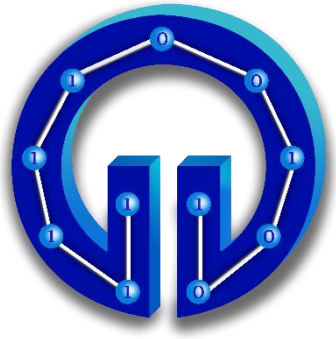
KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



KTÜ SOSYAL: KARADENİZ TEKNİK ÜNİVERSİTESİ İLETİŞİM VE SOSYALLEŞME PLATFORMU MOBİL UYGULAMASI

BİTİRME PROJESİ

Sercan TOR
İlyas AKTURAN
Halil İbrahim TİRGİL

2020-2021 GÜZ DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

KTÜ SOSYAL: KARADENİZ TEKNİK ÜNİVERSİTESİ İLETİŞİM VE SOSYALLEŞME PLATFORMU MOBİL UYGULAMASI

BİTİRME PROJESİ

Sercan TOR
İlyas AKTURAN
Halil İbrahim TİRGİL

2020-2021 GÜZ DÖNEMİ

IEEE Etik Kuralları
IEEE Code of Ethics

Mesleğime karşı şahsi sorumluluğumu kabul ederek, hizmet ettiğim toplumlara ve üyelerine en yüksek etik ve mesleki davranışta bulunmaya söz verdiğimi ve aşağıdaki etik kurallarını kabul ettiğimi ifade ederim:

1. Kamu güvenliği, sağlığı ve refahı ile uyumlu kararlar vermenin sorumluluğunu kabul etmek ve kamu veya çevreyi tehdit edebilecek faktörleri derhal açıklamak;
2. Mümkün olabilecek çıkar çatışması, ister gerçekten var olması isterse sadece algı olması, durumlarından kaçınmak. Çıkar çatışması olması durumunda, etkilenen taraflara durumu bildirmek;
3. Mevcut verilere dayalı tahminlerde ve fikir beyan etmelerde gerçekçi ve dürüst olmak;
4. Her türlü rüşveti reddetmek;
5. Mütenasip uygulamalarını ve muhtemel sonuçlarını gözeterek teknoloji anlayışını geliştirmek;
6. Teknik yeterliliklerimizi sürdürmek ve geliştirmek, yeterli eğitim veya tecrübe olması veya işin zorluk sınırları ifade edilmesi durumunda ancak başkaları için teknolojik sorumlulukları üstlenmek;
7. Teknik bir çalışma hakkında yansız bir eleştiri için uğraşmak, eleştiriyi kabul etmek ve eleştiriyi yapmak; hatları kabul etmek ve düzeltmek; diğer katkı sunanların emeklerini ifade etmek;
8. Bütün kişilere adilane davranmak; ırk, din, cinsiyet, yaş, milliyet, cinsi tercih, cinsiyet kimliği, veya cinsiyet ifadesi üzerinden ayırımcılık yapma durumuna girişmemek;
9. Yanlış veya kötü amaçlı eylemler sonucu kimsenin yaralanması, mülklerinin zarar görmesi, itibarlarının veya istihdamlarının zedelenmesi durumlarının oluşmasından kaçınmak;
10. Meslektaşlara ve yardımcı personele mesleki gelişimlerinde yardımcı olmak ve onları desteklemek.

IEEE Yönetim Kurulu tarafından Ağustos 1990'da onaylanmıştır.

ÖNSÖZ

“KTÜ Sosyal: Karadeniz Teknik Üniversitesi İletişim ve Sosyalleşme Platformu Mobil Uygulaması” isimli çalışma, Karadeniz Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü’nde bahar dönemi bitirme projesi olarak hazırlanmıştır.

Gerçekleştirilen projenin başlangıcından itibaren sonuna kadar her aşamada bize yardımcı olan, fikirlerini esirgemeyen ve bizi yönlendiren hocamız Doç. Dr. Bekir Dizdaroğlu’na teşekkür ederiz.

Ayrıca üniversite hayatımız boyunca bizden hiçbir desteğini esirgemeyen ailelerimize, hocalarımıza, arkadaşlarımıza ve sevdiklerimize gönülden teşekkür eder saygı ve sevgilerimizi sunarız.

Sercan TOR
İlyas AKTURAN
Halil İbrahim TİRGİL
Trabzon 2021

İÇİNDEKİLER

ÖZET

Proje konusunu seçerken kapsamının geniş olması, kullanıcılar tarafından kolay ulaşılabilir olması ve hedeflenen kitleye hitap etmesi istenmiştir. Bundan yola çıkarak günümüzde en çok kullanılan şeylerin başında gelen telefonlara üretilen mobil uygulamalar dikkatimizi çekmiştir. Mobil uygulamaların bu denli yüksek kullanılması ve yapılacak olan projeye daha uygun olması da tercih sebebi olmuştur. Kullanıcıların mobil cihaz tercihleri incelendiğinde Android ve iOS cihazların kullanılmasından dolayı her ikisinde de aynı anda çalışabileceğinden mobil uygulama geliştirme tercihi Dart dili ve Google Flutter Framework'ü tercih edilmiştir.

Uygulamayı inceleyecek olursak en başta KTÜ özelinde çalıştırılması düşünülen bu uygulama sadece KTÜ öğrencilerinin ve KTÜ akademisyenlerinin sahip olduğu özel, okula ait olan mail adresleriyle giriş yapılabilen bir uygulama olacaktır. Bu uygulamada öğrenciler ve akademisyenler aynı platformda buluşacak. Gerektiği durumlarda öğrenciler akademisyenlerle daha rahat iletişime geçebilecek, oluşturulan topluluklar ve katılım gösterilen gruplar sayesinde duyurular ve paylaşımlar daha düzenli ve sağlıklı bir hale gelecek, tercih durumuna bağlı olarak öğrenciler hobi toplulukları oluşturup kendi aralarında hem sosyal hem akademik etkinlikler yapabilecektir. Kullanımı tamamen kişiye bağlı olarak geliştirilen bu uygulamada hedeflenen şey okul özelinde genel bir topluluk oluşturmaktır.

Proje planlaması, iş paylaşımı ve iş ilerleyişi için web tabanlı bir uygulama olan Trello kullanılmıştır. Bu platform üzerinden ekip üyelerine dağıtılan görevler belirli zaman aralıklarıyla kontrol edildi ve iş ilerleyişi daha sağlıklı bir şekilde sağlanmıştır.

Görsel tasarımı Figma üzerinde tasarlanan bu uygulama yine Figma üzerinde yapmış olduğumuz prototiplere sadık kalınarak Flutter üzerinde tarafımızdan gerçeklenmiştir. Web tabanlı olan Figma platformunun tercih sebebi hem vektörel ve kaliteli tasarım yapılabilmesi hem de grup çalışmasına uygun bir şekilde ortak çalışma platformu olmasıdır. Bu platformda tasarlanan mobil uygulama sayfalarımız sade ve kullanıcıya hitap edecek şekilde Flutter'da kodlanmıştır.

Arayüz kullanıcıyı zorlamayan ve düşündürmeden işlem yapmaya müsait bir yapıya sahip olan uygulamamızda ana sayfamız, arama sayfamız, topluluklar sayfamız ve profil sayfalarımız kolayca ulaşılacak şekilde tasarlanmıştır.

Backend kısmında ise Express Framework'ü ve TypeScript dili yardımıyla yazılmış olan bir Node.JS API ile veri tabanı ve mobil uygulamanın yazıldığı Flutter arasında köprü görevi gören bir yapı oluşturulmuştur. API sayesinde mobil uygulama ve veri tabanı optimize ve senkron bir şekilde çalıştırılmıştır.

Veri tabanı kısmında ise SQL dilinin özelliklerinden faydalanan nesne ilişkisel tabanlı bir yapıya sahip olan PostgreSQL kullanılmıştır. Bu sayede geliştirilen uygulamadaki veriler depolanmıştır.

1. GENEL BİLGİLER

1.1. Giriş

Projeyi bu konu üzerine seçmemizin temel sebebi okul, fakülteler ve bölümlerin kendi özelinde oluşturduğu haberleşme grupları veya sosyal aktiviteler için okula bağlı kurulan kulüplerin farklı sosyal mecralar üzerinden kurulması denebilir. Ayrıca herkesin kullanmadığı veya kullanmayı tercih etmediği mecralar üzerinde kurulan, önemli duyuruların ve bilgilerin paylaşıldığı bu gruplara erişim sağlayamayan kullanıcılar da dikkate alınmıştır. Bu hususlar ele alındığında okulun kendi özelinde bir mobil uygulama sahibi olması fikri bize mantıklı gelmiştir ve proje konusu bu şekilde seçilmiştir.

Yapılan bu proje kapsamında, bu mobil uygulama kullanıcılar tarafından hem Android hem de iOS cihazlarına rahatça kurulacak bir şekilde tasarlandı. Projede ön yüz geliştirmesi responsive bir şekilde yapılmıştır ve her cihaza hitap eder hale getirilmiştir. Veri tabanı, api ve mobil uygulamadan oluşan bu proje kullanıcılarının bir araya gelip bir topluluk oluşturabildiği, hobilerine göre insanları bir araya getirip yeni sosyal ilişkilerin kurulabildiği, akademisyenlerine ulaşıp akademik anlamda konu danışma işleminin yapabildiği, gerektiği zaman tez veya proje arkadaşı arayıp bulabildiği bir ortam olacak şekilde tasarlanmıştır.

Kurulan bu platforma fotoğraflar yüklenebilir, verilen ödev içeriği buradan paylaşılabilir, önemli ders duyuruları oluşturulan gruplar üzerinden yapılabilir ve farklı kullanım şekillerinde kullanıcılar uygulamayı değerlendirebilir.

1.2. Veri Tabanı

Bu projede geliştirilen uygulama kullanıcılarının bilgilerinin tutulduğu, gruplarda paylaşılan fotoğrafların, gönderilerin, gönderilere yapılan yorumların ve buna benzer verilerin tutulduğu bir veri tabanı PostgreSQL ile oluşturuldu.

1.2.1. PostgreSQL

PostgreSQL özgür ve açık kaynak kodlu, SQL destekli bir ilişkisel veri tabanı yönetim sistemidir. [1]

PostgreSQL, geliştiricilerin ve proje yöneticilerinin büyük küçük fark etmeksizin yazılımlar geliştirilirken kullandıkları veriyi yönetmesine ve hata payı olmadan saklamasına olanak sağlar. Ücretsiz ve açık kaynak kodlu olması sebebiyle de diğer veri tabanlarının aksine tamamen uyarlanabilir bir yapıdadır. Birçok majör işletim sistemi ve yine aynı zamanda birçok programlama dili tarafından desteklenmesi de kullanım sebeplerinden biridir. [2]

1.3. Arayüz Tasarımı

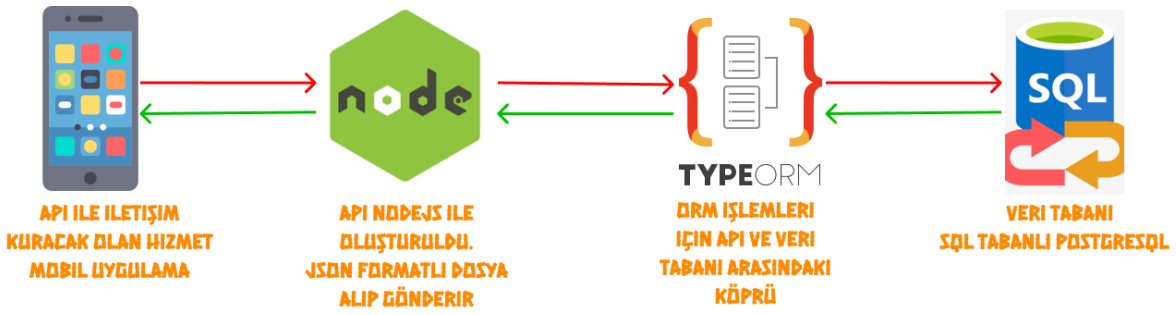
1.3.1. Figma

Figma farklı ekran görüntülerine göre tasarım yapmanıza olanak sağlayan, birden fazla kişi ile senkronize olarak çalışabildiğiniz tüm bilgisayarlarda çalışabilen, farklı türlerde çıktı alabildiğiniz bir arayüz tasarım aracıdır. Figma tarayıcıda, Windows işletim sistemlerinde ve macOS cihazlarda da çalışabilir. Yapılan çalışmaların farklı ekran boyutlarında nasıl göründüğü de bu platformda desteklenen özelliklerden biridir. Ayrıca bileşen kütüphanesi oluşturma özelliği de destekleyen bu platform sayesinde oluşturulan bileşenler etkili bir şekilde kullanılır ve proje takımıyla da rahatlıkla paylaşılabilir. [3]

Bu platformun kullanım amacı ön yüz tasarımıdır. İlk başta kağıt üstünde modellenilen uygulama sayfa dizaynları bilgisayar ortamına taşınmalıydı. Grup çalışmasına ve ortak bir şekilde değişiklik yapmaya uygun olan bu platform üzerinden projede geliştirilen uygulamanın ara yüz tasarımı kaliteli bir şekilde tasarlandı. Proje kodlanmaya başlandığında kafa karışıklıkları bu sayede engellenmiş oldu. Planlı bir şekilde ilerlemek açısından faydalı gördüğümüz bu platform yardımıyla ara yüz tasarımızı gerçekleştirmiş olduk.

