: ^5.0.6
31    geodesy: ^0.3.2
32    pedantic: ^1.11.0
33    google_fonts: ^1.1.2
34    intl: ^0.17.0
35    pie_chart: ^5.0.0
36    auto_size_text: ^2.1.0
37    url_launcher: ^6.0.5
38
```

**Şekil 4.6 :** Mobil uygulamada kullanılan paketler.

Uygulama kapsamında farklı servis ve veri kaynaklarına istekler gönderildiği için http paketi entegre edilmiştir. Kandilli Rasathanesi tarafından sunulan deprem verileri XML formatında olduğu için xml paketi ile veriler işlenmektedir. Uygulamanın temel bileşenlerinden biri olan harita görünümü için flutter_map paketi tercih edilmiştir. Leaflet harita kütüphanesinin Dart diline entegre edilmiş versiyonu olan paket açık kaynak olması nedeniyle birçok farklı deneysel ve stabil özelliğe sahiptir. Hem Flutter hem de Leaflet paketler üzerine kurulmuş bir yapıya sahip olduklarında uyumlu bir şekilde çalışmaktadırlar. Harita üzerinde kümeleme ve pasta grafiği görünümleri de paketler vasıtası ile uygulamaya entegre edilmişlerdir. Kümeleme için flutter_map_marker_cluster, pasta grafiği içinse pie_chart paketi kullanılmıştır.

Uygulamanın önemli özelliklerinden biri olan anlık bildirimler, OneSignal ile sağlanmıştır. Anlık bildirimler OneSignal tarafından kullanıcı segmenti temelinde akıllı telefonlara gönderilmektedir. Altı farklı kullanıcı segmenti için sınırsız sayıda anlık bildirim hakkı tanıyan OneSignal tez kapsamında geliştirilen uygulama için yeterli olduğu görülmüştür (URL-16).

Herhangi bir kullanıcı giriş sistemi olmayan uygulamada, kullanıcılar depremler ile ilgili filtre, akıllı telefona gönderilecek anlık bildirimler gibi özellikler için kişisel tercihlerini hive paketi ile cihazlarında saklayabilmektedirler. OOP yapısını desteklemesi, diğer lokal veri saklama çözümlerine göre hızlı ve güvenli olması nedeniyle hive paketi tercih edilmiştir (URL-17).

Diğer önemli bir özellik olan kullanıcının anlık konumunu tespit edebilmek için get_it ve location paketleri beraber kullanılmıştır. Kullanıcının konumunun alınması süreci asenkron yapıda olduğu için direkt istek yapıldığında uygulamada çökmeler meydana gelebilmektedir. Ancak get_it paketi ile bu tarz istekler daha düzenli ve hızlı yapılabilmektedir.

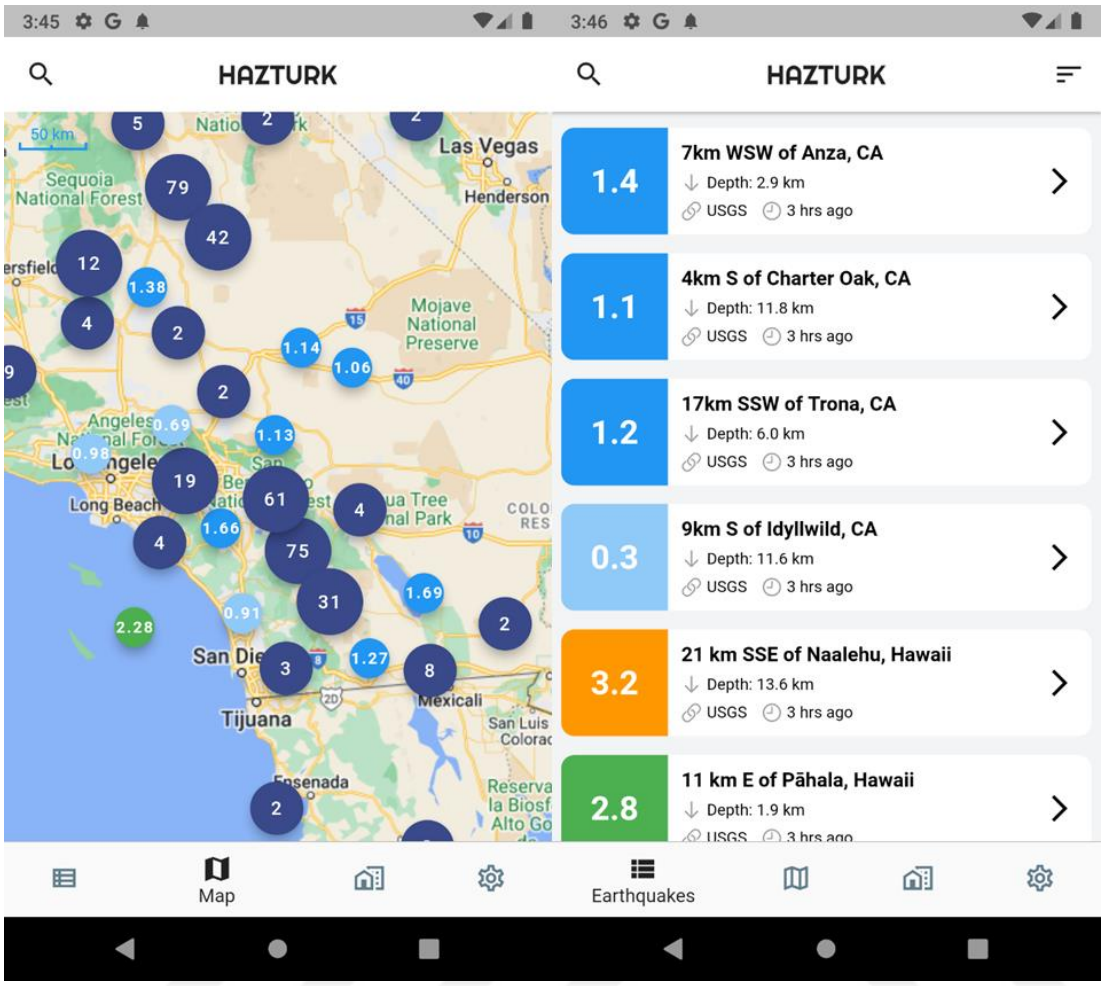
Geodesy, küresel bir dünya modeline göre jeodezik ve trigonometrik hesaplamalar yapmak için yazılmış bir dart paketidir. Tez kapsamında kullanıcı konum izni verdiyse, seçtiği deprem ile arasındaki kuş uçuşu mesafeyi hesaplamak için dahil edilmiştir.

### 4.1.3 Uygulama geliştirme

Tasarımın ve klasör yapısının oluşturulması ve uygulamada olması öngörülen özelliklere yönelik paketlerin seçilmesinin ardından, uygulamanın kodlanması kısmına geçilmiştir. Uygulamada olması planlanan tüm fonksiyonlar ve kullanıcının uygulamayı kullanırken karşı karşıya kalabileceği durumlara yönelik bir strateji ile kodların yazılması gerekmektedir. Projenin sürdürülebilirliğinin sağlanabilmesi için temiz kod felsefesine uygun, olabildiğince küçük parçalara ayrılmış ve kendini tekrar etmeyen kodlar yazılmaya özen gösterilmiştir.

