İZMİR KÂTİP ÇELEBİ ÜNİVERSİTESİ
2010
FEN BİLİMLERİ ENSTİTÜSÜ

LocTag (Konum etiketleme ve not alma uygulaması)

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans

Meriç Gerçeker

ORCID 0000-0002-1137-605X

Danışman: Dr. Öğr. Üyesi Osman Gökalp

Ocak 2023

İzmir Kâtip Çelebi Üniversitesi Fen Bilimleri Enstitüsü öğrencisi **Meriç Gerçeker** tarafından hazırlanan **LocTag (Konum etiketleme ve not alma uygulaması)** başlıklı bu çalışma tarafımızca okunmuş olup, yapılan değerlendirme sonucunda kapsam ve nitelik açısından başarılı bulunarak TEZSİZ YÜKSEK LİSANS olarak kabul edilmiştir.

ONAYLAYANLAR:

Tez Danışmanı: **Dr. Öğr. Üyesi Osman Gökalp**
İzmir Kâtip Çelebi Üniversitesi

Yazarlık Beyanı

Ben, **Meriç Gerçeker**, başlığı **LocTag (Konum etiketleme ve not alma uygulaması)** olan bu tezimin ve tezin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu tezin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Tezin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Tezde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 06.01.2023

_____

LocTag (Konum etiketleme ve not alma uygulaması)

Öz

Tez çalışması kapsamında tasarlanan ve geliştirilen mobil uygulama ile, haritadan anlık konum işaretlemesi yapılabilmektedir. Bu işaretlemelere eklenme imkanı olan yorum özelliği sayesinde, daha önce işaretlenmiş konumlar ayırt edilebilmektedir. Kayıtlı konum bilgileri, liste halinde görüntülenebildiği gibi, harita üzerinden de doğrudan görüntülenebilmektedir. Konumlamanın çalışma prensibi gereği, işaretlemenin sadece yorum metni düzenlenebilmektedir. Kullanıcılar dilerlerse, işaretledikleri konumu silebilmektedirler.

Anahtar Sözcükler: Mühendislik, teknoloji, mobil uygulama, yazılım

LocTag (Location tagging and note taking app)

Abstract

With the mobile application designed and developed within the scope of the thesis study, instant location marking can be made from the map. Thanks to the comment feature, which can be added to these markings, previously marked locations can be distinguished. The recorded location information can be displayed as a list or directly on the map. Due to the working principle of positioning, only the comment text of the marking can be edited. Users can delete the location they marked, if they wish.

Keywords: Engineering, technology, mobile application, software

Teşekkür

Tez çalışmasına katkılarından dolayı danışman hocam Dr. Öğr. Üyesi Osman Gökalp'e, Enstitü Anabilim Dalı Başkanı Doç. Dr. Aytuğ Onan'a, iş arkadaşlarıma ve aileme teşekkürlerimi sunuyorum.

İçindekiler

Şekiller Listesi

Bölüm 1

Giriş

Mobil cihazlarımız bizlerin hayatlarını kolaylaştırıyor ve yapmamız gereken işlerimizi kısa sürelerde tamamlamamızı sağlıyor. Bir ödeme yapmak istediğimizde yada bir araştırma yapmamız gerektiğinde, saniyeler içinde bunları gerçekleştirebiliyoruz. Tabi ki bu işlemleri gerçekleştirmek için aracı araçlar kullanıyoruz. Bu araçlara mobil uygulama adını veriyoruz. Mobil uygulamalar, kullanıcıların, sistemler ile bir mobil cihaz kullanarak haberleşmesini sağlıyor. Mobil cihazlar bir telefon olabileceği gibi, bir akıllı saat yada tablet bilgisayar da olabilmekte.

Mobil uygulamaların bu kadar revaçta olması nedeniyle; yazılım alanında ilerleme kaydetmek isteyenler, mobil uygulama geliştiriciliği alanına yöneliyorlar. Şu an 2 farklı mobil işletim sistemi mevcut. Google tarafından piyasaya sunulmuş olan Android ve Apple tarafından piyasaya sunulmuş olan iOS. Windows Phone ve BlackBerryOS adında 2 farklı işletim sistemi daha mevcuttu. Fakat zamanla popülerliği azaldı ve kullanım oranı çok düşük seviyelere geldi.