1.4. Sunucu

Projede okulun öğrencilerine sağladığı imkanlardan da faydalanarak Microsoft'un bir servisi olan Azure Web Server kiralandı.



Şekil 1.1 [9]

1.4.1. Typescript

TypeScript açık kaynak kodlu bir programlama dilidir. Microsoft tarafından geliştirilen bu dil desteğini de aynı zamanda Microsoft'tan alır. TypeScript içindeki derleyici sayesinde yazılan kodu JavaScript koduna çevirir. Gerek istemci taraflı, gerekse sunucu taraflı yazılım geliştirmede kullanılabilmektedir. [4]

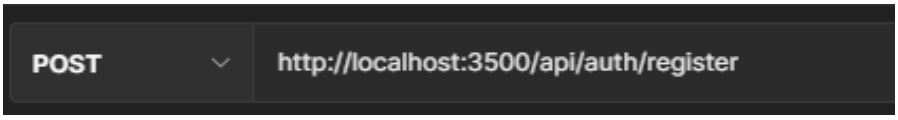
TypeScript, JavaScriptin bir supersetidenebilir, JavaScriptin varsayılan olarak sunmadığı tipli (strong-typed) stili bize varsayılan olarak sunar. Bu durum, geliştiricinin kod yazarken daha az hata yapmasında önemli bir rol oynar.

Bu projedeki kullanım rolü ise uygulama ve veri tabanı arasında kurulacak iletişim için gerekli olan API içindir. Projenin içinde barındırdığı API bu dille kodlanmıştır.

1.4.2. JSON

JSON (JavaScript Object Notation), JavaScript alt yapısı üzerine kurulmuş veri aktarım formatıdır. JSON'un geliştirilme sebebi XML yapısının hem JavaScript'e uygun olmaması hem de daha çok yer kaplamasıdır. [5]

JSON verisi key ve value değerinden oluşur. Key değeri tutulacak verinin açıklamasıyken value değeri ise tutulacak veriye girilen parametre gibi yorumlanabilir. Geliştirdiğimiz projeye ait kullanıcının kayıt olma aşamasında yollanan HTTP isteği (Şekil 1.2), yollanan JSON verisi (Tablo 1.1) ve response JSON verisi (Tablo 1.2) aşağıdaki gösterildiği gibidir.



Şekil 1.2 İstek Yollanacak Endpoint

Tablo 1.1 Manuel Olarak Gönderilen JSON Verisi

```
{
    "name": "halil tirgil",
    "fieldOfStudy": "Bilgisayar Mühendisliği",
    "email":"348357@ogr.ktu.edu.tr",
    "password":"123456"
}
```

Tablo 1.2 Response JSON Verisi

```
{
    "id": "5452c1cd-8b49-40bf-82d8-8d8b6e031425",
    "name": "halil tirgil",
    "password": "$2b$10$HH3vbwuyCbGf/6itSCF6/.fy2sRtJn8tcKEcluhlysZ6jQHs09vym",
    "fieldOfStudy": "Bilgisayar Mühendisliği",
    "role": null,
    "email": "348357@ogr.ktu.edu.tr",
"photo_url":“http://bis.ktu.edu.tr/personel/a5f2cf78b1260c9726fb4dba2f3c6e4d.jpg”,
    "created": "2021-06-14T13:02:47.377Z"
}
```

1.4.3. REST

REST bir veri transfer yöntemidir. **Re**presentational **S**tate **T**ransfer kavramının kısaltmasıdır. İstemci ve sunucu arasındaki iletişimin hızlı ve sağlıklı bir şekilde yapılmasını sağlar ve bunun için HTTP metodlarını kullanır.

HTTP metodları 4 ana başlıkta incelenebilir. GET, PUT, POST ve DELETE. Örneklerle anlatacak olursak GET metodu kullanıcı oluşturmayı temsil ederken, PUT ve POST kullanıcı verisi göndermeyi (güncellemeyi); DELETE ise kullanıcı bilgilerini silmeyi temsil eder.

Örnek olarak bütün kullanıcıların response'unu alabileceğimiz bir uç noktamız olsun. Bu uç noktayı yazarken, URI'ımız

GET (HTPP Verb)
```
/api/get-users
```

Olarak değil

GET (HTPP Verb)
```
/api/users
```

Olarak tanımlanmıştır, yani GET HTTP fiili, bu uç noktanın isimlendirilmesinde kritik bir rol oynar. REST mimarisinde URI'ların isimlendirilmesiyle birlikte önemli olan diğer bir konu ise API tarafından geri dönülen HTTP Response kodlarıdır. Projede şöyle bir uç nokta olsun:

GET (HTPP Verb)
```
/api/users/:userId/posts
```

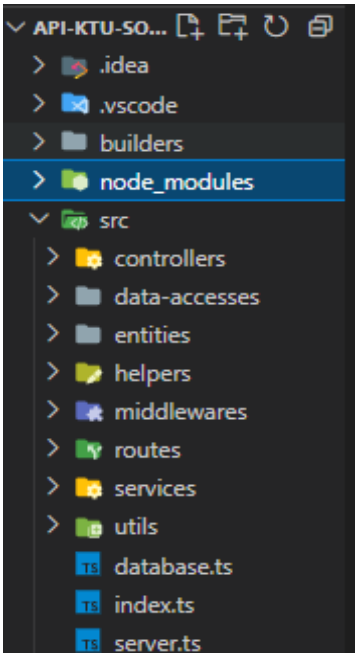
Bu uç noktanın döndürmesi gereken HTTP kodu 200 dür. Bir başka örnek olarak; eğer veri tabanında bir kayıt oluşturma işlemi var ise client'a döndürülen HTTP kodu 201 olmalıdır. Projede bu mimariye dikkat edilerek API inşaa edilmiştir.

1.4.4 Node.JS

Node.Js bir JavaScript runtime programıdır. Yani; JavaScript programlarının desktop ortamında çalışmasını sağlar. Arka tarafında Google Chrome'unözverili çalışan V8 motoru vardır.

Projede Node.JS kullanma sebebi API tasarım frameworklerinin oldukça sezgisel ve el verişli olmasıdır. Ayrıca Node.JS ile birlikte kullandığımız npm paket yöneticisi hem API'i deploy ederken hem de inşaa ederken kolaylık sağlamıştır.

Node.JS tek thread ile çalışan, ancak non-blocking olan bir runtimedır. Kendiliğinden asenkron temellidir. Bunu JavaScript'in event-loop ve callback fonksiyonlarıyla gerçekleştirir. Ancak Node.JS'in darboğaz olduğu konu CPU gerektiren işlemlerdir. Projedeki API çok CPU gerektirmediğinden, Node.JS proje için uygun bir adaydır. Ayrıca projede gelecekte mesajlaşma özelliği getirilmek istenirse Node.JS'in soket implementasyonu kolayca yerleştirilebilir.



Şekil 1.3 API Dosya Yapısı

1.5. Git

Git, yazılım geliştirme süreçlerinde kullanılan, hız odaklı, dağıtık çalışan bir sürüm kontrol ve kaynak kod yönetim sistemidir. İlk sürümü Linux çekirdeği'nin geliştirilmesinde kullanılmak üzere 2005 yılında bizzat Linus Torvalds tarafından tasarlanıp geliştirilmiş, son Eclipse kullanıcı topluluğu anketi verilerine göre 2013 yılı itibarıyla %30 pazar payına ulaşmıştır.

Git sürüm kontrol sistemini kullanan her bir çalışma dizini (proje), internet erişimi ya da merkezi bir depo olmaksızın tüm tarihçeyi tutan ve sürüm kontrol sisteminin tamamını içinde barındıran tam yetkili birer depodur. Ayni çalışma dizininin birçok depodan birindeki kopyasında yapılan değişiklikler diğerlerine güven temelli bir değerlendirmeyle kabul edilir; Güvenilmeyenden değişiklik alınmaz, o kendi ayrı sürümünü geliştirmeye devam eder. [6]

Git platformunun bu projedeki rolü açıklanacak olursa, üstlenilen proje hem grup çalışması ile gerçekleştirildiği için hem de karmaşık bir yapıya sahip olduğu için sürekli değişiklikler ve güncellemeler yapılması gerekir. Proje üyeleri ilk önce geliştirilen API/Mobil uygulama şekil alana kadar tek bir dala (branch) daha sonra özellik (feature) branchleri açarak projenin gelişimine devam edildi.

Git branchleri açılırken, belirli bir yapı (convention) kullanıldı. Örneğin grup ekranlarını eklemek istediğimizde, branch ismi

```
feature/groupScreen
```

olarak eklendi, daha sonra master branchine Merge Request açılarak, aramızdan seçilen biri tarafından review edilip merge edildi.

1.6. CI/CD

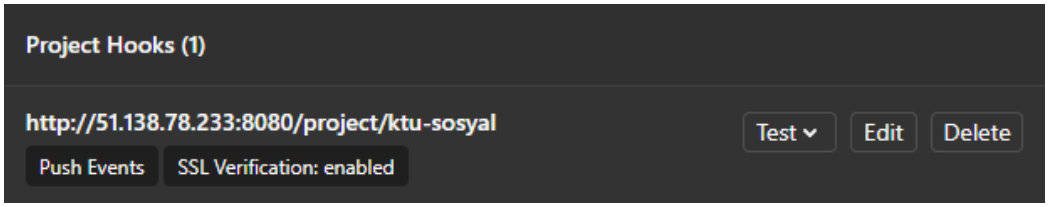
CI/CD, uygulama geliştirme aşamalarına otomasyonu entegre ederek uygulamaları müşterilere kesintisiz şekilde dağıtılması yöntemidir. [10]

1.6.1. Jenkins

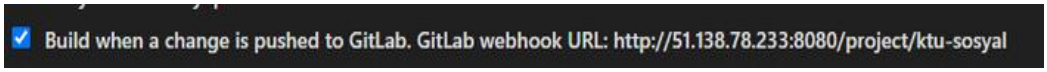
Git kullanılarak proje bulut üstünden yönetelebilir bir hale getirilir. Jenkins otomasyonunun projedeki görevi ise şu şekilde açıklanabilir:

Webhooklar yardımıyla bulut üzerinde oluşan değişiklikleri algılar ve geliştiricinin istediği komutları otomatik olarak koşmasını sağlar. Jenkins birçok dil ve eklenti desteğine sahiptir. Bu da projeyi geliştirirken geliştiricilere kolaylıklar sağlamaktadır.

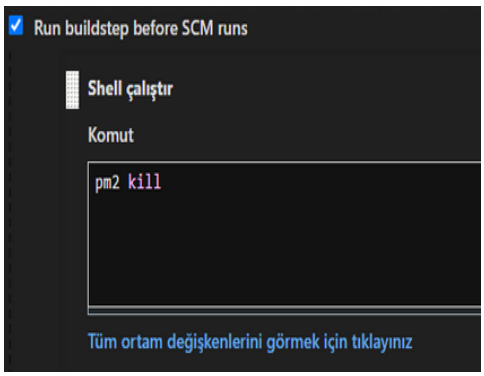
Aşağıdaki şekiller (Şekil 1.4 ve Şekil 1.5) build işlemi yapılmadan önce koşulması gereken işlemlerdir.