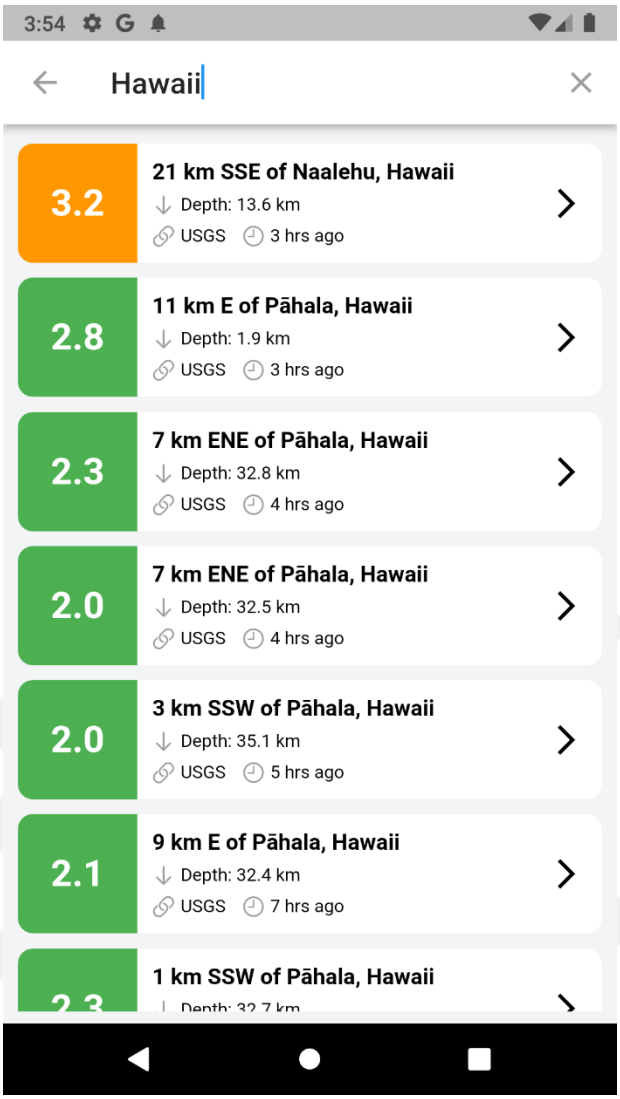
Native Android ve iOS geliştirme araçlarından farklı olarak Flutter, arayüz ve algoritma geliştirmeleri yalnızca dart uzantılı dosyalar üzerinden gerçekleştirilir. Kimi zaman tasarım ve geliştirilen uygulamanın farklı görünmesine neden olan bu yaklaşım, geliştirmede tüm hakimiyetin geliştiricide olmasına olanak tanır.

Uygulamanın temel özellikleri ile ilgili farklı çalışma prensipleri bulunmaktadır. Uygulamanın ana ekranı olanı depremlerin listelendiği ekranda geçerli bir internet bağlantısı bulunuyorsa ve seçilen deprem kaynağının ilgili servisi isteklere cevap veriyorsa, sonuçlar liste olarak gösterilmektedir. Eğer geçerli bir internet bağlantısı yoksa veya deprem veri kaynağı olan servis gönderilen isteklere cevap vermiyorsa hata oluşur ve ilgili hata ekranda gösterilir. Harita ekranı için aynı çalışma prensibi işlemektedir. Farklı olarak, sadece başarılı istekler için geri dönen deprem verisi için harita üzerinde gösterim yapılmaktadır. Dünya genelinde meydana gelen depremlerin uygulamada gösterimi için U.S. Geological Survey tarafından sunulan GeoJSON türündeki veriler iki adet fonksiyon ile hem liste görünümü hem de harita görünümü için uygulamada kullanılmaktadır. Kullanıcının liste görünümünde seçtiği sıralama türlerine göre sıralı bir şekilde dönüştürülen veri, OOP yapısına uygun bir şekilde kullanıldığı için kod tekrarı ve karmaşık bir yapıyı ortadan kaldırmaktadır. (Ek – A.8)



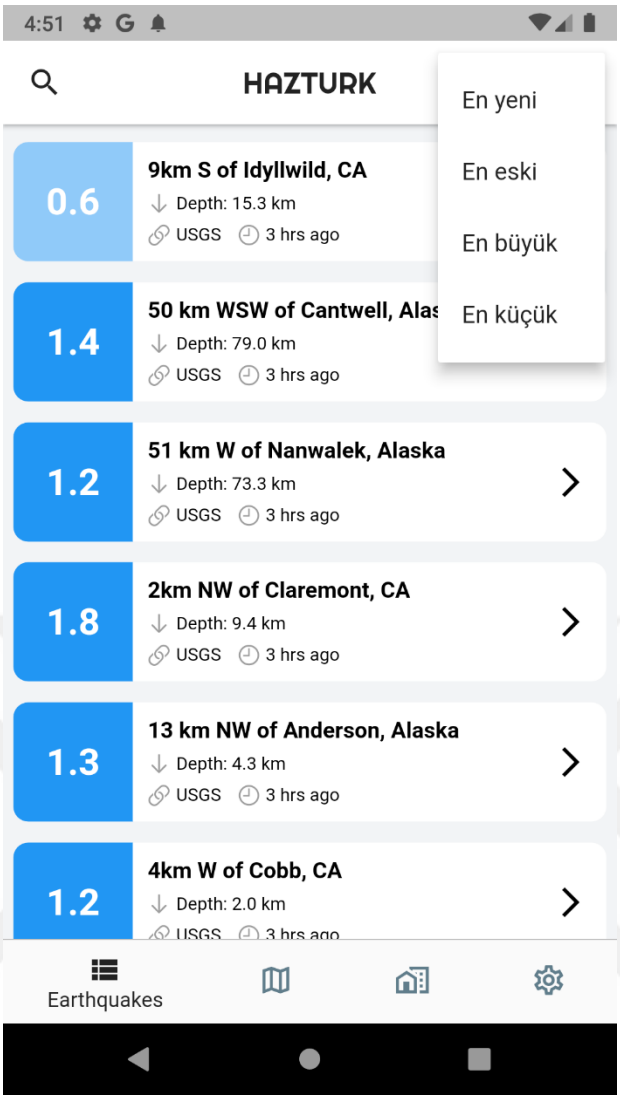

**Şekil 4.7 :** U.S.G.S tarafından servis edilen veriler.

Deprem listesi ve harita görünümünde ekranın üst-sol kısmında bulunan ve büyüteç simgesi ile tasvir edilen buton ile meydana gelmiş deprem verisi içinde arama yapılabilmektedir. Şekil 2.8'de görüldüğü üzere "Hawaii" anahtar kelimesi ilgili alana yazıldığında yalnızca deprem başlığında "Hawaii" içeren depremler liste halinde gösterilmektedir.



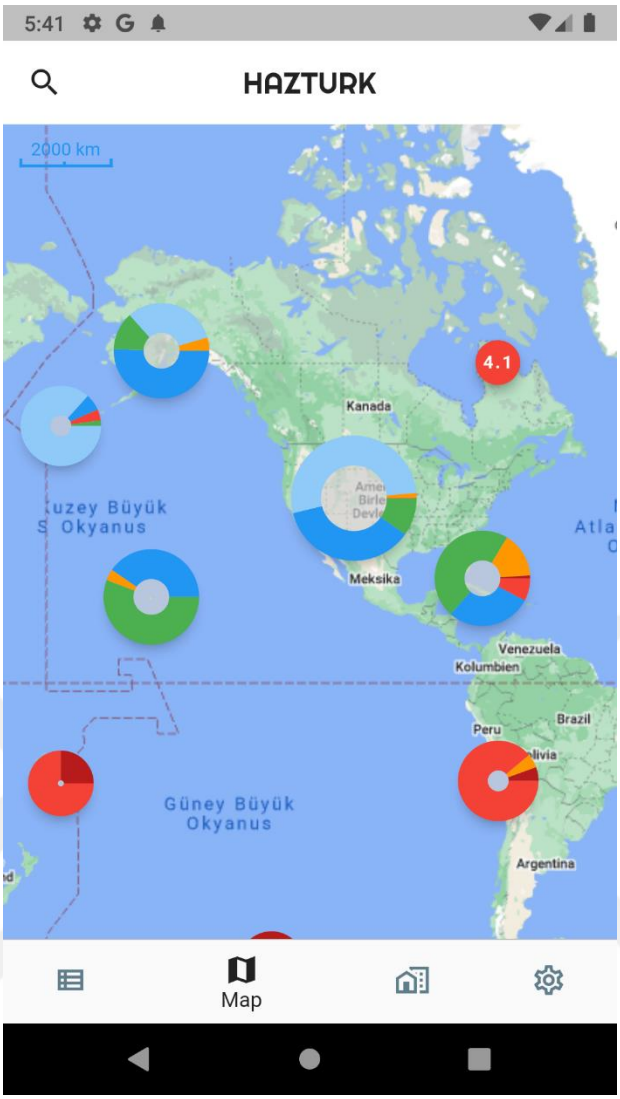