En popüler 2 mobil işletim sisteminin yazımında, farklı 2 dil ve bazı durumlarda farklı 2 işletim sistemine sahip bilgisayar kullanılıyor. Android için Windows yada Mac bir bilgisayar ile derleme yapma imkanına sahipken, iOS derleme yapabilmek için bir Mac cihaza sahip olmanız gerekiyor. Yine Android için uygulama geliştirken Java veya Kotlin dillerini kullanabilmekteyiz. iOS için ise, Swift yada Objective-C kullanmamız gerekmekte.

Mobil uygulama ve geliştirme süreçlerinin bu şekilde popüler hale gelmesi ile birlikte, cross-platform framework'ler geliştirilmeye başlandı. İlk olarak Windows

tarafından Xamarin framework'u piyasaya sunuldu. Ardından Facebook tarafından React Native ve en son Google tarafından Flutter. Cross-platform framework'ler sayesinde, tek kod yazdıktan sonra; hem Android, hem de iOS için çıktı üretme imkanına sahip olunabiliyor. Bu sayede iş gücü azalmış ve çıkan çıktılar yakın sonuçlar vermiş oluyor.

Android ve iOS arasında değişiklik yapan veya her ikisini de kullanan kullanıcılar için, bu ortak yapı büyük kolaylık sağlıyor. Neredeyse birebir aynı çıktılar nedeniyle, kolayca adapte olabilme fırsatı yakalıyorlar.

Bölüm 2

Mobil Uygulama Geliştirme Süreçleri

2.1 Kullanılan programlama teknolojileri

Bu tezin kapsamında geliştirilen mobil uygulamada, öncelikle hangi dili veya framekwork'u kullanamam gerektiğine karar vermeliydim. Native Android yazmak için elimde 2 farklı dil seçeneği var. Birisi köklü bir dil olan Java, diğeri ise 2011 yılında piyasaya çıkmış Kotlin.

Kotlin, alternatif dile göre daha güncel ve kullanım oranı daha fazla. En son sürümü 2022 yılının Ağustos ayında yayınlandı ve güncellenmeye devam ediyor. Ayrıca Google I/O 2017'de de, resmi Android programlama dili olarak duyurulmuştu. Yazım olarak iOS Swift'e benzerliğiyle dikkat çekmekte.

Native Android ile uygulamamızı geliştirdiğimizi varsayarsak, iOS cihazlarda kullanım imkanı sunmamış olacağız. Bu imkanı sağlamak için, Native iOS programlamayı da gerçekleştirmem gerekiyor. Native iOS için de yine 2 farklı dil seçeneğim var. Birisi 1983 yılında piyasaya çıkmış olan Objective-C. Diğeri ise 2014 yılında piyasaya çıkmış olan Swift. Bu tercihlerde de Swift oldukça önde.

Swift, güçlü yapısı, kolay ve anlaşılır kodlama imkanı sunması ile ön plana çıkıyor. Piyasaya çıktığı tarihten itibaren, çoğu araştırma sonuçlarında ilk sıralarda yer almış ve kullanım oranı gün geçtikçe artmaya devam ediyor.

Şimdi tekrardan başa dönersek, uygulamamızı 2 farklı işletim sisteminde de kullanmak istiyoruz. Bunun için Kotlin ile Native Android, Swift ile Native iOS yazmak istediğimizi varsayalım. Uygulamadaki ekranlarımızı Android için ayrı, iOS için ayrı tasarlamamız gerekecek. Bazı özelliklerin, birinde olup birinde olmadığını

gözlemleyeceğiz. Bir kütüphane kullanmak isteyeceğiz. Android'de farklı, iOS'ta farklı sonuçlar verme ihtimali olacak. Şu haliyle bir API bağlantımız yok. Fakat olduğu zamanlarda, yine 2 farklı tarzda kodlama nedeniyle, endpoint'lerde ayrıştırma yapmamız gerekebilecek.

Şansımızın yaver gittiğini ve tüm kütüphanelerimizin uyumlu, senkron çalıştığını düşünelim. Tasarımsal olarak da hemen hemen aynı görüntüleri elde ettik. Fakat sonuca baktığımızda, 2 farklı kodlama yapmış olacağız. Aynı işlemleri, 2 farklı dilde yazacağız. Uygulama için harcadığımız iş gücümüz 2 katına çıkmış olacak. Haliyle sonuca ulaşma süremiz de uzamış olacak.