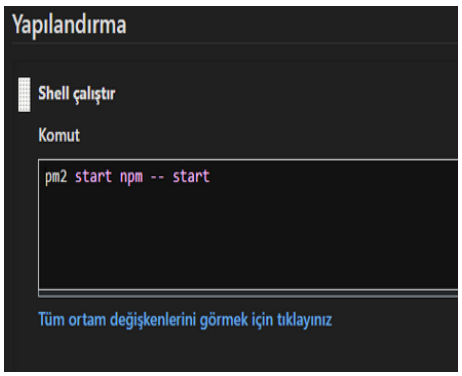
Şekil 1.4 Projede Değişiklik Olduğunda İstek Gönderilen URL



Şekil 1.5 Adrese Gelen İsteği Algılama



Şekil 1.6 Build Öncesi Komut



Şekil 1.7 Build Sonrası Komut

1.7. Mobil Uygulama

Mobil uygulama geliştirilirken hitap edilen cihazlar daha çok telefonlar ve tabletlerdir. Telefon ve tabletlerin en çok tercih ettiği işletim sistemleri ise Android ve iOS olarak bilinir. Android cihazlara uygulama geliştiriliyorsa daha çok Kotlin, iOS cihazlara uygulama geliştiriliyor ise daha çok native olarak görülen Swift dili kullanılır. Geliştirdiğimiz uygulama kullanıcıları hem Android hem de iOS cihaz sahibi olabileceğinden ötürü fazla iş yükünün önüne geçmek için geliştirme çalışmalarımızı cross-platform olan Flutter'ı kullanıldık.

1.7.1. Flutter ve Dart

Flutter, Google tarafından oluşturulan ve Mayıs 2017'de yayınlanan ücretsiz ve açık kaynaklı bir mobil UI frameworktur. Birkaç kelimeyle, yalnızca bir kod tabanıyla yerel bir mobil uygulama oluşturmanıza olanak tanır. Bu, iki farklı uygulama (iOS ve Android için) oluşturmak için bir programlama dili ve bir kod tabanı kullanabileceğiniz anlamına gelir. [7]

Flutter SDK ve framework olarak iki önemli kısımdan oluşur. SDK uygulama geliştirirken size yardımcı olan bir araç koleksiyonudur ve kodunuzu yerel makine koduna derlemek için araçlar içerir. Framework ise ihtiyaçlara göre kişiselleştirilebilen UI öğeleri içerir. Flutter sayesinde kodu anlık olarak değiştirebilir ve hot reload özelliği kullanılarak yapılan anlık değişiklikler uygulamada gözlemlenir. Bu geliştiriciye performanslı bir şekilde kodlama fırsatı sunar.

C, C++ ve Dart dili ile kodlanan bu platform sayesinde birçok kütüphaneden yararlanılabilir. Diğer mobil uygulama geliştirme ortamlarının aksine ağaç mantığı ile çalışan Flutter, içinde birçok framework, widget ve kütüphane barındırır. Flutter içerisinde kodlama yapılırken, nesne tabanlı olan ve söz dizimi C, C++ dillerine çok benzeyen Dart dilini kullanır.

Flutter, Google destekli olduğu için sürekli gelişen her türlü güncel teknolojiden faydalanır haldedir. Ayrıca açık kaynak kodlu olduğu için sürekli olarak gelişime açıktır.

2. YAPILAN ÇALIŞMALAR

Flutter Framework ile kodlanan bu proje MVVM mimarisine uygun bir şekilde kodlanmıştır. Projede isimlendirme olarak camelCase tercih edilmiştir ve kodlanırken fonksiyonel kodlamaya dikkat edilmiştir. Projede State yönetiminin verimli bir şekilde sağlanması için Provider mimarisinden faydalanılmıştır.

Provider mimarisi Notifier ve Consumer yapılarını da içinde barındırır. [8] Bu mimariyi hayattan bir örnek vererek açıklayanacak olursa şu şekilde ifade edilebilir:

Bir restoranta giden müşteri yiyeceğini bekler. Müşteri yiyecek hazır olduğunda bunu Notifier ile öğrenir. Garson ise yiyeceği müşteriye getirir yani bu durumda garson Provider'ı temsil eder. Son olarak da bu yiyeceği tüketecek kişi yani müşteri de bu denklemde Consumer'ı temsil eder.

Uygulama arası sayfa geçişleri ise gezinti çubuğu mimarisiyle entegre bir şekilde rooting şekilde yönlendirmeler ile yapılmaktadır.

Yapılan çalışmaları alt başlıklar halinde inceleyecek olursak:

2.1. Kullanıcı Kaydı ve Kullanıcı Girişi

Kullanıcıların sisteme kayıt olması ve kayıt olan bu kullanıcıların veri tabanında tutulması için API üzerinden gönderilen istek ile alınan bilgiler eşliğinde kullanıcı kaydı verilen bilgilere göre gerçekleştirilir. Sisteme kayıt esnasında kullanıcıdan okul maili, şifresi, hangi bölümde okuduğu, adı ve soyadı gibi bilgiler alınır.

Kayıt olan kullanıcının uygulamaya giriş esnasından itibaren hesabının aktif olarak uygulamada açık kalması için JWT token kullanılmıştır. Buna göre kullanıcı uygulamada belirlenen süre boyunca aktif bir şekilde uygulamayı kullanabilir. Bu işlemlerin yapıldığı kod parçaları ve sonucunda tasarlanan uygulama ekran çıktıları aşağıdaki gibidir.

Tablo 2.1. Giriş Yapma ve Kayıt Olma Kaynak Kodu

```
class SignInSignUp extends StatefulWidget {
  @override
  _SignInSignUpState createState() => _SignInSignUpState();
}

class _SignInSignUpState extends State<SignInSignUp> {
  final List<bool> isSelected = [true, false];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      extendBody: true,
      backgroundColor: Colors.white,
      body: SingleChildScrollView(
        child: Column(
          children: [
            Stack(
              children: [
                Container(
                  height: MediaQuery.of(context).size.height / 1.5,
                  decoration: BoxDecoration(
                    borderRadius: BorderRadius.only(
                      bottomLeft: Radius.elliptical(100, 100),
                      bottomRight: Radius.elliptical(100, 100),
                    ),
                    gradient: KtuColors.gradientLogin,
                  ),
                ),
                Column(
                  children: [
```

```
                    _buildKtuLogo(),
                    _buildToggleButtons(),
                    // The box that containst the forms
                    Consumer<SignInValidation>(
                      builder: (context, signInValidationProvider, _) {
                        return Padding(
                           padding: EdgeInsets.only(
                              top: MediaQuery.of(context).size.height / 40.0),
                          child: Container(
                            // If we're in the sign in form, make the box
smaller!
                            height: signInValidationProvider
                                        .toggleLoginLogout ==
                                    false
                                ? (MediaQuery.of(context).size.height / 1.8)
                                : (MediaQuery.of(context).size.height / 1.6),
                            width: MediaQuery.of(context).size.width / 1.3,
                            decoration: BoxDecoration(
                              color: Colors.white,
                              borderRadius: BorderRadius.circular(20.0),
                              boxShadow: [
                                BoxShadow(
                                  color: Colors.black26,
                                  spreadRadius: 8,
                                  blurRadius: 10,
                                  offset: Offset(0, 3),
                                ),
                              ],
                            ),
                            child: SingleChildScrollView(
                              child: Column(
                                children: [
                                  signInValidationProvider.toggleLoginLogout ==
                                          false
                                      ? SignInForm()
                                      : SignUpForm(),
                                ],
                              ),
                            ),
                          ),
                        );
                      },
                    )
                  ],
                ),
              ],
            ),
          ],
        ),
      ),
    );
  }

  Widget _buildKtuLogo() {
    return Padding(
      padding: EdgeInsets.only(top: MediaQuery.of(context).size.height / 15),
      child: Align(
        child: SvgPicture.asset(
```

```
          LogoConstants.ktuLogo,
          height: MediaQuery.of(context).size.height / 8,
        ),
        alignment: Alignment.center,
      ),
    );
  }

  Widget _buildToggleButtons() {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 15.0),
      child: Container(
        height: 35.0,
        decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(50.0),
            color: KtuColors.ktuDarkBlue),
        child: Consumer<SignInValidation>(
          builder: (context, signInValidationProvider, _) {
            return ToggleButtons(
              borderRadius: BorderRadius.circular(20.0),
              selectedColor: Colors.black,
              fillColor: KtuColors.ktuWhite,
              renderBorder: false,
              color: Colors.white,
              children: <Widget>[
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 50.0),
                  child: Text('Öğrenci', style: KtuTextStyles.regular),
                ),
                Padding(
                  padding: const EdgeInsets.symmetric(horizontal: 30.0),
                  child: Text('Akademisyen', style: KtuTextStyles.regular),
                ),
              ],
              onPressed: (int index) {
                setState(() {
                  if (index == 0) {
                    isSelected[index] = true;
                    isSelected[index + 1] = false;
                  } else {
                    isSelected[index] = true;
                    isSelected[index - 1] = false;
                  }
                  signInValidationProvider.setEmailType(index);
                });
              },
              isSelected: isSelected,
            );
          },
        ),
      ),
    );
  }
}
```

Yukarıdaki kod dizininde bir sayfa oluşturulur ve yazılan özel widgetlar ile toggle yaparak olmak istenilen sayfa seçelir. Uygulama açıldığında kullanıcıyı karşılayan sayfanın kodunun bir kısmı bu şekildedir.

SignInForm (Tablo 2.3) ve SignUpForm (Tablo 2.2) adı altında yazılan iki özel widget ile giriş yapma ve kayıt olma sayfaları kullanıcıya gösterilir. State yönetimini daha iyi sağlamak ve sayfalar arası bilgileri daha düzenli bir şekilde işlemek için kullanılan provider yöntemi yardımıyla ise kullanıcı bilgileri alınır ve tutulur.