**Şekil 4.8 :** Örnek arama ile kısıtlanmış deprem verisi.

Meydana gelmiş deprem liste ekranının sağ üst kısmında bulunan buton ile depremler arası sıralama yapılabilmektedir. Zamana ve büyüklüğe göre sıralanabilen depremler için bulunan menüde, şekil 2.9'da görüldüğü üzere, "En yeni", "En eski", "En büyük" ve "En küçük" seçenekleri bulunmaktadır. Varsayılan değer olarak "En yeni" seçeneği tanımlanmıştır. Ek – A.8 içinde bulunan "parseUSGS" fonksiyonunun aldığı parametrelerden olan "isReverse" zamana göre, isBuyuk ise meydana gelmiş depremleri büyüklüklerine göre sıralamaya imkan vermektedir.




**Şekil 4.9 :** Deprem sıralama ölçütleri.

Harita görünümü ekranında temel anlamda harita üzerinde ilgili meydana gelmiş deprem verisi kullanıcıya gösterilmektedir. Liste görünümünde olduğu gibi üst menünün sol kısmında depremler arasında arama yapılabilmektedir. Özellikle dünya çapında meydana gelmiş depremlerin uzun tarih aralığı için her birinin tekil olarak harita üzerinde gösterilmesi ciddi performans kayıplarına neden olmaktadır. Bu performans kaybının önüne geçebilmek için, meydana gelmiş depremler cluster (kümeleme) görünümü ile kullanıcıya sunulmaktadır. Bu anlamda kümelemenin yapıldığı depremler için, ilgili kümeye kullanıcı tıkladığında bu depremlere yakınlaşmış olacaktır. Ayarlar ekranında bahsedilecek olan pie chart (pasta grafiği) görünümü ile depremler gösterilebilmektedir. Kullanıcı ayarlardan bu seçeneği aktif ettiğinde harita görünümündeki kümelenmiş depremler büyüklüklerine göre renklendirilmiş pasta grafiği şeklinde gösterilmektedirler.




**Şekil 4.10 :** Kümelenmiş depremler için pasta grafiği görünümü.

Deprem listesi, harita ve tekil deprem görünümünde görselleştirme yapılırken meydana gelen depremin büyüklüğü göz önünde bulundurulmaktadır. Renklendirme çizelge 2.1'de görüldüğü üzere yapılmıştır. Kısa bir fonksiyon ile tanımlanan bu renklendirme temiz kod anlayışı içinde tekrarların önüne geçmiştir. (Ek – A.9)

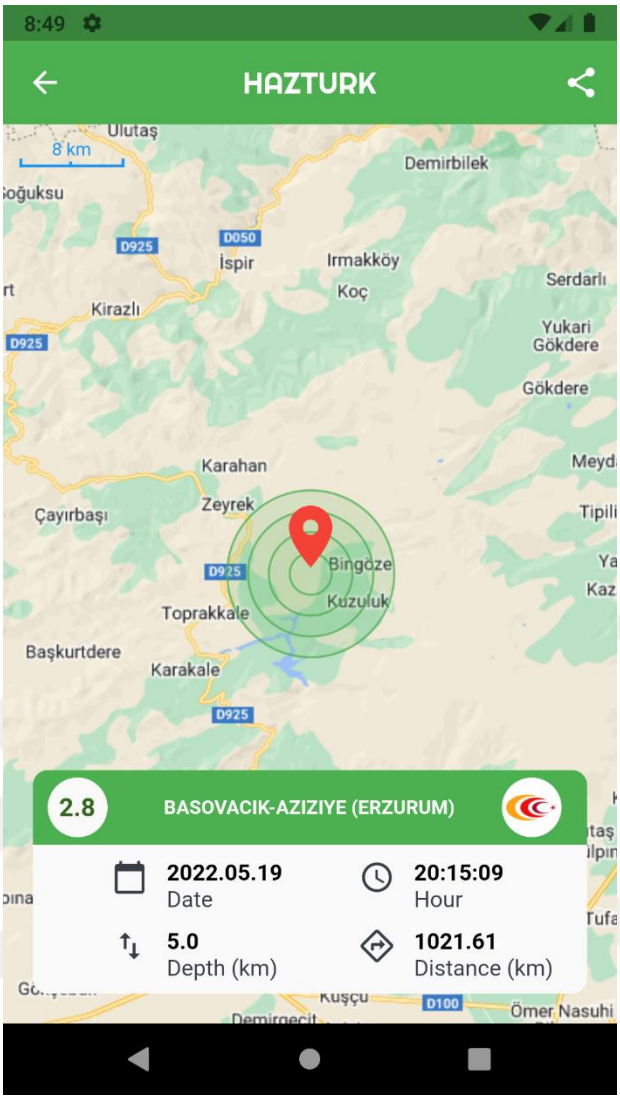
| Deprem Büyüklük Aralığı | Renk Kodu |
|---|---|
| Büyüklük < 1 | #90CAF9 |
| 1 <= Büyüklük < 2 | #2196F3 |
| 2 <= Büyüklük < 3 | #4CAF50 |
| 3 <= Büyüklük < 4 | #FF5722 |
| 4 <= Büyüklük < 5 | #F44336 |
| 5 < Büyüklük | #B71C1C |

**Çizelge 4.1 :** Deprem büyüklük aralıklarına göre renkler.

Toplanma alanları ekranında, harita üzerinde bulunan toplanma ve geçici barınma alanı verisi ile kullanıcıya sunulmaktadır. Toplanma ve geçici barınma alanları verisi tez kapsamında İstanbul ili özelinde sınırlandırılmıştır. Veri bir PostgreSQL veritabanında saklanmaktadır ve Node.JS geliştirme ortamında Javascript yazılım dili ile geliştirilmiş bir restful servis ile uygulamadan gönderilen isteklere karşılık olarak bulunmaktadır.