Mobil programlama teknolojisi çok hızlı gelişmekte. Web ortamlarına nazaran, her geçen gün farklı önlemler almak gerekebiliyor. Android için örnek vermek gerekirse; Android 8'e uyumlu yazdığınız uygulamanız, Android 11 ile çalışmayabiliyor. Yeni versiyonlar çıktıkça, izinlerin kullanım şekilleri değişmiş olabiliyor. Bu durumlarda, uygulamanızda güncelleme yapmanız gerekecek. Diyelim ki; artık Android'in belirli bir versiyonlarında, arka planda konum alabilmek için, farklı şekilde izin almanız gerekiyor. Bunu da doğrudan yapamıyorsunuz ve kullanıcıyı ayarlar ekranına yönlendirmeniz gerekiyor. Bu duruma çözüm bulmak için şöyle bir yol izlemiş olabiliriz. Kullanıcıya bir uyarı ekranı hazırladık. Bu ekran, eğer arka planda konum toplama iznini daha önce yanıtlamamışsa çıkacak. Arka planda neden konum toplama iznine ihtiyacımız olduğunu açıkladığımız bir metin var. Bu metnin altında da izni yanıtla gibi bir butonumuz var. Bu butona bastığında, ayarlar ekranına yönlendiriyoruz. Bu yönlendirilen ekrandan, kullanıcı izni veriyor ya da reddediyor. Biz bu şekilde güncellememizi tamamladık ve markete göndermeye hazırız. Fakat bu olay sadece Android için geçerli olacak. iOS'ta herhangi bir değişiklik yok. iOS'taki versiyonumuz geride kalmış oldu. Uygulamanın business katmanında bir değişiklik yaptık. Çünkü izin vermezse, farklı şekilde bir süreç işleyecek. Bunu iOS için yapamadık. Bu durum şu an için bir problem gibi gözükmeyebilir. Ancak bu özel kodlamalar arttıkça, karmaşık bir hal almaya başlayacaktır.

Tüm bu sorunların önüne geçmek için, cross-platform framework'ler hayatımıza girdi. Artık farklı işletim sistemleri için, tek bir kod yazma ve bunun çıktılarını yaygınlaştırma imkanına sahibiz. Ekranları birebir aynı şekilde tasarlıyoruzi (Tabi ki istersek özelleştirme şansımız da var) 2 farklı kod yazmadığımız için, ilerleyen

süreçteki bakım işlemleri de, oldukça rahat ve güvenli oluyor. Ben uygulamamı tek başıma geliştirdiğim için, cross-platform framework kullanmaya karar verdim.

Cross-platform ile yazmaya karar verdikten sonra, şimdi de hangi framework'ü kullanmam gerektiğine karar vermeliyim. Xamarin öncülerinden biri. Arkasında Microsoft desteği var. C# ile geliştirilmesi ve benim Full Stack geliştirici hayatımda çoğunlukla C# tercih etmem nedeniyle, biraz ön plana çıkıyor. Fakat Xamarin özellikle ülkemizde çok fazla yaygın kullanılan bir framework değil. Özellikle topluluk desteği bakımından zorlanabileceğimi düşünüyorum. Ayrıca bazı kütüphaneler için desteklenmeme ihtimaliyle de karşılaşabilirim.

Bir diğer seçeneğim React Native. Facebook tarafından, 2015 yılında piyasaya çıktı. Güçlü kütüphane ve topluluk desteği ile ön plana çıkıyor. Programlama dilinin Javascript olması nedeniyle, kolay bir yazımı var. Özellikle React kullanan yazılımcılar, React Native yazma konusunda zorluk yaşamıyorlar.

Son seçeneğim de Flutter. Google tarafından 2017 yılında piyasa çıktı. Özellikle zengin ui widget desteği, her geçen gün artan kütüphane ve topluluk desteği ile son yılların en popüler framework'lerinden biri oldu. Ülkemizde de bir çok şirket, uygulamalarını Flutter'a dönüştürmeye ya da yeni uygulamalarını Flutter ile geliştirmeye devam ediyor. Hot Reload özelliği ile, yazılan kodun anlık derlenmesi ve ekranda gözlemlenebilmesi, kodlama hızını oldukça arttırıyor. Yine Google tarafından 2011 yılında piyasaya çıkan Dart programlama dili kullanılıyor. Nesne tabanlı bir dil olması sebebiyle, C# geliştiriciler kolayca adapte olabiliyor. Öğrenme sürecini hızlı bir şekilde tamamlayabiliyorlar. Flutter'ın widget yazımlarında, iç içe yazılmış bir yapı kullanılıyor. JSON'ı andıran bu yapı, kolay anlaşılırlık sağlıyor. Bu gibi bir çok avantajı nedeniyle ben uygulamayı Flutter ile geliştirme kararı aldım.