Tablo 2.2 SignUp Form Kaynak Kodu

```
class SignUpForm extends StatefulWidget {
  @override
  _SignUpFormState createState() => _SignUpFormState();
}

class _SignUpFormState extends State<SignUpForm> {
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  final TextEditingController _nameController = TextEditingController();
  final TextEditingController _surnameController = TextEditingController();
  final TextEditingController _fieldOfStudyController = TextEditingController();

  String errorText = "";
  var _isLoading;
  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    _nameController.dispose();
    _surnameController.dispose();
    _fieldOfStudyController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Consumer<SignInValidation>(
      builder: (context, signInValidatonProvider, _) {
        return Column(
          children: [
            Container(
              padding: EdgeInsets.symmetric(
                  horizontal: MediaQuery.of(context).size.width / 15),
              // Asking for name of the user
              child: TextFormField(
                decoration: InputDecoration(
                    labelText: 'Ad',
                    labelStyle: KtuTextStyles.regular,
                    errorText: signInValidatonProvider.name.error),
                controller: _nameController,
              ),
            ),

            Container(
              padding: EdgeInsets.symmetric(
```

```
                horizontal: MediaQuery.of(context).size.width / 15),
            // Asking for surname of the user
            child: TextFormField(
              decoration: InputDecoration(
                  labelText: 'Soyad',
                  labelStyle: KtuTextStyles.regular,
                  errorText: signInValidatonProvider.surname.error),
              controller: _surnameController,
            ),
          ),

          Container(
            padding: EdgeInsets.symmetric(
                horizontal: MediaQuery.of(context).size.width / 15),
            // Asking for fielOfStudy of the user
            child: TextFormField(
              decoration: InputDecoration(
                  labelText: 'Bölüm',
                  labelStyle: KtuTextStyles.regular,
                  errorText: signInValidatonProvider.fieldOfStudy.error),
              controller: _fieldOfStudyController,
            ),
          ),

          Row(
            mainAxisSize: MainAxisSize.min,
            children: [
              // Asking for E-mail
              Expanded(
                child: Container(
                  padding: EdgeInsets.symmetric(
                      horizontal: MediaQuery.of(context).size.width / 15),
                  child: TextFormField(
                      style: KtuTextStyles.regular,
                      decoration: InputDecoration(
                          labelText: 'e-mail',
                          labelStyle: KtuTextStyles.regular,
                          errorText: signInValidatonProvider.email.error),
                      controller: _emailController),
                ),
              ),
              Expanded(
                child: Container(
                  padding: EdgeInsets.only(
                      top: MediaQuery.of(context).size.height / 30),
                  child: Text(signInValidatonProvider.emailtype,
                      style: KtuTextStyles.bold),
                ),
              )
            ],
          ),

          Container(
            padding: EdgeInsets.symmetric(
                horizontal: MediaQuery.of(context).size.width / 15),
            // Asking for password
            child: TextFormField(
              obscureText: true,
              decoration: InputDecoration(
```

```
                labelText: '$ifre',
                labelStyle: KtuTextStyles.regular,
                errorText: signInValidatonProvider.password.error),
            controller: _passwordController,
          ),
        ),

        // Giving space in between maybe value changed
        SizedBox(height: MediaQuery.of(context).size.height / 10000),

        // This is a bad way of doing this.
        _isLoading == true ? CircularProgressIndicator() : Container(),
        Container(
          width: MediaQuery.of(context).size.width / 3.5,
          height: MediaQuery.of(context).size.height / 20,
          child: Text(errorText),
        ),
        // Sign up button
        Container(
          width: MediaQuery.of(context).size.width / 3.5,
          height: MediaQuery.of(context).size.height / 20,
          decoration: BoxDecoration(
            borderRadius: BorderRadius.circular(20.0),
            color: Colors.green,
            boxShadow: [
              BoxShadow(
                color: Colors.black12,
                spreadRadius: 1,
                blurRadius: 4,
                offset: Offset(0, 3),
              )
            ],
          ),
          child: Consumer<AuthProvider>(
            builder: (context, authProvider, _) {
              // ignore: deprecated_member_use
              return OutlineButton(
                highlightedBorderColor: Color(0xFF555555),
                shape: RoundedRectangleBorder(
                    borderRadius: new BorderRadius.circular(20.0)),
                child: Text(
                  'Kayıt ol',
                  style: KtuTextStyles.regularWhite,
                ),
                onPressed: () async {
                  signInValidatonProvider.setName(_nameController.text);
                  signInValidatonProvider
                      .setSurname(_surnameController.text);
                  signInValidatonProvider
                      .setFieldOfStudy(_fieldOfStudyController.text);
                  signInValidatonProvider.setEmail(_emailController.text +
                      signInValidatonProvider.emailtype);
                  signInValidatonProvider
                      .setPassword(_passwordController.text);

                  String s = await authProvider.signUp(
                      _nameController.text,
                      _surnameController.text,
                      _fieldOfStudyController.text,
```

```
                        _emailController.text +
                            signInValidatonProvider.emailtype,
                        _passwordController.text);

                    if (signInValidatonProvider.password.error == null &&
                        signInValidatonProvider.email.error == null) {
                      setState(() {
                        errorText = s;
                      });
                    }
                  },
                );
              },
            ),
          ),

          // Giving space in between
          SizedBox(height: MediaQuery.of(context).size.height / 100),

          // Redirect to sign in
          Container(
            width: MediaQuery.of(context).size.width / 3.5,
            height: MediaQuery.of(context).size.height / 20,
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(20.0),
              color: Color(0xFF555555),
              boxShadow: [
                BoxShadow(
                  color: Colors.black12,
                  spreadRadius: 1,
                  blurRadius: 4,
                  offset: Offset(0, 3),
                )
              ],
            ),
            // ignore: deprecated_member_use
            child: OutlineButton(
              highlightedBorderColor: Colors.green,
              shape: RoundedRectangleBorder(
                  borderRadius: new BorderRadius.circular(20.0)),
              child: Text(
                'Geri',
                style: KtuTextStyles.regularWhite,
              ),
              onPressed: () {
                signInValidatonProvider.setToggleLoginLogout();
              },
            ),
          ),
        ],
      );
    },
  );
 }
}
```

Tablo 2.3 SignIn Form Kaynak Kodu

```
class SignInForm extends StatefulWidget {
  @override
  _SignInFormState createState() => _SignInFormState();
}

class _SignInFormState extends State<SignInForm> {
  final TextEditingController _passwordController = TextEditingController();
  final TextEditingController _emailController = TextEditingController();
  bool _isLoading = false;

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
    super.dispose();
  }

  _toggleLoading() {
    setState(() {
      _isLoading = !_isLoading;
    });
  }

  @override
  Widget build(BuildContext context) {
    bool status = false;
    return Consumer<SignInValidation>(
      builder: (context, signInValidatonProvider, _) {
        return Column(
          children: [
            Row(
              mainAxisSize: MainAxisSize.min,
              children: [
                // Asking for E-mail
                Expanded(
                  child: Container(
                    padding: EdgeInsets.symmetric(
                        horizontal: MediaQuery.of(context).size.width / 15),
                    child: TextFormField(
                        style: KtuTextStyles.regular,
                        decoration: InputDecoration(
                            labelText: 'e-mail',
                            labelStyle: KtuTextStyles.regular,
                            errorText: signInValidatonProvider.email.error),
                        controller: _emailController),
                  ),
                ),
                Expanded(
                  child: Container(
                    padding: EdgeInsets.only(
                        top: MediaQuery.of(context).size.height / 30),
                    child: Text(signInValidatonProvider.emailtype,
                        style: KtuTextStyles.bold),
                  ),
                )
              ],
            ),
```

```
Container(
  padding: EdgeInsets.symmetric(
      horizontal: MediaQuery.of(context).size.width / 15),

  // Asking for password
  child: TextFormField(
    obscureText: true,
    decoration: InputDecoration(
        labelText: 'Şifre',
        labelStyle: KtuTextStyles.regular,
        errorText: signInValidatonProvider.password.error),
    controller: _passwordController,
  ),
),
// Giving space in between
SizedBox(height: MediaQuery.of(context).size.height / 50),
_isLoading == true
    ? Container(
        padding: EdgeInsets.all(8.0),
        child: CircularProgressIndicator())
    : Container(),
// Sign in button
Container(
  width: MediaQuery.of(context).size.width / 3.5,
  height: MediaQuery.of(context).size.height / 20,
  decoration: BoxDecoration(
    borderRadius: BorderRadius.circular(20.0),
    color: Colors.green,
    boxShadow: [
      BoxShadow(
        color: Colors.black12,
        spreadRadius: 1,
        blurRadius: 4,
        offset: Offset(0, 3),
      )
    ],
  ),
  child:
      Consumer<AuthProvider>(builder: (context, authProvider, _) {
    // ignore: deprecated_member_use
    return OutlineButton(
      highlightedBorderColor: Colors.green,
      shape: RoundedRectangleBorder(
          borderRadius: new BorderRadius.circular(20.0)),
      child: Text(
        'Giriş Yap',
        style: KtuTextStyles.regularWhite,
      ),
      onPressed: () async {
        signInValidatonProvider.setEmail(_emailController.text +
            signInValidatonProvider.emailtype);
        signInValidatonProvider
            .setPassword(_passwordController.text);
        if (signInValidatonProvider.password.error == null &&
            signInValidatonProvider.email.error == null) {
          _toggleLoading();
          status = await authProvider.signIn(
              _emailController.text +
```

```
                              signInValidatonProvider.emailtype,
                          _passwordController.text);

                      if (status == false) {
                        _toggleLoading();
                        ScaffoldMessenger.of(context).showSnackBar(SnackBar(
                            content: Text("Kullanıcı adı ya da şifre hatalı"),
                            duration: Duration(seconds: 5)));
                      }
                    }
                  },
                );
              }),
            ),
            // Giving space in between
            SizedBox(height: MediaQuery.of(context).size.height / 50),

            // Redirect to sign up
            Container(
              width: MediaQuery.of(context).size.width / 3.5,
              height: MediaQuery.of(context).size.height / 20,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(20.0),
                color: Color(0xFF555555),
                boxShadow: [
                  BoxShadow(
                    color: Colors.black12,
                    spreadRadius: 1,
                    blurRadius: 4,
                    offset: Offset(0, 3),
                  )
                ],
              ),
              // ignore: deprecated_member_use
              child: OutlineButton(
                  highlightedBorderColor: Color(0xFF555555),
                  shape: RoundedRectangleBorder(
                      borderRadius: new BorderRadius.circular(20.0)),
                  child: Text(
                    'Kayıt ol',
                    style: KtuTextStyles.regularWhite,
                  ),
                  onPressed: () {
                    signInValidatonProvider.setToggleLoginLogout();
                  }),
            ),

            // Giving space in between
            SizedBox(height: MediaQuery.of(context).size.height / 100),
            SizedBox(height: MediaQuery.of(context).size.height / 100),
          ],
        );
      },
    );
  }
}
```

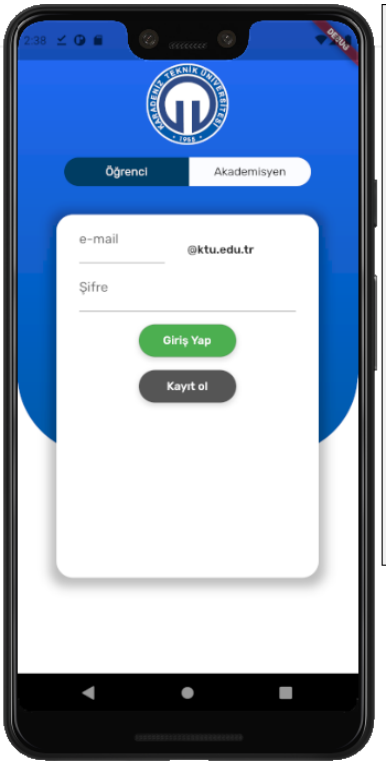
Yazılan RegExp yardımıyla (Tablo 2.4) kullanıcıdan alınan mail istenilen formda alınır ve oluşabilecek hataların önüne geçilir. Ayrıca ayarlanan mail formu yardımıyla kullanıcının Karadeniz Teknik Üniversitesi bünyesinde verilen üniversiteye özel mail adresi dışında bir mail girmesi önlenerek gerçek üniversite kullanıcıları bulunmuş olunur.

Tablo 2.4 Kullanıcı Girişi Provider Mimarisi

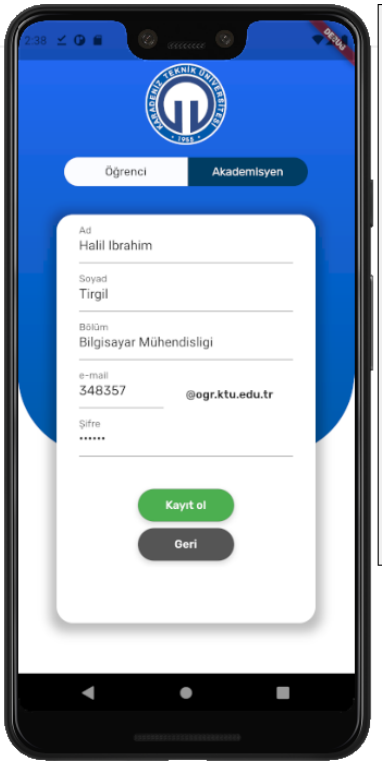
```
class SignInValidation with ChangeNotifier {
  ValidationItem _email = ValidationItem(null, null);
  ValidationItem _password = ValidationItem(null, null);
  ValidationItem _name = ValidationItem(null, null);
  ValidationItem _surname = ValidationItem(null, null);
  ValidationItem _fieldOfStudy = ValidationItem(null, null);
  String _emailType = '@ogr.ktu.edu.tr';
  bool _toggleLoginLogout = false;

  ValidationItem get email => _email;
  ValidationItem get password => _password;
  String get emailtype => _emailType;
  bool get toggleLoginLogout => _toggleLoginLogout;
  ValidationItem get name => _name;
  ValidationItem get surname => _surname;
  ValidationItem get fieldOfStudy => _fieldOfStudy;

  void setEmail(String value) {
    if   (RegExp(r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'*+-/=?^_`{|}~]+@[a-zA-Z0-9]+\.[a-
zA-Z]+")
            .hasMatch(value) &&
        value.endsWith('@ogr.ktu.edu.tr')) {
      _email = ValidationItem(value, null);
    } else {
      _email = ValidationItem(null, 'e-mail geçersiz');
    }
    notifyListeners();
  }

  void setName(String value) {
    if (RegExp(r'^[a-zA-Z]+(\s[a-zA-Z]+)?$').hasMatch(value)) {
      _name = ValidationItem(value, null);
    } else {
      _name = ValidationItem(null, 'İsimde sadece karakter olabilir');
    }
    notifyListeners();
  }

  void setSurname(String value) {
    if (RegExp(r'^[A-Za-z-ğüşöçİĞÜŞÖÇ]+$').hasMatch(value)) {
      _surname = ValidationItem(value, null);
    } else {
      _surname = ValidationItem(null, 'Soyadında sadece karakter olabilir');
    }
    notifyListeners();
  }

  void setFieldOfStudy(String value) {
    if (value.length > 5) {
      _fieldOfStudy = ValidationItem(value, null);
```

```
    } else {
      _fieldOfStudy =
          ValidationItem(null, 'Bölüm 5 karakterden fazla olmalıdır.');
    }
    notifyListeners();
  }

  void setEmailType(int index) {
    index == 1 ? _emailType = '@ktu.edu.tr' : _emailType = '@ogr.ktu.edu.tr';
    notifyListeners();
  }

  void setPassword(String value) {
    if (value.length < 6) {
      _password =
          ValidationItem(null, 'Şifre 6 karakterden fazla olması lazım.');
    } else {
      _password = ValidationItem(value, null);
    }
    notifyListeners();
  }

  void setToggleLoginLogout() {
    // To reset the fields when toggling
    _email = ValidationItem(null, null);
    _password = ValidationItem(null, null);
    _name = ValidationItem(null, null);
    _surname = ValidationItem(null, null);
    _fieldOfStudy = ValidationItem(null, null);

    _toggleLoginLogout = !_toggleLoginLogout;
    notifyListeners();
  }
}
```

Bu kodlamalar sonucunda mobil cihazda elde edilen sayfa görünümleri ise şu şekilde gösterilebilir. (Şekil 2.1 ve Şekil 2.2)



Şekil 2.1 Akademisyen Girişi



Şekil 2.2 Ögrenci Kaydı

Yukarıda uygulama içi iki farklı ekran gösterilmektedir. Kullanıcı akademisyen girişi yapmak isterse mail otomatik olarak diğer mail modeline göre ayarlanır. Yine aynı şekilde kullanıcı öğrenci kaydı yapmak isterse mail öğrenci maili olarak ayarlanmış olur.