Toplanma alanları ekranında bulunan ve sol üstte yer alan butonlardam birincisi, kullanıcı konum izni verdiyse kullanıcıya en yakın toplanma veya geçici barınma alanını ve Mapbox Directions API ile oluşturulan en hızlı yürüyerek gidelecek yolu harita üzerinde göstermektedir. (Ek – A.10) İkinci buton seçili ise harita bileşeni üzerinde basılı tutulan nokta için en yakın toplanma veya geçici barınma alanı ve bu iki nokta arasındaki yol haritada gösterilmektedir. (Ek – A.11) İlgili konum analizleri servis kısmında tamamlandıktan sonra, toplanma veya geçici barınma alanı ve servisten iletilen iki nokta arası yol offline bir şekilde kullanıcının akıllı telefonuna kaydedilebilecektir. (Ek – A.12)

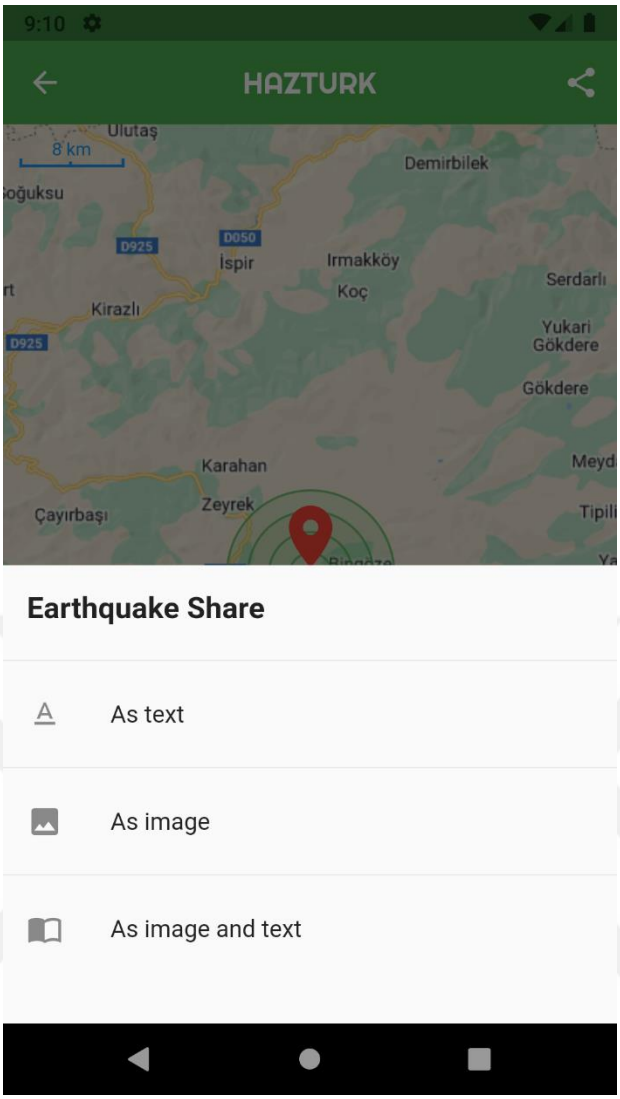
Deprem listesi ekranı, deprem harita ekranı üzerinden tekil bir deprem seçildiğinde veya akıllı telefona gelen deprem bildirimine tıklanıldığında deprem detay sayfası kullanıcıya sunulmaktadır.




**Şekil 4.11 :** Deprem detay ekranı.

Harita bileşeni üzerinde kırmızı pin ile işaretlenen depremin merkez üssünün konumunda, depremin gerçekleşme büyüklüğünü tasvir etmek amacı ile daireler kullanıcıya gösterilmektedir. Ekranın ve deprem detaylarının bulunduğu kart görünümünün üst kısmı Ek – A.9'de tanımlanan fonksiyondan geri dönen renk ile renklendirilmektedir. Ayrıca deprem servisinden gelen konumun adı, depremin büyüklüğü, depremin olduğu saat ve tarih, depremin kilometre cinsinden derinliği ve son olarak kullanıcı uygulamaya konum izni verdiyse kullanıcı ile depremin merkez üssü arasındaki kuş uçuşu uzaklık geodesy paketi ile kilometre cinsinden hesaplanıp kullanıcıya sunulmaktadır.

Seçilen deprem, ekranın sol üst kısmında bulunan paylaş butonu ile farklı platformlarda farklı şekillerde paylaşılabilmektedir. Kullanıcı paylaş butonuna tıkladığında üç seçeneğe sahip bir menü açılmaktadır.



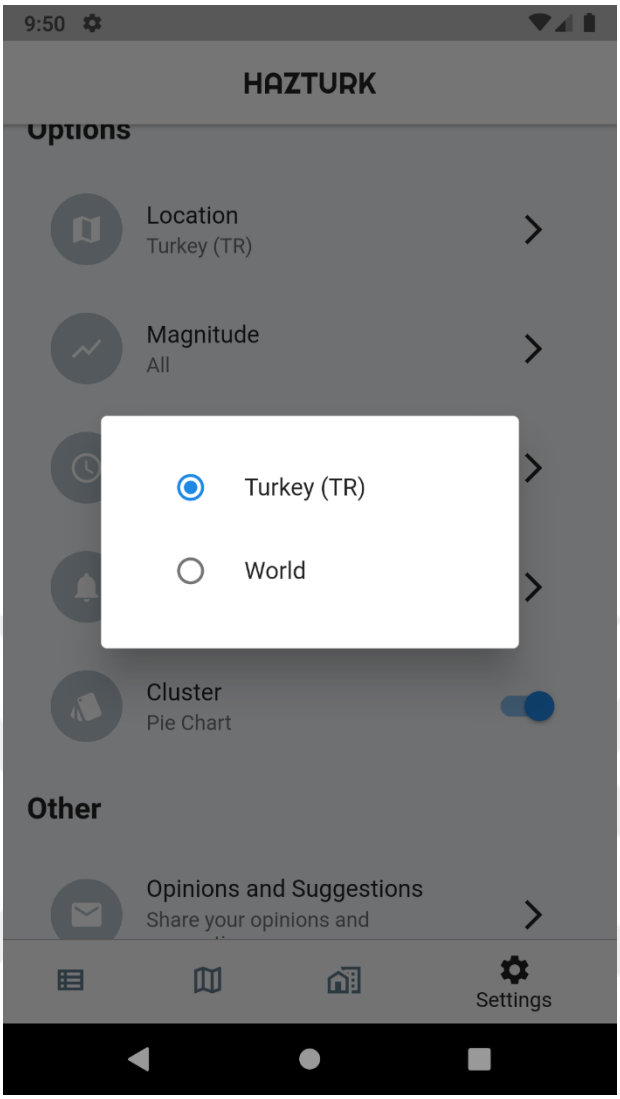

**Şekil 4.12 :** Deprem paylaş seçenekleri

Açılan menüde ilk seçenek olan yazı olarak paylaş ile deprem detayları yalnızca yazı olarak farklı platformlarda paylaşılabilmektedir. İkinci seçenek olan resim olarak paylaş seçeneği ile ekranın hali hazırdaki durumunun ekran görüntüsü alınıp paylaşıma açılabilmektedir. Son olarak hem yazı hem de resim olarak paylaşım yapılabilen seçenek ile deprem detayları yazı ve ekran görüntüsü olarak paylaşılabilmektedir.

Yazı olarak paylaş seçeneklerinde bulunan taslak, depremin büyüklüğünü, konum adını, meydana gelme zamanını, yeryüzünden derinliğini ve enlem, boylam bilgilerini içermektedir.