İçerisinde harita barındıran mobil uygulamalarda, platform'a göre farklılıklar yaşanabilmekte. Örneğin; Android işletim sistemi için Google haritaları baz alınırken, iOS işletim sistemi için Apple haritaları tercih edilebilmekte. Hatta Huawei'nin almış olduğu kararlar nedeniyle, bazı modellerinde Android işletim sistemi kullanılmasına rağmen Google servislerine erişilememekte. Bu durumda Google Maps'in kullanımını da gerçekleştirememiş olmaktayız. Flutter ile tek tip kod yazdığımız için, bu farklılığı en aza indirme fırsatını yakalayabiliyoruz.

2.2 Kullanılan kütüphaneler

Uygulamamızın temel işlevi olan Harita için bir kütüphane belirlemek gerekmekte. Bu kütüphaneyi belirlerken, en önemli kriter işletim sistemi olacak. Çünkü Android işletim sistemli bir cihazda, doğrudan Apple haritalar kullanamıyoruz. Google haritalar hem sağladığı imkanlar, hem topluluk desteği, hem de yaygın kullanımı ile ön plana çıkıyor. Alternatif olarak Yandex haritalar ve OpenStreetMap seçenekleri de mevcut. Yandex haritalar, dökümantasyon yönünden biraz geride kalıyor. OpenStreetMap'in en büyük avantajı ise, açık kaynak bir kütüphane olması. Bu özelliği sayesinde, karşılaşabileceğimiz hatalara doğrudan kendimiz müdahale edebilir veya kullanıcıların müdahalelerini kullanabiliriz. Böylece hatayı en kısa sürede giderebiliriz.

Google haritaların maliyetinin yüksek olması nedeniyle, OpenStreetMap'i kullanma kararı verdim. Ancak uygulamanın artan kullanımına bağlı olarak, Google haritalara geçişin gerçekleşmesi gerekmekte. Çünkü gerek stillendirme, gerekse kullanım alışkanlığı gereği; diğer haritalara göre bir adım önde.

Mobil uygulamaların olmazsa olmaz yapısı da, uygulamanın "life cycle" adı verilen yaşam döngüsüdür. Kullanıcı uygulamayı açtığı andan itibaren, hangi ekranla karşılaşacak, hangi işlemi yaparsa ne gibi değişiklik olacak vb. Bu işlemi, Flutter'ın güçlü ve kabul görmüş kütüphanelerinden biri olan Provider ile yapacağım. Provider kütüphanesi ile uygulamamda, değişiklikleri kolayca render edebileceğim. Bu yenilemeyi de tüm ekranda değil, sadece ilgili kısımlarda yaparak gerçekleştireceğim. Böylelikle ekranda oluşabilecek olan data bozulması veya kasmaların da önüne geçmiş olacağım. State yönetimi sayesinde, loading olaylarını tek bir noktadan yürüteceğim. Herhangi bir bekleme anında, kullancıyı bilgilendirmiş olacağım.

Uygulamamda Model - View - View Model (MVVM) mimarisini kullandım. UI taraflardaki görünüm olaylarının hepsini View katmanımda gerçekleştirdim. Daha çok business işlemlerini de, View Model'ler ile gerçekleştirdim. Örneğin; bir listenin ekranda nasıl gözükeceği, ne yapınca yenileneceği, tıklanınca ne olacağına View katmanı karar veriyor. Listenin datalarının alınması, refresh ve onclick eventleri, loading aşamalarına ise View Model'ler karar veriyor. Bu sayede okunabilir ve karmaşıklıktan uzak bir yapı elde etmiş oldum.