2.2. Ana Sayfa Ekranı

Projedeki Ana Sayfa mimarisinin işleyişi Gezinti Çubuğu üzerinde işlevselleştirilmiştir. Bu mimaride kaynak olarak belirtilen sayfamız ana sayfadır. Ana sayfa değiştiği an geçerli indeks değerimiz diğer değişen sayfaya eşit olacak şekilde ayarlanıyor. Ana sayfada kullanıcılar bulunduğu grupta paylaşılan gönderiler gösterilmektedir. (Şekil 2.3)

Yapılan bu değişiklikleri anlık olarak ekrana göstermek için ise Provider paketi özelliklerinden yararlanılıyor. Provider paketi sayesinde sayfa mimarisi daha düzenli bir şekilde geliştirilmiş oluyor.

Tablo 2.5 Ana Sayfa Kaynak Kodu

```
class Root extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Consumer<AuthProvider>(builder: (context, authProvider, _) {
      if (authProvider.isSignedIn) {
        return _buildScreens();
      } else {
```

```
        return SignInSignUp();
      }
    });
  }
}

Widget _buildScreens() {
  return Builder(
    builder: (BuildContext context) => Consumer<NavigationProvider>(
      builder: (context, navigationProvider, child) {
        // Create bottom navigation bar items from screens.
        final bottomNavigationBarItems = navigationProvider.screens
            .map((screen) => BottomNavigationBarItem(
                icon: _buildNavigationBarIcons(screen.label),
                label: screen.label))
            .toList();

        // Initialize [Navigator] instance for each screen.
        final screens = navigationProvider.screens
            .map(
              (screen) => Navigator(
                key: screen.navigatorState,
                onGenerateRoute: screen.onGenerateRoute,
              ),
            )
            .toList();

        return WillPopScope(
          onWillPop: () async => navigationProvider.onWillPop(context),
          child: Scaffold(
            drawer: Drawer(
              child: BuildDrawerBar(context),
            ),
            body: IndexedStack(
              children: screens,
              index: navigationProvider.currentTabIndex,
            ),
            bottomNavigationBar: BottomNavigationBar(
              items: bottomNavigationBarItems,
              currentIndex: navigationProvider.currentTabIndex,
              onTap: (tab)  {
                navigationProvider.setTab(tab, context);
              },
              type: BottomNavigationBarType.fixed,
              backgroundColor: KtuColors.ktuLightBlue,
              // Doesn't work.
              unselectedItemColor: KtuColors.ktuDarkBlue,
              selectedItemColor: KtuColors.ktuWhite,
              showSelectedLabels: false,
              showUnselectedLabels: false,
            ),
          ),
        );
      },
    ),
  );
}

Widget _buildNavigationBarIcons(String label) {
```

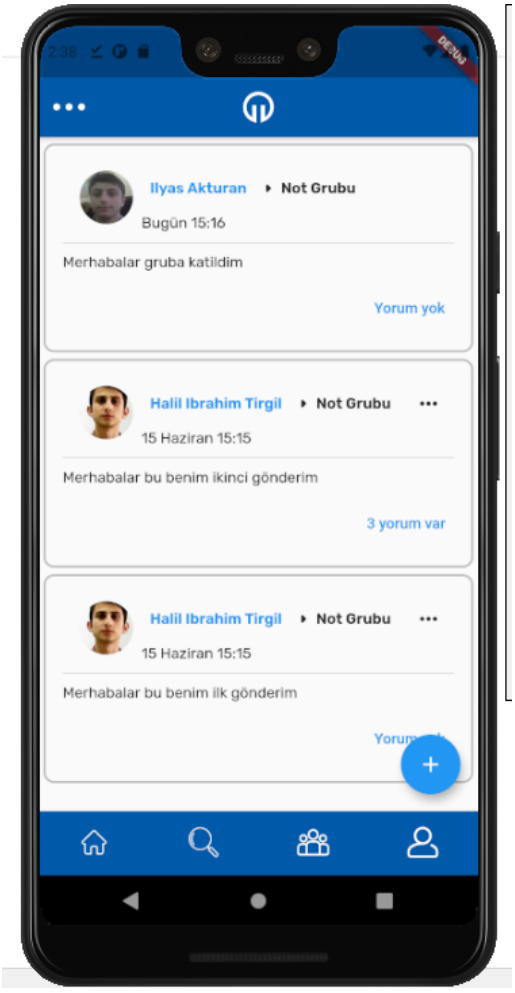
```
  switch (label) {
    case 'Ana Sayfa':
      return SvgPicture.asset(LogoConstants.homeIcon, width: 23.0);
    case 'Arama':
      return SvgPicture.asset(LogoConstants.searchIcon, width: 28.0);
    case 'Duyuru':
      return SvgPicture.asset(LogoConstants.notificationIcon,
          width: 24, height: 30, color: Colors.white);
    case 'Topluluk':
      return SvgPicture.asset(LogoConstants.groupIcon, width: 29.0);
    default:
      return SvgPicture.asset(LogoConstants.homeIcon, width: 23.0);
  }
}
```

Tablo 2.6 Gezinti Çubuğu Provider Kaynak Kodu

```
const HOME_SCREEN = 0;
const SEARCH_SCREEN = 1;
const NOTIFICATION_SCREEN = 2;
const GROUP_SCREEN = 3;

class NavigationProvider with ChangeNotifier {
  static NavigationProvider of(BuildContext context) =>
      Provider.of<NavigationProvider>(context, listen: false);

  int _currentScreenIndex = HOME_SCREEN;

  final Map<int, Screen> _screens = {
    HOME_SCREEN: Screen(
      label: 'Ana Sayfa',
      child: HomeScreen(),
      initialRoute: HomeScreen.route,
      navigatorState: GlobalKey<NavigatorState>(),
      onGenerateRoute: (settings) {
        print('Generating route: ${settings.name}');
        switch (settings.name) {
          case CommentScreen.route:
            return MaterialPageRoute(
                builder: (_) => CommentScreen(
                    map: settings.arguments as Map<String, String>?));
          case ProfileScreen.route:
            return MaterialPageRoute(
                builder: (_) => ProfileScreen(
                    map: settings.arguments as Map<String, String>?));
          case NotificationScreen.route:
            return MaterialPageRoute(builder: (_) => NotificationScreen());
          case GroupScreen.route:
            return MaterialPageRoute(builder: (_) => GroupScreen());
          case SettingsScreen.route:
            return MaterialPageRoute(builder: (_) => SettingsScreen());
          case HelpScreen.route:
            return MaterialPageRoute(builder: (_) => HelpScreen());
          case SettingsAboutScreen.route:
            return MaterialPageRoute(builder: (_) => SettingsAboutScreen());
          case SettingsAccountScreen.route:
            return MaterialPageRoute(builder: (_) => SettingsAccountScreen());
          case SettingsNotificationScreen.route:
```

```
            return MaterialPageRoute(
                builder: (_) => SettingsNotificationScreen());
          case SettingsSecurityScreen.route:
            return MaterialPageRoute(builder: (_) => SettingsSecurityScreen());
          case SettingsThemeScreen.route:
            return MaterialPageRoute(builder: (_) => SettingsThemeScreen());
          default:
            return MaterialPageRoute(builder: (_) => HomeScreen());
        }
      },
      scrollController: ScrollController(),
    ),

// -> HOME_SCREEN için yapılan routing diğer 3 tab için de yapılır //

  int get currentTabIndex => _currentScreenIndex;
  List<Screen> get screens => _screens.values.toList();
  Screen get currentScreen => _screens[_currentScreenIndex]!;

  void setTab(int tab, context) async {
    if (tab == currentTabIndex) {
      Navigator.pushReplacement(context,
          MaterialPageRoute(builder: (BuildContext context) => Root()));
      _scrollToStart(tab);
    } else {
      if (tab == 1) {
        Provider.of<GroupProvider>(context, listen: false).getAllGroups();
      }
      _currentScreenIndex = tab;
      notifyListeners();
    }
  }

  Route<dynamic> onGenerateRoute(RouteSettings settings) {
    print('Generating route: ${settings.name}');
    switch (settings.name!) {
      // We can push another screen  here, like settings
      case SettingsScreen.route:
        return MaterialPageRoute(builder: (_) => SettingsScreen());
      case HelpScreen.route:
        return MaterialPageRoute(builder: (_) => HelpScreen());
      case ProfileScreen.route:
        return MaterialPageRoute(builder: (_) => ProfileScreen());
      case MessageScreen.route:
        return MaterialPageRoute(builder: (_) => MessageScreen());
      default:
        return MaterialPageRoute(builder: (_) => Root());
    }
  }

  Future<bool> onWillPop(BuildContext context) async {
    final currentNavigatorState = currentScreen.navigatorState.currentState;

    if (currentNavigatorState!.canPop()) {
      currentNavigatorState.pop();
      return false;
    } else {
      if (currentTabIndex != HOME_SCREEN) {
        setTab(HOME_SCREEN, context);
```

```
        notifyListeners();
        return false;
      } else {
        SystemChannels.platform.invokeMethod('SystemNavigator.pop');
        return false;
      }
    }
  }

  void _scrollToStart(int tab) async {
    if (currentScreen.scrollController != null &&
        currentScreen.scrollController!.hasClients) {
      await currentScreen.scrollController!.animateTo(
        0.0,
        curve: Curves.easeOut,
        duration: const Duration(milliseconds: 300),
      );
    }
  }
}
```



Şekil 2.3 Ana Sayfa Modeli

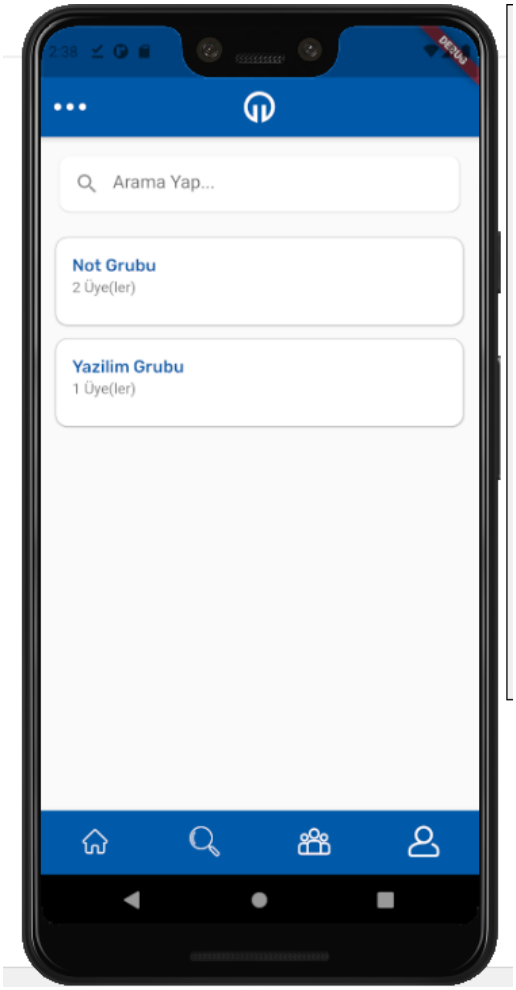
2.3. Arama Sayfası

Uygulama kullanıcılarının kurmuş olduğu grupların gösterildiği sayfadır. Bu sayfa yardımıyla yeni gruplar keşfedilebilir ve katılma işlemi gerçekleştirilebilir. Katılmış olunan gruplarda gönderi paylaşımı yapılabilir. Sayfanın kaynak kodu tabloda (Tablo 2.7), uygulama içi ekran görüntüsü ise şekilde (Şekil 2.4) gösterildiği gibidir.