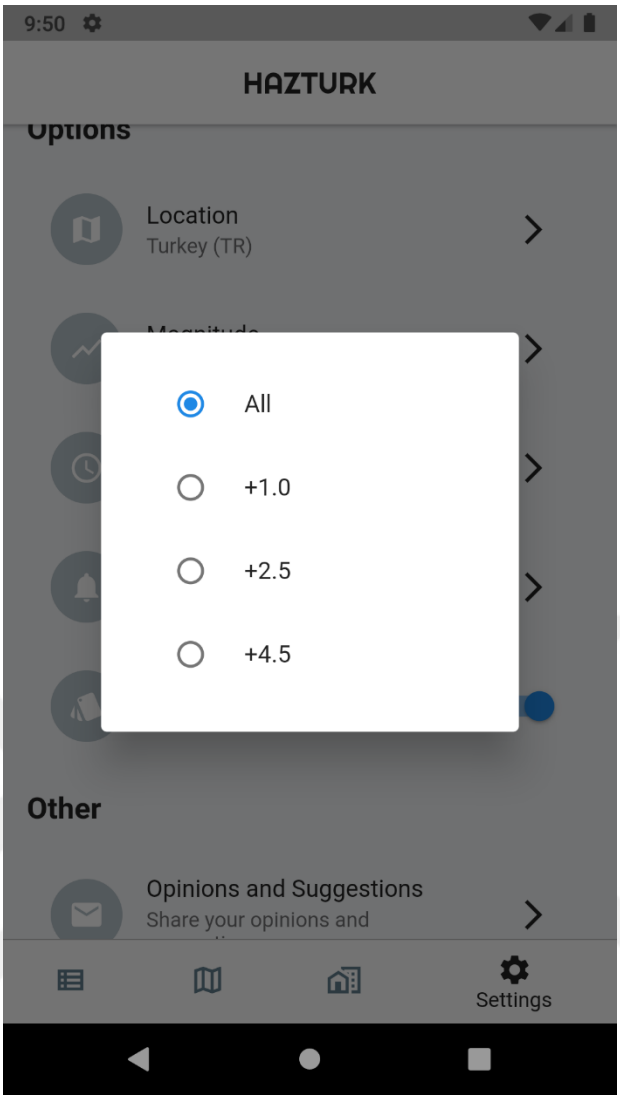

**Şekil 4.13 :** Sosyal medyada paylaşılmış bir deprem.

Uygulamada bulunan birçok özellik için özelleştirmeler yapmaya imkan sağlayan ayarlar ekranı seçenekler ve diğer olmak üzere iki ana bölümden oluşmaktadır. Seçenekler bölümünde kullanıcının tercihlerine yönelik ayarlar bulunmaktadır. Seçili ayarlar kullanıcının uygulama deneyimini arttırmak amacı ile her bir seçenek satırında gösterilmektedir. Konum seçeneği seçildiğinde depremlerin sunulduğu servis seçilmektedir. Türkiye seçili ise Kandilli Rasathanesi, Dünya seçili ise USGS tarafından sunulan veriler uygulamada sunulmaktadır.



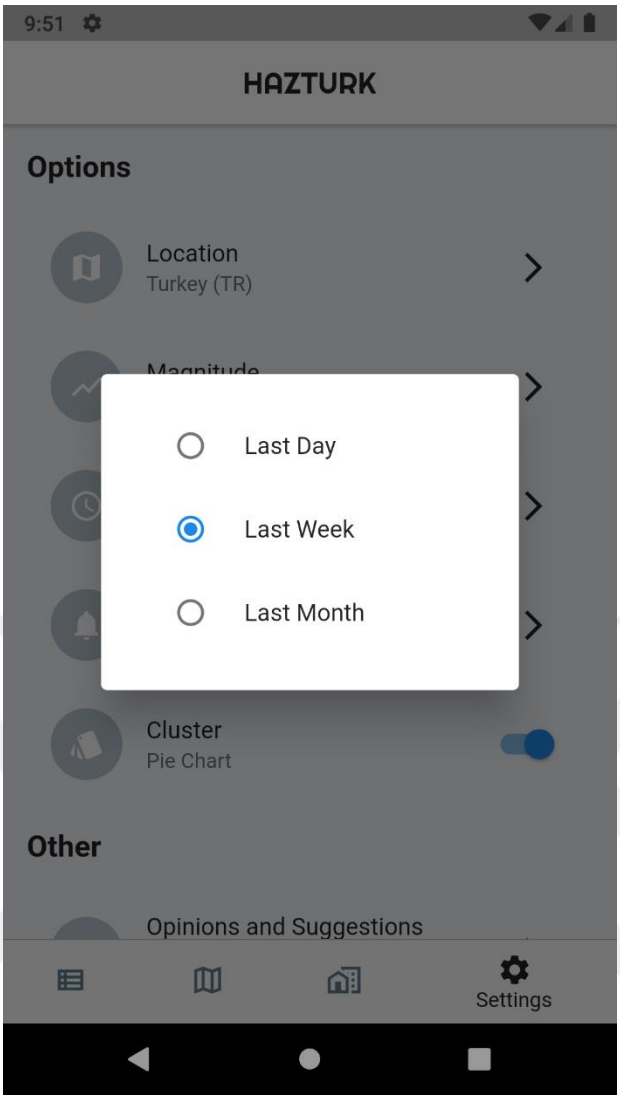

**Şekil 4.14 :** Konum seçeneği.

Büyüklük seçeneği ile liste ve harita görünümlerinde gösterilecek depremlerin en az büyüklük değeri belirlenmekte olup "Tümü", "+1.0", "+2.5", "+4.5" seçenekleri bulunmaktadır.




**Şekil 4.15 :** Büyüklük seçeneği.

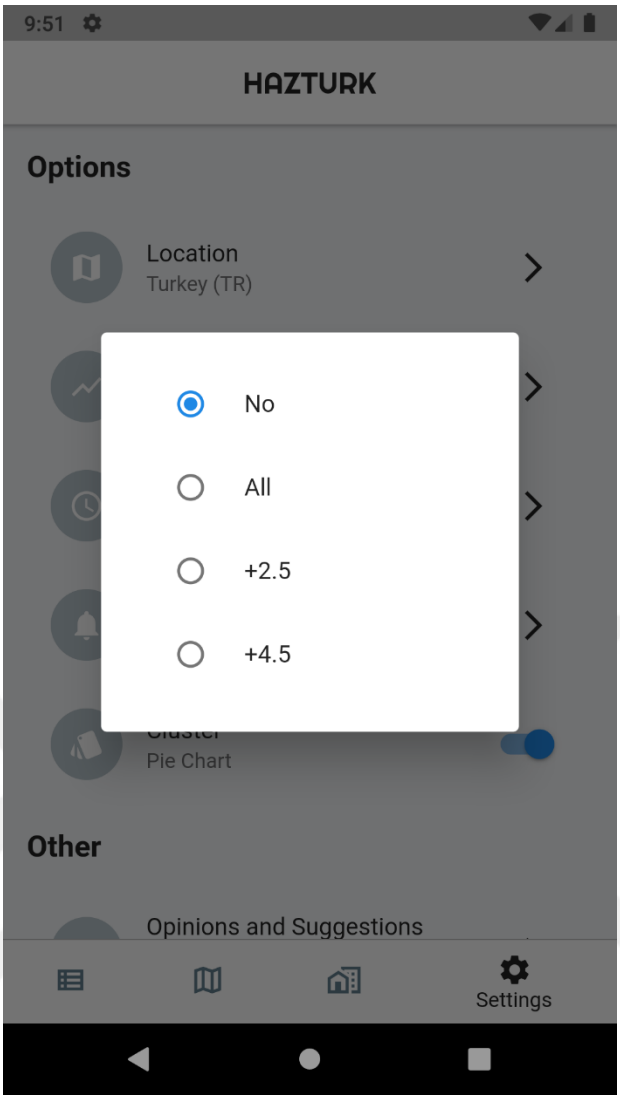
Uygulamada bulunan deprem verisi zaman seçeneği ile son yirmi dört saat, son bir hafta ve son bir ay şeklinde filtrelenebilmektedir.