View Model'lerin veya uygulama içinde kullanılan servislerin; factory, singleton gibi tasarım desenlerine uygun olarak alınması büyük önem arz ediyor. Gereksiz yeni nesnelerin oluşturulması, hem akışta karmaşıklık yaratabilir, hem de sistemi düzgün çalışmasını engelleyebilir. Bunun önüne geçmek için, Flutter'ın get_it kütüphanesini kullandım. Bu kütüphane ile, ViewModel gibi dart dosyalarımı factory ile, LocationService gibi dart dosyalarımı ise singleton yapısı ile almış oldum. Bunun için locator adından bir dart class'ı oluşturdum. Bu class'ın içinde tanımlamalarını yaptım.

Harita özelliğimiz olduğu için, mutlaka konum için de bir servis hazırlamam gerekti. Bunu da yine Flutter paketlerinden, location ile gerçekleştirdim. LocationService adında bir dart sınıfı oluşturdum ve içine izinleri almak için bir yapı kurdum. Bu yapı ile eğer kullanıcının izni yoksa, izin istiyor ve izin durumuna göre konum bilgisini elde etmemizi sağlıyor.

Uygulama içindeki sayfa geçişleri ve veri aktarımı için herhangi bir kütüphane kullanmadım. Fakat NavigationService adından bir dart sınıfı oluşturdum. Bu sınıfın içinde, sayfalara uygulanacak olan push ve pop olaylarını fonksiyonlar ile yönettim. Ayrıca her sınıfı ve linklerini, burada tanımladım. Böylelikle sayfaları, standart yapıya kavuşturmuş oldum.

Kaydettiğimiz konum verilerini, bir yerde depolamam ve kullanıcının tekrardan uygulamayı açtığında listelemem gerekiyordu. Bunu cihazın hafızasında yapmam gerekiyor çünkü bir API bağlantım yok. Bunun için ya mobile uyumlu bir veritabanı sistemi kullanacaktım ya da Shared Preferences ile saklayacaktım. Shared Preferences'a, string, int, bool, string listesi türünde veriler yazabiliyoruz. Bunları cihaz hafızasında, xml olarak saklıyor. Tablolu bir yapıya sahip olmaması nedeniyle, biraz NoSQL veritabanı sistemlerine benzetebiliriz. Ben olası API dönüşümü için, tablolu bir yapıyı tercih ettim. Bunun için de SQLite veritabanı sisteminin Flutter kütüphanesi olan Sqflite'ı kullandım. Veritabanı işlemlerini yönetebilmek için, CustomDatabase adından bir dart sınıfı oluşturdum. Bu sınıfın içinde, singleton tasarım desenine uygun olacak bir yapı kurdum. Bu yapı sayesinde, uygulama açıldığında, CustomDatabase adında bir nesne yaratılıyor. Daha sonraki tüm isteklerde, bu nesnenin örneği dönülüyor.

Oluşturduğum CustomDatabase sınıfının içinde, SQLite veritabanına ait Database nesnesini de sakladım. İlk olarak cihazın hafızasında, veritabanı nesnesinin olup olmadığı kontrol ediliyor. Kontrol sonrasında; eğer veritabanı dosyası yoksa, belirttiğim SQL sorgusunu çalıştırarak, bir veritabanı oluşturuyor. Bu SQL sorgusunun içinde; Latitude, Longitude gibi konum ve yorum verisini saklayabileceğim bir tablo oluşturdum. Yine sınıfın içinde, bu tabloya yapılacak SELECT, INSERT, UPDATE, DELETE fonksiyonlarını da yazdım. Bu sayede, veritabanı sistemini de tek noktadan yönetmiş oldum.

2.3 Harita kütüphanesi seçimi

Uygulamamızın temel işlevi olan Harita için bir kütüphane belirlemek gerekmekte. Bu kütüphaneyi belirlerken, en önemli kriter işletim sistemi olacak. Çünkü Android işletim sistemli bir cihazda, doğrudan Apple haritalar kullanamıyoruz. Google haritalar hem sağladığı imkanlar, hem topluluk desteği, hem de yaygın kullanımı ile ön plana çıkıyor. Alternatif olarak Yandex haritalar ve OpenStreetMap seçenekleri de mevcut. Yandex haritalar, dökümantasyon yönünden biraz geride kalıyor. OpenStreetMap'in en büyük avantajı ise, açık kaynak bir kütüphane olması. Bu özelliği sayesinde, karşılaşabileceğimiz hatalara doğrudan kendimiz müdahale edebilir veya kullanıcıların müdahalelerini kullanabiliriz. Böylece hatayı en kısa sürede giderebiliriz.