Tablo 2.7 Arama Sayfası Kaynak Kodu

```
class SearchScreen extends StatefulWidget {
  static const route = '/search';

  @override
  _SearchScreenState createState() => _SearchScreenState();
}

class _SearchScreenState extends State<SearchScreen> {
  RefreshController _refreshController =
      RefreshController(initialRefresh: false);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      drawer: Drawer(
        child: BuildDrawerBar(context),
      ),
      appBar: AppBar(
        automaticallyImplyLeading: false,
        backgroundColor: KtuColors.ktuLightBlue,
        title: SvgPicture.asset(LogoConstants.appBarLogo, width: 30.0),
        leading: Builder(
          builder: (BuildContext context) => IconButton(
            icon: SvgPicture.asset(
              LogoConstants.drawerIcon,
              width: 30.0,
            ),
            onPressed: () => Scaffold.of(context).openDrawer(),
          ),
        ),
        centerTitle: true,
      ),
      body: SmartRefresher(
        enablePullDown: true,
        enablePullUp: true,
        controller: _refreshController,
        onRefresh: _onRefresh,
        onLoading: _onLoading,
        child: SingleChildScrollView(
          child: Column(
            children: [
              searchBar(),
              SearchScreenGroups(),
            ],
          ),
        ),
      ),
    );
```

```
  }

searchBar() {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Card(
        margin: EdgeInsets.all(10),
        //color: Colors.white,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(10.0),
        ),
        child: Padding(
          padding: const EdgeInsets.all(2.0),
          child: TextField(
            decoration: InputDecoration(
              hintStyle: TextStyle(fontSize: 17),
              prefixIcon: Icon(Icons.search),
              border: InputBorder.none,
              hintText: 'Arama Yap...',
            ),
            onChanged: (text) {
              text = text.toLowerCase();
              setState(() {
                Provider.of<GroupProvider>(context, listen: false)
                    .filterGroups(text);
              });
            },
          ),
        ),
      ),
    );
  }

  void _onRefresh() {
    Provider.of<GroupProvider>(context, listen: false).getAllGroups();
    setState(() {});
    _refreshController.refreshCompleted();
  }

  void _onLoading() {}

  @override
  void dispose() {
    // TODO: implement dispose
    _refreshController.dispose();
    super.dispose();
  }
}
```

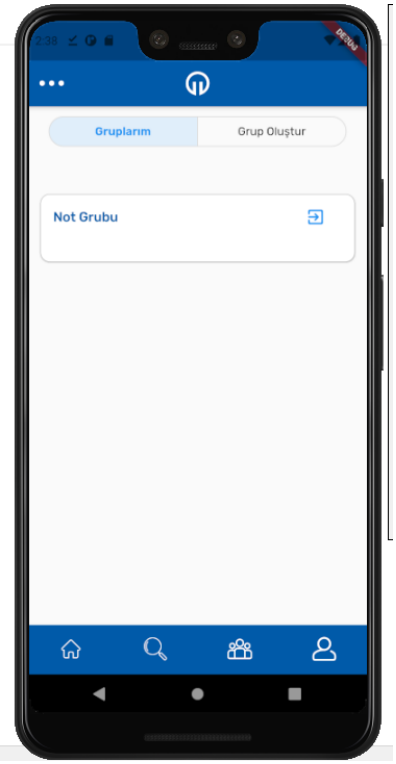


Şekil 2.4 Arama Sayfası

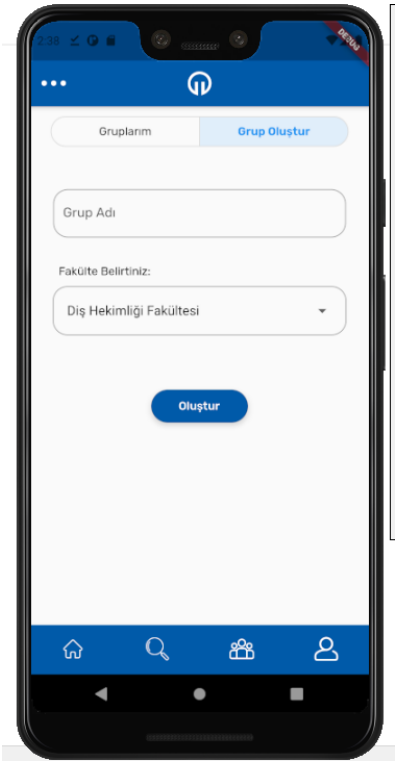
2.4. Gruplarım ve Grup Oluştur Sayfası

Gezinti çubuğunun bir elemanı olan bu sayfa iki kısımdan oluşmaktadır. Gruplarım kısmı oluşturulmuş olan grupları görme ve yöneticisi olunan grupların silinebildiği bir ekrandır. Bu ekranın görüntüsü şekildeki (Şekil 2.5) gibidir.

Grup oluştur kısmında ise kullanıcı, grup adı ve hangi fakülteden olduğunu belirterek yeni bir topluluk oluşturup diğer kullanıcılarla birlikte etkileşime geçebilmektedir. Bu ekranın görüntüsü şekildeki (Şekil 2.6) gibidir.



Şekil 2.5 Gruplarım Sayfası



Şekil 2.6 Gruplarım Sayfası

2.5. Çekmece Çubuğu Mimarisi

Yapılan çekmece çubuğu, ara yüze dökülmüş hali ve kod yapısı aşağıdaki gibidir.

Tablo 2.8 Çekmece Çubuğu Kaynak Kodu

```
class BuildDrawerBar extends StatelessWidget {
  BuildDrawerBar(BuildContext context);
  @override
  Widget build(BuildContext context) {
    var authProvider = Provider.of<AuthProvider>(context, listen: false);
    var _lineColor = Colors.grey.shade400;
    var drawerHeader = DrawerHeader(
      child: Container(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: <Widget>[
            Row(
              children: <Widget>[
                Container(
                  margin: EdgeInsets.fromLTRB(4.0, 20.0, 16.0, 0.0),
                  width: MediaQuery.of(context).size.width / 7,
                  height: MediaQuery.of(context).size.height / 12,
                   child: CachedNetworkImage(
                    imageBuilder: (context, imageProvider) => Container(
                      decoration: BoxDecoration(
                        shape: BoxShape.circle,
                        image: DecorationImage(
                            image: imageProvider, fit: BoxFit.cover),
```

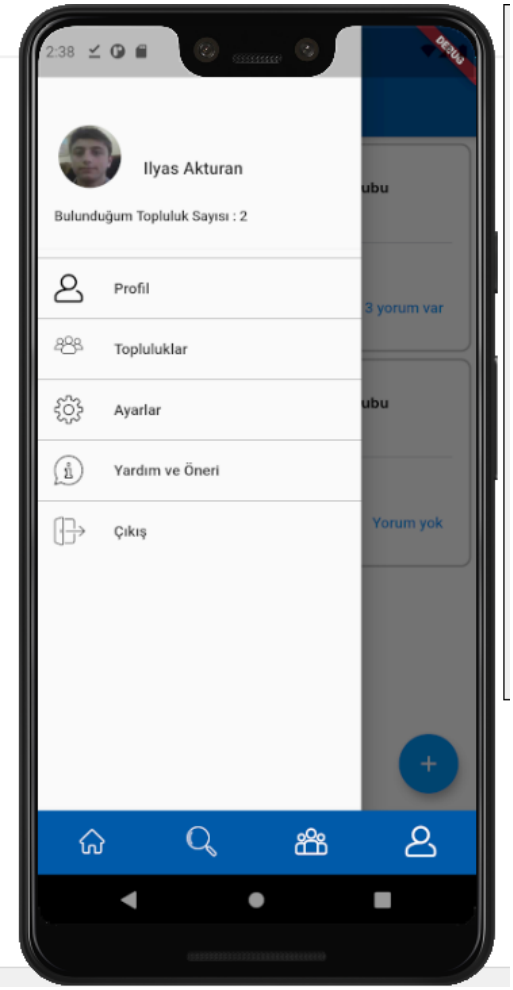
```
                      ),
                    ),
                    imageUrl: authProvider.currentUser.photoURL!,
                    placeholder: (context, url) => CircularProgressIndicator(),
                    errorWidget: (context, url, error) => Icon(Icons.error),
                  ),
                ),
                Padding(
                  padding: EdgeInsets.fromLTRB(5.0, 50.0, 0.0, 8.0),
                   child: Text(
                    authProvider.currentUser.name!,
                    style: TextStyle(fontSize: 16.0),
                  ),
                ),
              ],
            ),
            Padding(
              padding: EdgeInsets.fromLTRB(0.0, 10.0, 0.0, 10.0),
              child: Row(
                children: <Widget>[
                   Text(
                    "Bulunduğum           Topluluk           Sayısı           :
${authProvider.currentUser.groups!.length}",
                    style: TextStyle(fontSize: 13.0),
                  )
                ],
              ),
            )
          ],
        ),
      ),
    );

    return ListView(
      children: <Widget>[
        drawerHeader,
        Container(
          decoration:
              BoxDecoration(border: Border(top: BorderSide(color: _lineColor))),
          child: ListTile(
            leading: SvgPicture.asset(
              LogoConstants.profileIcon2,
              width: 27.0,
            ),
            title: Text('Profil'),
            onTap: () {
              Navigator.pop(context);
              Navigator.pushNamed(context, '/profile',
                  arguments: <String, String>{
                    'userId': authProvider.currentUser.id!
                  });
            },
          ),
        ),
        Container(
          decoration:
              BoxDecoration(border: Border(top: BorderSide(color: _lineColor))),
          child: ListTile(
            leading: SvgPicture.asset(
```

```
              LogoConstants.notificationIcon,
              width: 27.0,
            ),
            title: Text('Duyurular'),
            onTap: () {
              Navigator.pop(context);
              Navigator.pop(context);
              Navigator.pushNamed(context, "/notification");
            },
          ),
        ),
        Container(
          decoration:
              BoxDecoration(border: Border(top: BorderSide(color: _lineColor))),
          child: ListTile(
            leading: SvgPicture.asset(
              LogoConstants.communityIcon,
              width: 27.0,
            ),
            title: Text('Topluluklar'),
            onTap: () {
              Navigator.pop(context);
              Navigator.pop(context);
              Navigator.pushNamed(context, "/group");
            },
          ),
        ),
        Container(
          decoration:
              BoxDecoration(border: Border(top: BorderSide(color: _lineColor))),
          child: ListTile(
            leading: SvgPicture.asset(
              LogoConstants.settingsIcon,
              width: 27.0,
            ),
            title: Text('Ayarlar'),
            onTap: () {
              Navigator.pop(context);
              Navigator.pushNamed(context, '/settings');
            },
          ),
        ),
        Container(
          decoration:
              BoxDecoration(border: Border(top: BorderSide(color: _lineColor))),
          child: ListTile(
            leading: SvgPicture.asset(
              LogoConstants.helpIcon,
              width: 27.0,
            ),
            title: Text('Yardım ve Öneri'),
            onTap: () {
              Navigator.pop(context);
              Navigator.pushNamed(context, '/help');
            },
          ),
        ),
        Consumer<AuthProvider>(builder: (context, authProvider, _) {
          return Container(
```

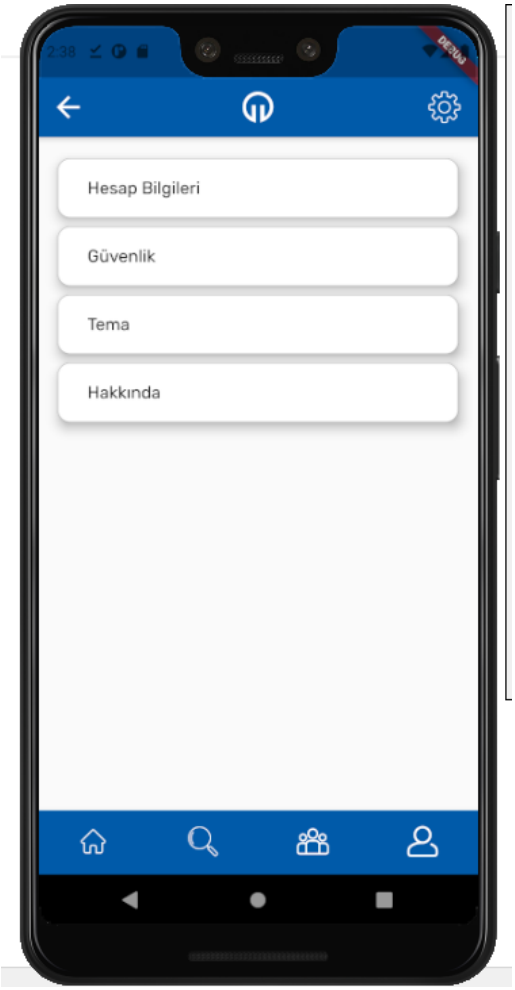
```
            decoration: BoxDecoration(
                border: Border(top: BorderSide(color: _lineColor))),
            child: ListTile(
              leading: SvgPicture.asset(
                LogoConstants.logoutIcon,
                width: 29.0,
              ),
              title: Text('Çıkış'),
              onTap: () async {
                authProvider.logOut();
              },
            ),
          );
        }),
      ],
    );
  } }
```

Yukarıdaki kod ile yapılan birkaç işlemden şöyle bahsedebiliriz. Ana sayfada bulunan üç nokta ikonuna basıldığı an açılacak şekilde ayarlanan bu sayfada bir ListView widgeti kullanıldı. Bu widget yardımı ile kaç adet eleman oluşturulmak istenirse o kadar eleman bu widgeta dahil edilir. İşlevselleştirmek için ise onTap fonksiyonlarını kullanmak gerekmektedir. Bu sayede açılan çekmece çubuğunda (Şekil 2.7) gösterilen sayfalara gidilebilir.