**Şekil 4.16 :** Zaman seçeneği.

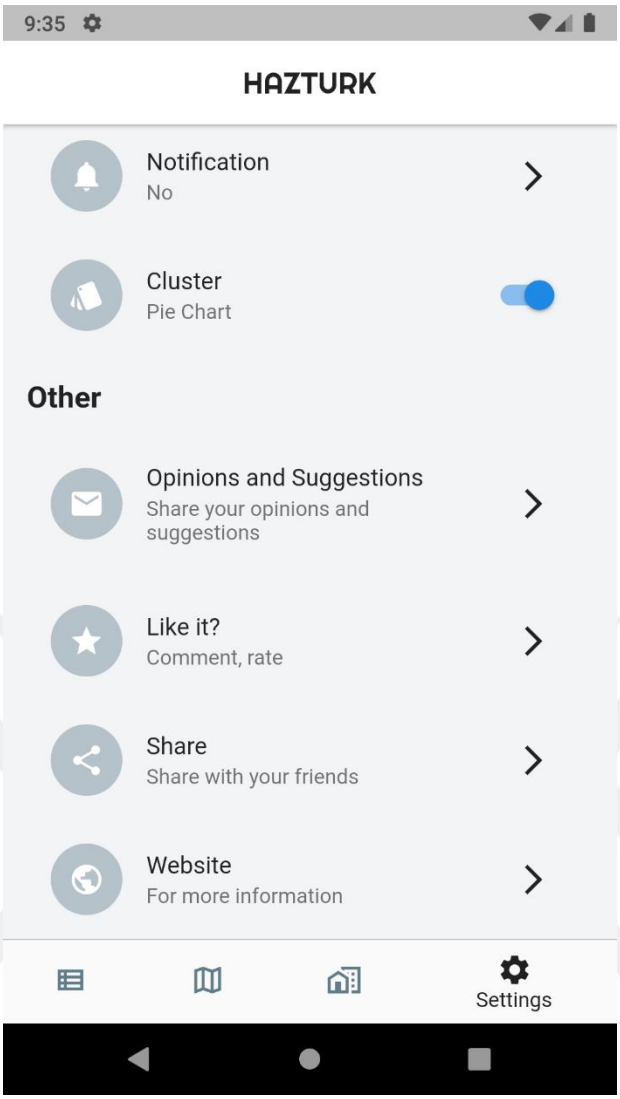
Uygulamada bulunmakta olan anlık bildirim özelliği için de bir seçenek ayarlar ekranında bulunmaktadır. Hali hazırda yalnızca Türkiye'de meydana gelen depremler için bulunan anlık bildirim özelliği OneSignal ile sağlanmaktadır.

Varsayılan olarak Türkiye'de meydana gelen ve Kandilli Rasathanesi tarafından sunulan depremlerin tümü için anlık bildirim alma üzerine kurulan uygulamada kullanıcı, sağlanan seçeneklere göre almak istediği bildirimleri filtreleyebilmektedir. Şekil 2.17'de görüldüğü üzere "Yok", "Hepsi", "+2.5" ve "+4.5" seçenekleri kullanıcı tarafından seçilebilmektedir. Yok seçeneği seçildiğinde depremler hakkında hiçbir bildirim alınmamakta, hepsi seçeneği ile tüm deprem bildirimleri, +2.5 ve +4.5 seçenekleri ile de meydana gelen depremin büyüklüğüne göre anlık bildirimler akıllı telefonlara gönderilmektedir.




**Şekil 4.17 :** Anlık bildirim seçeneği.

Son olarak ayarlar ekranının seçenekler bölümünde bulunan kümeleme seçeneği ile meydana gelen depremlerin harita üzerinde pasta grafiği olarak gösterilip gösterilmemesi kullanıcı tarafından seçilmektedir.




**Şekil 4.18 :** Ayarlar ekranı diğer bölümü.

Şekil 2.18'de görüldüğü üzere ayarlar ekranının diğer bölümü dört farklı satırdan oluşmaktadır. "Görüş ve öneri" satırı seçildiğinde varsayılan e-posta uygulaması konu ve alıcı bilgiler dolu bir şekilde açılıp kullanıcıdan görüş ve önerilerinin bildirilmesi beklenmektedir. "Beğendiniz mi?" satırı seçildiğinde ilgili işletim sisteminin mağazasında uygulama sayfası açılarak kullanıcının yorum ve puanlama yapması beklenmektedir. "Paylaş" satırı ile uygulamanın mağaza linki farklı platformlar üzerinden paylaşıma açılırken, "Web Sitesi" satırı ile HAZTURK web sitesinin anasayfası açılmaktadır.

# 5. SONUÇ VE ÖNERİLER

Günümüzde gelişen teknoloji sayesinde, her geçen gün akıllı telefon ve internet kullanım oranları artmaktadır. Bu duruma paralel olarak internet erişimine sahip mobil uygulama kullanımı, akıllı telefon kullanıcıları arasında yaygınlaşmaktadır. Tez kapsamında geliştirilen uygulama ile Android işletim sistemine sahip akıllı telefonlara yönelik deprem bilgi mobil uygulaması geliştirilmiştir. Uygulamada farklı iki kaynaktan servis edilen, Türkiye ve dünyada meydana gelen depremler kullanıcıya ulaştırılmaktadır. Kullanıcının meydana gelen depremler ile ilgili ayrıntılı bilgiye ulaşabildiği uygulamaya konum izni verdiği takdirde, meydana gelen deprem ile arasındaki kuş uçuşu mesafeye uygulama üzerinden ulaşabilmektedir. Meydana gelmiş depremleri sunan servislerden gelen veriler uygulama liste ve harita üzerinde gösterilmektedir. Liste görünümünde Google'ın yayınlamış olduğu Material Design standartlarına uygun bir tasarım geliştirilmiştir. Özellikle bir aylık verilerin harita üzerinde ciddi bir yük oluşturmasından dolayı uygulamada performans düşüşleri gözlenmiştir. Bu performans kaybından dolayı veriler harita üzerinde kümeleme yöntemi kullanılarak kullanıcıya sunulmuştur.

Meydana gelen depremleri liste ve harita görünümü olarak kullanıcıya sunan uygulamada yalnızca Türkiye'de gerçekleşen depremler için anlık bildirimler gönderilmektedir. Kullanıcı almak istediği anlık bildirimleri, meydana gelen depremin büyüklüğüne göre filtreleyebilmektedir. Gerçekleşen depremlerin anlık bildirim sistemi ile kullanıcıya ulaştırılması için devamlı olarak kontrol edilmesi gerekmektedir. Bu kontrol insan gücü ile sürdürülebilir şekilde yönetilemeyeceği için belirli sabit bir aralıkta deprem verisini sağlayan servis kontrol edilmektedir. Javascript yazılım dili ile hazırlanan proje docker ortamında çalışmaktadır.

Mobil uygulama için İstanbul ili özelinde üretilmiş toplanma ve geçici barınma alanları verisi, bir PostgreSQL veritabanında saklanıp, Node.js projesi ile mobil uygulamaya sunulmaktadır. Mobil uygulamadan veritabanına direkt erişim güvenlik açıklarına neden olacağı için bir Node.js projesi geliştirilmiştir. Veritabanı erişim bilgilerini işleyerek veritabanında PostGIS eklentisi ile konumsal analizler yapılarak kullanıcının sağladığı konum verisine en yakın toplanma veya geçici barınma alanı ve yol bilgisi uygulamaya döndürülmektedir. Bu sayede kullanıcılar için olası bir deprem senoryasında nasıl hareket etmeleri konusunda da bilinç sahibi olmaktadırlar.

Mobil uygulamanın sonraki versiyonlarında dünyada meydana gelen depremler için de anlık bildirimler gönderilmesi uygulamanın kapsamını arttıracaktır. Ayrıca toplanma ve geçici barınma alanları hakkındaki servis ilgili resmi kurumlar tarafından paylaşılırsa Türkiye genelinde bulunan kullanıcıların tamamı için toplanma ve geçici barınma alanlarının konumları hakkında bir bilinç oluşacaktır.