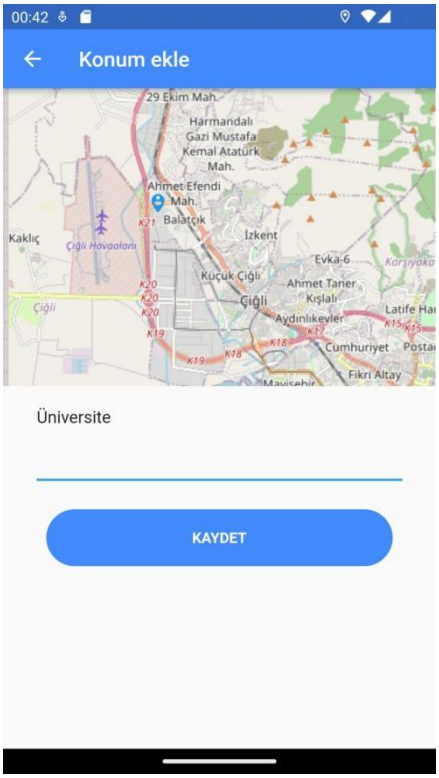
Google haritaların maliyetinin yüksek olması nedeniyle, OpenStreetMap'i kullanma kararı verdim. Ancak uygulamanın artan kullanımına bağlı olarak, Google haritalara geçişin gerçekleşmesi gerekmekte. Çünkü gerek stillendirme, gerekse kullanım alışkanlığı gereği; diğer haritalara göre bir adım önde.

2.4 Uygulamanın İşlevselliği

Gün içerisinde çeşitli yerlere uğrayabiliyoruz. Kimi zaman bir cafe, kimi zaman bir market, kimi zaman bir arkadaşımız evi. Daha sonrasında tekrardan bu konumlara uğramak veya birisine önermek isteyebiliriz. Eğer çeşitli mesajlaşma uygulamaları üzerinden konum paylaşmadıysak, buna ulaşma şansımız olmuyor. Konumu aklımızda tutsak dahi, harita üzerinden doğru noktayı bulmakta zorlanabiliyoruz.

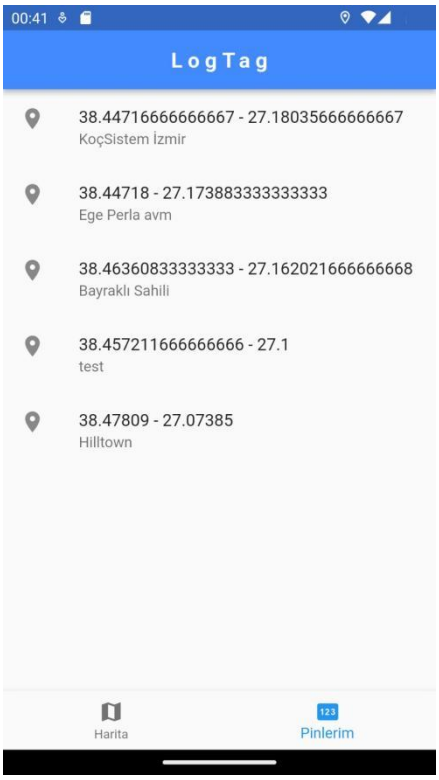
Bu sorunlara çözüm bulmak amacıyla geliştirdiğim uygulamamda, kullanıcılar haritadan anlık olarak bulunduğu konumları işaretleyebiliyor. Bu işaretlemelerine, ekstradan bir yorum metni de ilave edebiliyorlar. Yorum metni sayesinde, işaretledikleri konumun amacını yada kendilerine bir notunu kaydetmiş oluyorlar.

Konum kaydetme ekranında yer alan yarım harita üzerinde, işaretlenen konum gözlemlenebilmiş oluyor. Ayrıca yine yorum metninin de yazılması sağlanıyor. Daha sonrasında, bu işaretlenen noktalar, cihaz hafızasında saklanıyor.