Ayarlar sayfasında kullanıcıyı kendi hesap bilgilerinin olduğu bir sayfa, güvenlik kısmından şifre değişimi yapabildiği bir sayfa, uygulamayı koyu temaya geçirebileceği bir sayfa ve uygulama hakkında bilgiler barındıran bir sekme karşılar. (Şekil 2.8)



Şekil 2.7 Çekmece Çubuğu



Şekil 2.8 Ayarlar Sayfası

2.6. Profil Sayfası

Çekmece çubuğundan erişilebilen profil sayfasında, kullanıcı, kendi profilini ve dahil olduğu gruplarda paylaştığı gönderilerin var olduğu bir ekranla karşılaşır. Bu sayfada kullanıcının kişisel bilgilerinin bazıları da yer alır. (Şekil 2.9)

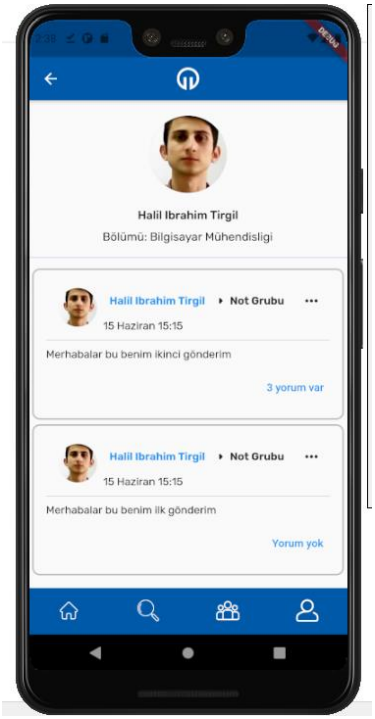
Kullanıcılar kendi profillerini görüntüleyebildikleri gibi dahil oldukları gruplardaki kullanıcıların profillerini de görüntüleyebilirler. Bu sayfanın kaynak kodu ise aşağıdaki tablodaki (Tablo 2.9) gibidir.

Tablo 2.9 Profil Kaynak Kodu

```
class ProfileScreen extends StatefulWidget {
  static const route = '/profile';
  final Map<String, String>? map;

  const ProfileScreen({Key? key, this.map}) : super(key: key);
```

```
  @override
  _ProfileScreenState createState() => _ProfileScreenState();
}
class _ProfileScreenState extends State<ProfileScreen> {
  void initState() {
    Provider.of<AuthProvider>(context, listen: false).setMyPosts();
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    UserRepository userRepository = UserRepository();
    final userId = widget.map!['userId'];
    return Scaffold(
      appBar: AppBar(
        title: SvgPicture.asset(LogoConstants.appBarLogo, width: 30.0),
        centerTitle: true,
        backgroundColor: KtuColors.ktuLightBlue,
      ),
      body: FutureBuilder(
        future: userRepository.getUser(userId!),
        builder: (BuildContext context, AsyncSnapshot snapshot) {
          if (snapshot.data == null) {
            print("if");
            return Center(
              child: CircularProgressIndicator(),
            );
          } else {
            print("else");
            print(snapshot.data.name);
            return SingleChildScrollView(
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  buildProfilePhoto(context, snapshot.data.photoURL),
                  buildProfileInformation(context, snapshot.data.name),
                  Divider(
                    color: KtuColors.ktuGradientStart,
                  ),
                  SingleChildScrollView(
                    child: ListView.builder(
                      primary: false,
                      shrinkWrap: true,
                      itemCount: userRepository.posts.length,
                      itemBuilder: (context, index) {
                        if (userRepository.posts.length == 0) {
                          Text("Gönderi Yok");
                        }
                        return PostWidget(
                            post: userRepository.posts[index],
                            groupId: userRepository.posts[index].group!.id);
                      },
                    ),
                  )
                ],
              ),
            );
          }
        },
```

```
      ),
    );
  }
}

Widget buildProfilePhoto(BuildContext context, String photoURL) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 20.0),
    child: Center(
      child: CircleAvatar(
        radius: 50,
        child: CachedNetworkImage(
          imageBuilder: (context, imageProvider) => Container(
            decoration: BoxDecoration(
              shape: BoxShape.circle,
              image: DecorationImage(image: imageProvider, fit: BoxFit.cover),
            ),
          ),
          imageUrl: photoURL,
          placeholder: (context, url) => CircularProgressIndicator(),
          errorWidget: (context, url, error) => Icon(Icons.error),
        ),
      ),
    ),
  );
}

Widget buildProfileInformation(BuildContext context, String name) {
  var authProvider = Provider.of<AuthProvider>(context);
  return Column(
    children: [
      Text(
        name,
        style: TextStyle(
          fontFamily: 'Rubik',
          fontSize: 15,
          fontWeight: FontWeight.w500,
        ),
      ),
      SizedBox(height: MediaQuery.of(context).size.height / 80),
      Text(
        "Bölümü: " + authProvider.currentUser.fieldOfStudy!,
        style: TextStyle(
          fontFamily: 'Rubik',
          fontSize: 15,
          fontWeight: FontWeight.w400,
        ),
      ),
      SizedBox(height: MediaQuery.of(context).size.height / 80),
      buildButton(),
    ], ); }
```



Şekil 2.9 Profil Sayfası

2.7. Yardım ve Öneri Sayfası

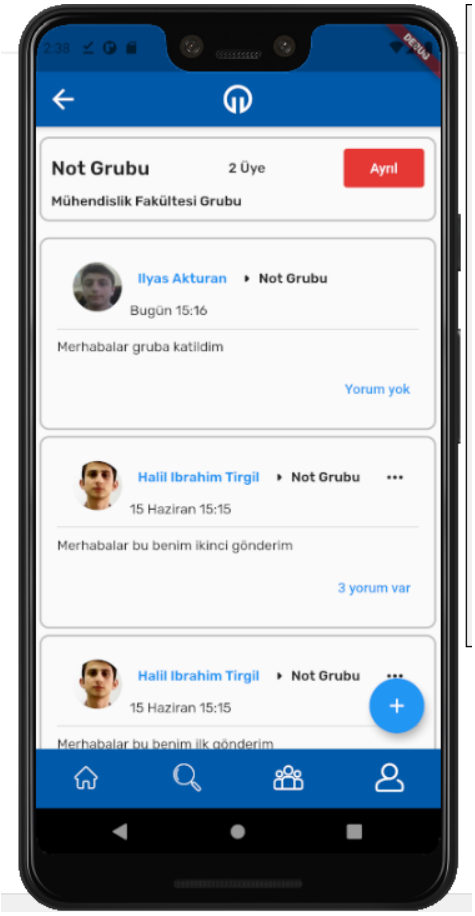
Çekmece çubuğundan erişilen yardım ve öneri kısmında ise kullanıcıların uygulamayı daha kullanışlı hale getireceğini düşündükleri herhangi bir düşünceyi ya da yaşadığı bir sorunu raporlayabileceği bir sayfa vardır. (Şekil 2.10)

Şekil 2.10 Yardım Öneri Sayfası

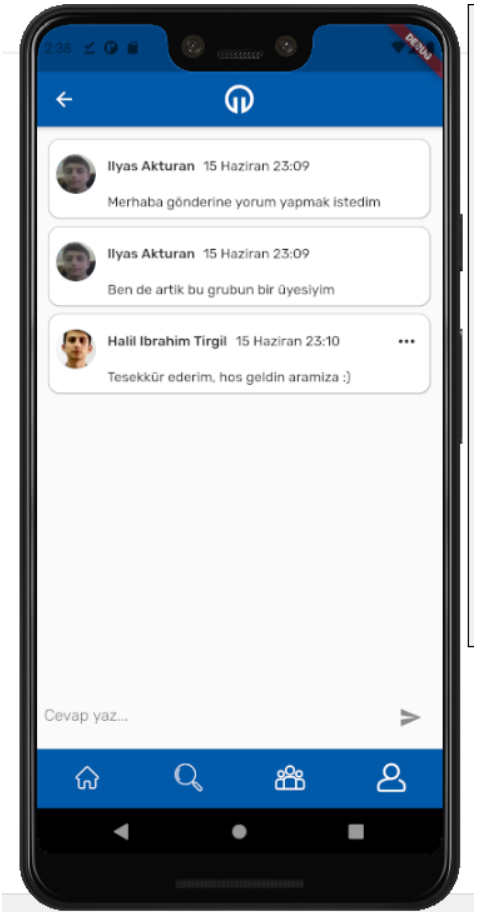
2.8. Grup Detayları Sayfası

Arama sayfasındaki oluşturulmuş grupların üstüne basıldığı zaman açılan bu sayfada ilk olarak kullanıcıyı grup bilgisi ve grupta paylaşılan gönderiler karşılar. Kullanıcılar paylaştıkları gönderileri ya da yaptıkları yorumları istedikleri durumlarda silebilir. Şekilde (Şekil 2.11) görüldüğü gibi.

Bu sayfada herhangi bir gönderinin altına bir yorum eklenebilir. Eğer gönderilere herhangi bir yorum eklenmişse, bunlar, gönderide bulunan yorum sayısını gösteren alana basılarak görüntülenebilir. (Şekil 2.12)



Şekil 2.11 Grup Sayfası

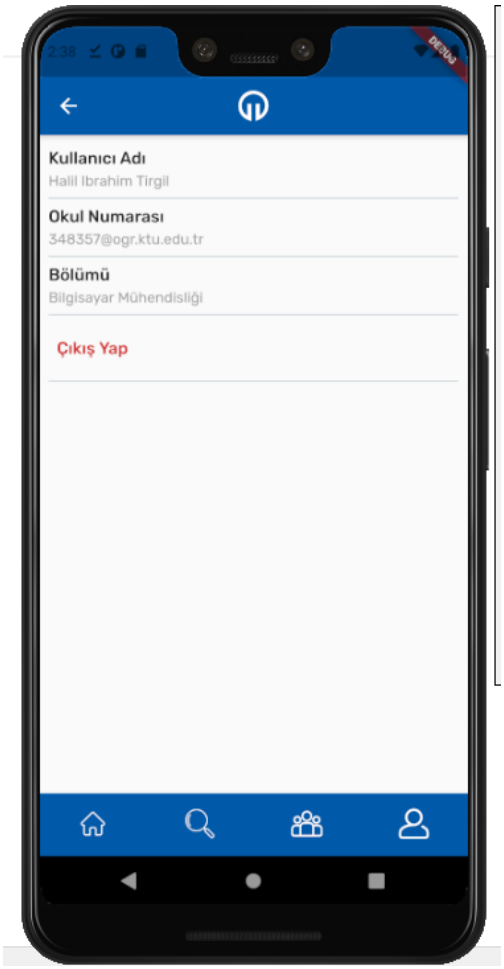


Şekil 2.12 Yorumlar Sayfası

2.9. Çıkış Yapma

Uygulamadan çıkış yapma iki farklı yerden gerçekleştirilebilir. Bunların birincisi çekmece çubuğundaki çıkış yap butonudur.