## KAYNAKLAR


**Atalar, D.** (2020). Arayüz Tasarımında Kullanıcı Deneyimi Amaçlı Prototip Tasarımı (Lisansüstü tezi). Selçuk Üniversitesi, Sosyal Bilimler Enstitüsü, Konya.

**Özkılıç, E. N.** (2020). İstanbul'da deprem sonrası toplanma alanlarının kapasitelerinin ve erişilebilirliklerinin cbs yardımıyla analizi ve değerlendirilmesi. (Lisansüstü tezi). İstanbul Teknik Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.

**Url-1**<*http://www.koeri.boun.edu.tr/sismo/2/deprem-bilgileri/buyuk-depremler/*>, erişim tarihi 23.05.2022.

**Url-2**<*https://play.google.com/store/apps/details?id=org.emsc_csem.lastquake/*>, erişim tarihi 15.05.2022.

**Url-3** <*https://play.google.com/store/apps/details?id=dbs.uygulama/*>, erişim tarihi 15.05.2022.

**Url-4**<*https://play.google.com/store/apps/details?id=com.basarsoft.afaddeprem/*>, erişim tarihi 15.05.2022.

**Url-5**<*https://play.google.com/store/apps/details?id=tr.gov.icisleri.afad*>, erişim tarihi 15.05.2022.

**Url-6** <*https://play.google.com/store/apps/details?id=com.jrustonapps*

*.myearthquakealerts*>, erişim tarihi 15.05.2022.

**Url-7**<*https://play.google.com/store/apps/details?id=disasterAlert.PDC/*>, erişim tarihi 15.05.2022.

**Url-8**<*https://play.google.com/store/apps/details?id=edu.berkeley.bsl.myshake/*>, erişim tarihi 15.05.2022.

**Url-9** <*https://play.google.com/store/apps/details?id=com.joshclemm*

*.android.quake/*>, erişim tarihi 15.05.2022.

**Url-10**<*https://play.google.com/store/apps/details?id=com.beylikduzu.abis/*>, erişim tarihi 15.05.2022.

**Url-11** <*https://bulutistan.com/blog/docker-nedir/*>, erişim tarihi 15.05.2022.

**Url-12**<*https://dev.socrata.com/docs/datatypes/multipolygon.html#*>, erişim tarihi 20.05.2022.

**Url-13**<*https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/)*>, erişim tarihi 20.05.2022.

**Url-14**<*https://developer.android.com/training/location/permissions/*>, erişim tarihi 20.05.2022.

**Url-15**<*https://developer.android.com/training/basics/network-ops/connecting/*>, erişim tarihi 20.05.2022.

**Url-16**<*https://onesignal.com/pricing*>, erişim tarihi 20.05.2022.

**Url-17**<*https://objectbox.io/flutter-databases-sqflite-hive-objectbox-and-moor/*>, erişim tarihi 20.05.2022.

## EKLER

**EK A.1 :** Docker konfigürasyon dosyası.

```
FROM node:latest
WORKDIR /app
COPY . .
RUN npm i
CMD ["node", "index.js"]
```

**EK A.2 :** Node.js proje dosyası.

```
{
  "name": "earthquake-nodejs",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "type": "module",
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cheerio": "^1.0.0-rc.10",
    "node-fetch": "^3.0.0"
  }
}
```

**EK A.3 :** Anlık bildirim isteği gönderen metod.

```
let sendNotification = (earthquake) => {
  let includedSegments = ["allmag"];
  if (earthquake.mag >= 2.5) {
    includedSegments.push("twofivemag");
  }
  if (earthquake.mag >= 4.5) {
    includedSegments.push("fourfivemag");
  }
  fetch("https://onesignal.com/api/v1/notifications", {
    method: "POST",
    headers: {
      Authorization: "Basic " + configJSON.OS_REST_KEY,
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      app_id: configJSON.OS_APP_ID,
      included_segments: includedSegments,
      contents: { en: `${earthquake.mag}-${earthquake.location}` },
      headings: { en: "DEPREM" },
      data: {
        date: earthquake.date,
        location: earthquake.location,
        lat: earthquake.lat,
        lng: earthquake.lng,
        mag: earthquake.mag,
        depth: earthquake.depth,
      },
    }),
  })
    .then((response) => console.log("Sent", response.text))
    .catch((error) => console.log("Sending error", error));
  writeEQToFile(`${earthquake.mag}-${earthquake.location}`);
};
```

**EK A.4 :** Konfigürasyon bilgileri.

```
{
    "USERNAME":"",
    "PASSWORD":"",
    "DATABASE":"",
    "HOST":"",
    "PORT":"",
    "MAPBOXAPI":"",
    "TABLE":"",
    "LOCALHOST": ""
}
```

**EK A.5 :** Veritabanı bağlantısını gerçekleştiren kod parçacığı.

```
const pool = new Pool({
  user: config.USERNAME,
  password: config.PASSWORD,
  host: config.HOST,
  database: config.DATABASE,
  port: parseInt(config.PORT),
});
```

**EK A.6 :** Kullanıcının servise ilettiği konum ve buffer çapına göre toplanma ve geçici barınma alanlarını döndüren fonksiyon.

```
const distanceTA = (request, response) => {
  const lng = parseFloat(request.params.lng);
  const lat = parseFloat(request.params.lat);
  let distance = parseInt(request.params.distance);
  const sql =
    "select ta.gid, ta.layer, ta.kod,
st_x(st_transform(st_centroid(ta.geom),4326)),
st_y(st_transform(st_centroid(ta.geom),4326)) from " +
    config.TABLE +
    " ta where
st_contains(ST_Buffer(st_transform(ST_GeomFromText('POINT(" +
    lng +
    " " +
    lat +
    ")', 4326), 5254)," +
    distance +
    ",'quad_segs=8'),ta.geom);";
  pool.query(sql, [], (error, results) => {
    if (error) {
      throw error;
    }
    response.status(200).json(results.rows);
  });
};
```

**EK A.7 :** Kullanıcıya toplanma ve geçici barınma alanının ve yol bilgisini döndüren fonksiyon.

```
{
  var cikti;
  const lng = req.params.lng;
  const lat = req.params.lat;
  const datalist = [];
  let url = `http://${config.LOCALHOST}:9000/ta/${lng};${lat};500`;
  let fetchdata = await fetch(url);
  let data = await fetchdata.json();
  console.log(data);
  if (data.length > 0) {
    for (var i = 0; i < data.length; i++) {
      datalist.push(
        `https://api.mapbox.com/directions/v5/mapbox/walking/${lng},${l
at};${data[i].st_x},${data[i].st_y}?geometries=geojson&access_token=${c
onfig.MAPBOXAPI}`
      );
    }
  } else {
    url = `http://${config.LOCALHOST}:9000/ta/${lng};${lat};1000`;
    fetchdata = await fetch(url);
    data = await fetchdata.json();
    console.log(data);
    if (data.length > 0) {
      for (var i = 0; i < data.length; i++) {
        datalist.push(
          `https://api.mapbox.com/directions/v5/mapbox/walking/${lng},$
{lat};${data[i].st_x},${data[i].st_y}?geometries=geojson&access_token=$
{config.MAPBOXAPI}`
        );
      }
    } else {
      url = `http://${config.LOCALHOST}:9000/ta/${lng};${lat};1500`;
      fetchdata = await fetch(url);
      data = await fetchdata.json();
      console.log(data);
      if (data.length > 0) {
        for (var i = 0; i < data.length; i++) {
          datalist.push(
            `https://api.mapbox.com/directions/v5/mapbox/walking/${lng}
,${lat};${data[i].st_x},${data[i].st_y}?geometries=geojson&access_token
=${config.MAPBOXAPI}`
          );
        }
      }
    }
  }
  if (datalist.length > 0) {
```

```
    var durations = [];
    const gelenler = await getParallel(datalist);
    for (var k = 0; k < gelenler.length; k++) {
      if (gelenler[k].routes.length > 0) {
        durations.push(gelenler[k].routes[0].duration);
      }
    }
    var minIndex = durations.indexOf(Math.min.apply(Math, durations));
    cikti = gelenler[minIndex];
  } else {
    cikti = { data: "no data" };
  }
  res.status(200).json(cikti);
}
```