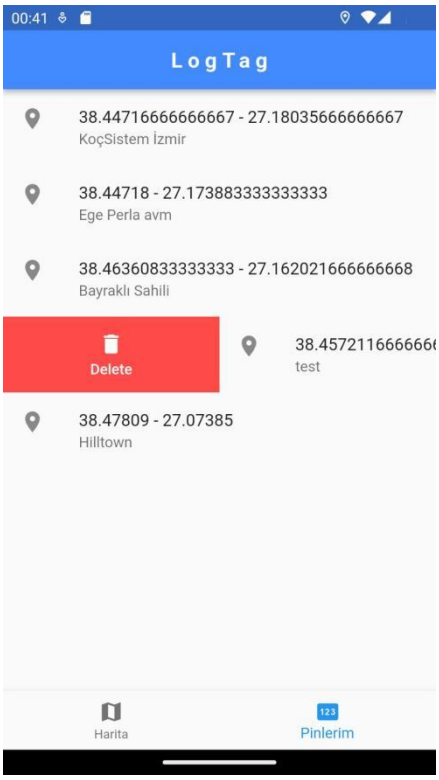
Şekil 2.1: Konum ekleme ekranı.

Daha önce işaretlediği noktaları, pinlerim sayfasından görüntüleyebiliyor. Liste yapısında konum bilgilerini ve varsa yorumlarını toplu şekilde görme fırsatı yakalayabiliyorlar.



Şekil 2.2: Pinlerim ekranı.

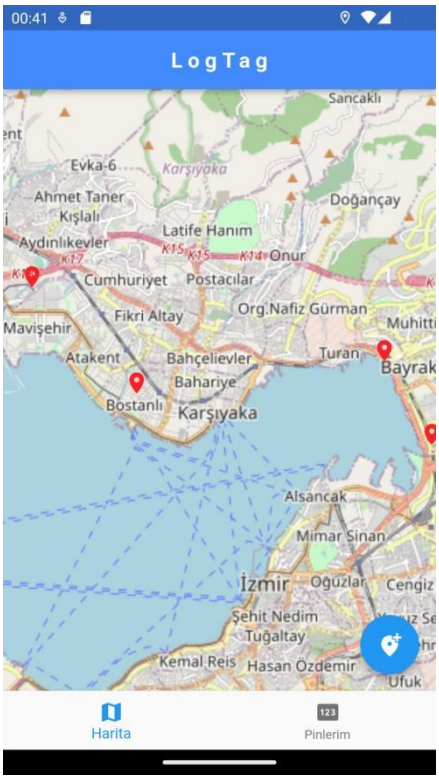
Liste üzerindeki konumlarında, girdikleri yorumları düzenleyebiliyorlar. Eğer yanlış bir konum girilirse veya artık kullanılmak istenmiyorsa bu konumların silinmesi de sağlanıyor. Genellikle iOS uygulamalarda yer alan sürükleyerek silme özelliği mevcut. Listeden istenen kayıtın sağa doğru sürüklenmesi ile, silme butonu devreye giriyor. Bu butona tıklanma ile kayıt siliniyor ve liste yenileniyor.



Şekil 2.3: Pinlerim ekranındaki silme özelliği.

Konumların liste görünümünün yanında, harita üzerinde pin görünümü de mevcut. Haritadaki yer alan pinler ile, konumları tam olarak oldukları noktalarda gözlemleyebiliyorlar. Ayrıca kendi konumlarını da takip edebildikleri haritada, yakınlarındanki işaretlenmiş pinleri kolayca ayırt edebiliyorlar.

Harita ekranında sağ altta yer alan buton ile, kullanıcı yeni pin kaydedebiliyor. Bu butona tıkladıktan sonra, LocationService sınıfından güncel konum bilgisi alınıyor. Bu konum bilgisi, Konum Ekle ekranına gönderiliyor. Bu sayede, 2 ekran arası veri aktarımını da gerçekleştirmiş oluyor.



Şekil 2.4: Harita ekranı.

Ekler

Özgeçmiş

Adı Soyadı:	Meriç Gerçeker
E-mail (1):	mericgerceker1997@gmail.com
E-mail (2):	meric.gerceker@ext.kocsistem.com.tr

Eğitim:

2015–2019	Süleyman Demirel Üniversitesi, Bilgisayar Müh., Lisans
2021–	İzmir Kâtip Çelebi Üniversitesi, Yazılım Müh., Yüksek Lisans

İş Deneyimi:

2020– 2021	Luna A.Ş - Yazılım AR&GE Mühendisi
2021	KoçSistem - Yazılım Mühendisi