İkincisi ise ayarlar kısmında hesap bilgileri sayfasına girildiğinde kullanıcının karşısına çıkar ve yine bu buton üzerinden uygulamadan çıkış yapılabilir. (Şekil 2.13)



Şekil 2.13 Hesap Bilgileri Çıkış Yapma

3. SONUÇLAR

Sonuçlara bakılacak olursa ilk olarak grup üyeleri proje yönetimi nasıl yapılır, proje geliştirme aşamalarında neler yapılır öğrenildi. Kullanılan bazı online platformlar üzerinden proje tasarım aşamaları gerçekleştirildi. Örneğin Trello'da görev paylaşımları yapıldı, GitLab üzerinde proje senkron bir şekilde geliştirilmeye devam etti, Figma üzerinde her bir grup üyesi önce mobil uygulamanın görsel tasarımını gerçekleştirdi.

Mobil uygulama geliştirme aşamalarında neler yapılmalı sorusuna doğru cevapların neler olduğu öğrenildi. Örnek verilecek olursa veri tabanı, API ve uygulama arasındaki bağlantılar ve bu bağlantıların nasıl ve neden kurulduğu öğrenildi. İlişkisel tablolar çıkarılarak ve birbirleriyle ilişkilendirilerek uygulamadaki mimari mantık geliştirildi.

Front-end'i back-end'e bağlarken kullanılan HTTP metodlarını ve nerede ne için kullanıldıkları öğrenildi. Aynı zamanda her mobil cihaza uygun bir şekilde responsive kodlama öğrenildi.

4. ÖNERİLER

Geliştirilen mobil uygulamaya sosyal medya uygulamalarındaki güncel gelişmeler takip edilerek mantıklı eklemeler yapılabilir. Ayrıca henüz uygulamada yer almaya kişiler arası özel mesaj özelliği de uygulamaya entegre edilebilir.

Uygulama içi bildirimler kullanıcı inaktifken kullanıcıya gönderilebilir.

İlerleyen zamanlarda fotoğraf paylaşımının üstüne dosya, pdf, docx ve buna benzer birçok farklı dosya türü ekleme özelliği getirilebilir.

Geniş kapsamda uygulama gelişime ve değişime çok müsait bir şekilde tasarlanmıştır ve bunlar entegre edilebilir

5. EKLER

5.1 Disiplinler Arası Çalışma

Grup üyelerinin çalıştay programı kapsamında yaptığı çalışmaların içerikleri aşağıdaki gibi gösterilebilir.

Halil İbrahim Tirgil Çalıştay Raporu:

KARADENİZ TEKNİK ÜNİVERSİTESİ

BİTİRME PROJESİ GİRİŞİMCİLİK ÇALIŞTAYI RAPORU

Rapor Tarihi: 17.05.2021

Grup No: G3

Grup Üyeleri:

348357- Halil İbrahim TİRGİL (Bilgisayar Mühendisliği)

295098-Burak KAYA (Bilgisayar Mühendisliği)

348369-Cihan AYTAŞ (Bilgisayar Mühendisliği)

365196- Ali KARATUT (Elektrik-Elektronik Mühendisliği)

365156-Ayşenur YILDIRIM (Elektrik-Elektronik Mühendisliği)

365173-Bahadır ÖZTÜRK (Elektrik-Elektronik Mühendisliği)

375795-Alperhan GÜNER (Elektrik-Elektronik Mühendisliği)

Grup Sorumlusu: Arş. Gör. Şeyma AYMAZ

1-Problem:

Kampüs veya belirli bir alan içerisindeki fazla enerji harcanması .

Elektrik santrallerinde oluşabilecek kesinti durumlar.(Yangın vb.)

Elektrik tesisleri, telleri gibi elemanların bakım maliyeti.

İyileşen Özellik: Enerji tüketiminin minimuma indirilmesi.

22.Enerji Kaybı: Yapılan işe katılmayan enerjinin kullanımı enerji kaybını azaltmak bazen enerji kullanımını iyileştirmekten başka teknikler gerektirdiği için bu ayrı bir kategoridir.

Kötüleşen Özellik: Mevcut sistemin iptalinin maliyeti ve yeni sistemin kurulum maliyeti. Güneş Işınlarının az olduğu günlerde enerji üretiminin azalması.

18. Aydınlatma Şiddeti: Birim alan başına düşen ışık akışı ve ayrıca aydınlık ışık kalitesi ve diğer aydınlatma özellikleri

Çelişki:Güneş panellerinin yeterli ışık alamadığı zamanlarda verimli bir şekilde çalışamaması ve kaynak sağlayamaması.(Çelişki Matrisinde 22 x 18nolu matris hücresinde önerilen yaratıcı ilkeler şu şekildedir: 1, 13, 15, 32)

1. Bölümleme

a. Bir cismin bağımsız parçalara bölünmesi

b. Bir cismin bölümlü hale getirilmesi

c. Bir cismin bölümlenme derecesinin artırılması

13. Ters eylem

a. Problemin özelliklerinin dayattığı bir eylem yerine zıt bir eylemin yerine getirilmesi

b. Cismin hareket eden bir parçasının ya da dış ortamın hareketsiz, hareketsiz parçanın ise hareketli hale getirilmesi

c. Cismin ters-yüz edilmesi

15. Dinamiklik

a. Cismin ya da bulunduğu ortamın çalışmasının her aşamasında optimal performans için otomatik olarak uyarlanması

b. Bir cismin birbirine göre konum değiştirebilecek öğelere bölünmesi

c. Bir cisim hareketsizse onun hareketli ya da değiştirilebilir hale getirilmesi

32. Rengi değiştirme

a. Bir cismin ya da çevresinin renginin değiştirilmesi

b. Görülmesi güç bir cismin ya da süreçlerin saydamlık derecesinin değiştirilmesi

c. Görülmesi güç cisimlerin ya da süreçlerin gözlemlenebilmesi için renkli katkı maddelerinin kullanılması

d. Bu tür katkı maddeleri halihazırda kullanılıyorsa, parlak izlerin ya da iz bırakan unsurlarınkullanılması

ÇÖZÜM: Güneş panellerinin daha iyi güneş ışığı alabileceği bir yerde konumlandırılıp, üretilen fazla enerjinin aküler yardımıyla depolanması.

2- Projenin Kapsamı:

Kampüste ve aydınlatılacak bölgede bulunan kişiler

3-Projenin Değer Önerisi:

Daha az enerji kullanılarak kampüsün aydınlatılması sağlanacak ve harekete duyarlı sensör yardımıyla enerji kullanımı azalacaktır. Böylelikle az enerjiyle aydınlatma sağlanacak hem de kaynaklar dengeli bir şekilde kullanılmış olacaktır.

4- Çözüm Önerisi:

Aydınlatma direklerinin üzerlerine yerleştirilen hareketli sensörler ve güneş paneli yardımıyla enerji üretilerek gereksiz enerji tüketimi azaltılacaktır. Güneş enerji sistemlerinin çevreye zararlı olmamasından dolayı güneş paneli kullanımı uygun görülmüştür.

5- Yapılacak projenin çalışması:

Bölgeye kurulan güneş panelleri yardımıyla, güneşten gelen ışınlarla birlikte elektrik üretilecek ve bu sayede bölgenin aydınlatılması sağlanacaktır. Kullanılacak sensör sayesinde ise herhangi bir hareket olmadığı durumlarda aydınlatma duracak ve böylelikle enerji tasarrufu sağlanmış olacaktır.

6-Projenin yenilikçi yönü:

Daha büyük santraller kullanmadan enerji üretimi.

Eski cihazlarda sensör kullanılmadığından, sensörentegre edilerek daha verimli kullanılabilir.

Led lambalar kullanılarak daha az enerji kullanımı ile bölgenin aydınlatılması.

Harekete duyarlı sensör ile led lambaların ömrü uzayacak ve minimum seviyede enerji kullanımı olacaktır.

7-Projenin Uygulanabilirliği:

Bilgiler pratiktir ve uygulanabilirliği mevcuttur.

Sistemin öngörülebilir yollar ve koşullarda istenen fonksiyonu gerçekleştirebilme özelliği vardır.

Daha önceden uygulanmış bir sistem olduğundan dolayı deneme yanılma yöntemleri kullanılmasına gerek olmadan uygulanabilir.

6. KAYNAKLAR

1) https://tr.wikipedia.org/wiki/PostgreSQL PostgreSQL. 5 Haziran 2021

2) https://wmaraci.com/nedir/postgresql PostgreSQL. 5 Haziran 2021

3) http://berkay22demirel.blogspot.com Figma. 5 Haziran 2021

4) https://tr.wikipedia.org/wiki/TypeScript TypeScript. 6 Haziran 2021

5) https://alkanfatih.com/json-nedir-nasil-kullanilir-json-olusturma/ JSON. 15 Haziran 2021

6) https://tr.wikipedia.org/wiki/Git_(yazılım) Git. 16 Ocak 2021

7) https://siberci.com/flutter-nedir/ Flutter. 16 Ocak 2021

8) https://www.vojtech.net/posts/flutter-bottom-navigation/ Ana Sayfa Mimarisi. 17 Ocak 2021

9) https://www.figma.com Proje Dizayn Çizimi. 5 Haziran 2021

10) https://www.redhat.com/en/topics/devops/what-is-ci-cd CI/CD. 15 Haziran 2021

STANDARTLAR ve KISITLAR FORMU

Projenin hazırlanmasında uyulan standart ve kısıtlarla ilgili olarak, aşağıdaki soruları cevaplayınız.

1. Projenizin tasarım boyutu nedir? (Yeni bir proje midir? Var olan bir projenin tekrarı mıdır? Bir projenin parçası mıdır? Sizin tasarımınız proje toplamının yüzde olarak ne kadarını oluşturmaktadır?)

Proje yeni bir projedir. Tasarım olarak daha önce okul bünyesinde geliştirildiğini görmediğimiz bir proje. Fikir olarak % 80 bize ait bir projedir. Esinlendiğimiz birkaç yer mevcuttur.

2. Projenizde bir mühendislik problemini kendiniz formüle edip, çözdünüz mü? Açıklayınız.

Hayır.

3. Önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız?

Veri tabanı dersinde öğrendiğimiz ilişkisel tablolar, nesne yönelimli programlama dersinde öğrenmiş olduğumuz nesne tabanlı programlama ve mobil programlama için Windows Programlama dersinde öğrendiğimiz bilgiler ışığında proje geliştirildi.

4. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir? (Proje konunuzla ilgili olarak kullandığınız ve kullanılması gereken standartları burada kod ve isimleri ile sıralayınız).

Flutter framework yardımı ile yazılan projede DRY (Don’t Repeat Yourself) disiplini benimsendi ve optimize bir şekilde kod yazımına önem gösterildi. Proje dosyalarını adlandırırken “snake_case”, kodlama yaparken ise “camelCase” yöntemleri kullanıldı.

5. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen boşlukları uygun yanıtlarla doldurunuz.

a) Ekonomi

Eğer proje Play Store ya da App Store platformlarına yüklenecekse geliştirici hesap ücreti bir ekonomik kısıt olarak değerlendirilebilir.

b) Çevre sorunları:

Geliştirilen projenin çevreye herhangi bir olumsuz etkisi bulunmamaktadır.

c) Sürdürülebilirlik:

Geliştirilen proje kapsamında kullanıcılar her zaman yenileneceği için gelecek süreçler için kullanıcı kaybı yaşanmayacağı için sürdürülebilirlik açısından olumsuz bir durum gözükmemektedir.

d) Üretilebilirlik:

Mobil cihazlarda mobil uygulama ihtiyacı olduğundan ve üniversitelerde aktif öğrenciler her zaman olacağından dolayı üretilebilirlik düzeyi yüksektir.

e) Etik:

Geliştirilen proje etik konusunda hiçbir şüpheye ve probleme yer vermemektedir.

f) Sağlık:

Geliştirilen proje sağlık açısından olumsuzluk içermez.

g) Güvenlik:

Geliştirilen proje güvenlik açısından bir sorun içermez.

h) Sosyal ve politik sorunlar:

Geliştirilmiş olan proje sosyal ve politik bir sorun içermez.