**EK A.8 :** GeoJSON verinin işlenmesi için yazılmış fonksiyon.

```
List<UsgsEQ> parseUSGS(List argumentList) {
  final responseBody = argumentList.elementAt(0);
  final isReverse = argumentList.elementAt(1);
  final isBuyuk = argumentList.elementAt(2);
  final parsed = jsonDecode(responseBody)['features'];
  var values = parsed
      .map<UsgsEQ>((e) => UsgsEQ(
            DateTime.fromMillisecondsSinceEpoch(
                    int.parse(e['properties']['time'].toString()))
                .toString(),
            e['properties']['place'],
            double.parse(e['geometry']['coordinates'].elementAt(1).toSt
ring()),
            double.parse(e['geometry']['coordinates'].first.toString())
,
            double.parse(e['properties']['mag'] != null
                ? e['properties']['mag'].toString()
                : '0'),
            double.parse(e['geometry']['coordinates'].last.toString()),
          ))
      .toList();
  if (isReverse) {
    values = values.reversed.toList();
  }
  if (isBuyuk is String) {
    if (isBuyuk == 'buyuk') {
      values.sort((UsgsEQ a, UsgsEQ b) => b.mag.compareTo(a.mag));
    } else if (isBuyuk == 'kucuk') {
      values.sort((UsgsEQ a, UsgsEQ b) => a.mag.compareTo(b.mag));
    }
  }
  return values;
}
```

**EK A.9 :** Deprem büyüklüğüne göre renk geri dönen fonksiyon.

```
Color earthquakeColor(String value) {
  var value2Double = double.parse(value);
  var defaultColor = Colors.blue[200];
  if (value2Double >= 5) {
    defaultColor = Colors.red[900];
  } else if (value2Double >= 4) {
    defaultColor = Colors.red;
  } else if (value2Double >= 3) {
    defaultColor = Colors.orange;
  } else if (value2Double >= 2) {
    defaultColor = Colors.green;
  } else if (value2Double >= 1) {
    defaultColor = Colors.blue;
  }
  return defaultColor;
}
```

**EK A.10 :** Kullanıcının konumuna en yakın toplanma veya geçici barınma alanını ve aradaki yolu haritaya ekleyen metod.

```
void _getCurrentLocation() async {
    var locationServices = getIt<LocationServices>();
    await locationServices.getPosition().then((position) async {
      mapController.move(LatLng(position.latitude, position.longitude),
15);      var response = await http.get(
          'http://${url}:9000/checkta/${position.longitude};${position.
latitude}');
      if (response.statusCode == HttpStatus.ok) {
        final jsonBody = jsonDecode(response.body);
        await Future.delayed(Duration(seconds: 1));
        var waypoints = jsonBody['waypoints'];
        var routes = jsonBody['routes'];
        markerList = [];
        polylineList = [];
        if (waypoints is List && waypoints.isNotEmpty) {
          markerList.add(Marker(
              point: LatLng(waypoints.first['location'].last,
                  waypoints.first['location'].first),
              builder: (context) => FloatingActionButton(
                    heroTag: null,
                    onPressed: null,
                    backgroundColor: Colors.red,
                  )
              ));
          markerList.add(Marker(
              point: LatLng(waypoints.last['location'].last,
                  waypoints.last['location'].first),
              builder: (context) => FloatingActionButton(
                    heroTag: null,
                    onPressed: null,
                    backgroundColor: Colors.green,
                  )
              ));
          if (routes is List && routes.isNotEmpty) {
            routes[0]['geometry']['coordinates'].forEach((pnt) {
              polylineList.add(LatLng(pnt.last, pnt.first));
            });
          }
          mapController.move(
              LatLng(waypoints.last['location'].last,
                  waypoints.last['location'].first),
              15);
          setState(() {});
        }
      }
    });
  }
```

**EK A.11 :** Tıklanan nokta için en yakın toplanma veya geçici barınma alanını ve aradaki yolu haritaya ekleyen metod.

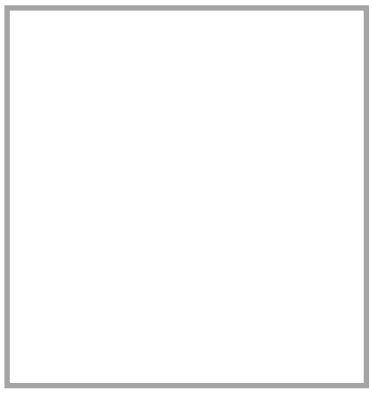
```
void _handleLongPress(LatLng latlng) async {
    var response = await http.get(
        'http://172.23.208.1:9000/checkta/${latlng.longitude};${latlng.
latitude}');
    if (response.statusCode == HttpStatus.ok) {
      final jsonBody = jsonDecode(response.body);
      await Future.delayed(Duration(seconds: 1));
      var waypoints = jsonBody['waypoints'];
      var routes = jsonBody['routes'];
      markerList = [];
      polylineList = [];
      if (waypoints is List && waypoints.isNotEmpty) {
        markerList.add(Marker(
            point: LatLng(waypoints.first['location'].last,
                waypoints.first['location'].first),
            builder: (context) => FloatingActionButton(
                  heroTag: null,
                  onPressed: null,
                  backgroundColor: Colors.red,
                )
            ));
        markerList.add(Marker(
            point: LatLng(waypoints.last['location'].last,
                waypoints.last['location'].first),
            builder: (context) => FloatingActionButton(
                  heroTag: null,
                  onPressed: null,
                  backgroundColor: Colors.green,
                )
            ));
        if (routes is List && routes.isNotEmpty) {
          routes[0]['geometry']['coordinates'].forEach((pnt) {
            polylineList.add(LatLng(pnt.last, pnt.first));
          });
        }
        mapController.move(
            LatLng(waypoints.last['location'].last,
                waypoints.last['location'].first),
            15);
        setState(() {});
      }
    }
  }
```

**EK A.12 :** Toplanma veya geçici barınma alanını kaydeden metod.

```
() {
  var userLocation =
      Provider.of<TaStateData>(context,
              listen: false)
          .markerList[0];
  var polylineList =
      Provider.of<TaStateData>(context,
              listen: false)
          .polylineList;
  var list = [];
  polylineList.forEach((x) =>
      list.add([x.latitude, x.longitude]));
  setPaths(
      '${userLocation.point.latitude},${userLocation.point.longitude}',
      list);
  Provider.of<TaStateData>(context,
          listen: false)
      .changeVisibilityAll(false);
}
```

## ÖZGEÇMİŞ

**Ad-Soyad** **:** Berkay Oruç



**ÖĞRENİM DURUMU:**

- **Lisans** **:** 2019, İstanbul Teknik Üniversitesi, İnşaat Fakültesi, Geomatik Mühendisliği

**MESLEKİ DENEYİM VE ÖDÜLLER:**

- 2019 – 2021 yılları arasında Odakent Çevre & Bilişim firmasında masaüstü ve web cbs uygulama geliştiricisi olarak görev aldı.
- 2021 yılından beri İstanbul Büyükşehir Belediyesi Coğrafi Bilgi Sistemi Müdürlüğü Yazılım Şefliğinde web cbs uygulama geliştiricisi olarak görev almaktadır.